Homotopy-Preserving, Optimal UAV-Trajectory Generation based on Spline Subdivision

Ruiqi Ni[1], Teseo Schneider[2], Daniele Panozzo[3], Zherong Pan[4], Xifeng Gao[1]

Abstract—Generating locally optimal UAV-trajectories is challenging due to the non-convex constraints of collision avoidance and actuation limits. We present the first local, optimization-based UAV-trajectory generator that simultaneously guarantees the validity and asymptotic optimality for known environments. *Validity:* Given a feasible initial guess, our algorithm guarantees the satisfaction of all constraints throughout the process of optimization. *Asymptotic Optimality:* We use an asymptotic exact piecewise approximation of the trajectory with an automatically adjustable resolution of its discretization. The trajectory converges under refinement to the first-order stationary point of the exact non-convex programming problem. Our method has additional practical advantages including joint optimality in terms of trajectory and time-allocation, and robustness to challenging environments as demonstrated in our experiments.

Index Terms—Motion planning, trajectory optimization, collision detection.

I. INTRODUCTION

Unmanned Aerial Vehicle (UAV) finds many real-world applications in inspection, search and rescue, and logistic automation. A key challenge in safe UAV-trajectory planning is to find locally optimal flying strategies, in terms of energy/time efficiency and smoothness, while accounting for various dynamic and kinematic constraints. A valid trajectory needs to satisfy two constraints to be physically realizable: the actuation limits must be respected (dynamic constraint), and the robot should maintain a safety distance from the boundary of the freespace (kinematic constraint). These two constraints combined define a non-convex and non-smooth feasible domain, which is notoriously difficult to handle for optimization-based motion planners [45], [29], [34].

Prior works tackle the non-convex, non-smooth constraints in one of three ways: infeasible recovery, relaxation, and global search. **Infeasible Recovery:** Nonlinear constrained optimizers, e.g. Sequential Quadratic Programming (SQP), have been used in [45], [40], [36] to directly handle non-convex constraints. These methods rely on the constraints' directional derivatives and penalty functions to pull infeasible solutions back to the feasible domain. However, their feasibility is not guaranteed: an example of a typical scenario of infeasible solutions involving an environment with multiple, thin-shell obstacles is discussed in [34]. **Relaxation:** In [9],

[6], relaxation schemes are used to limit the solutions to a (piecewise) convex and smooth subset. For example, the freespace is approximated by the union of convex subsets and a UAV-trajectory is represented by piecewise splines, where each piece is constrained in one convex subset. These relaxed formulations can be efficiently handled by modern (mixed-integer) convex optimization tools. However, these methods can only find sub-optimal solutions because the convex subset is a strict inner-approximation with a non-vanishing gap (and increasing the approximation power increases the algorithmic running time). For environments with narrow passages, relaxation can even turn a feasible problem into an infeasible one. **Global Search:** Kinodynamic-RRT* [41] and its variants can find globally optimal UAV trajectories in a discrete space and stochastic gradient descend [17] will approach locally optimal solutions only if the descendent direction is approximately by infinitely many samples. However, these algorithms take excessively large number of samples. To mitigate their computational cost, prior methods such as [23] have to terminate sampling early and then numerically rectify the solutions, and these numerical rectification methods still suffer from the limitations of infeasible recover or relaxation approaches.

Main Results: We propose a new formulation of local, optimization-based motion planning for UAV-trajectories with robustness and asymptotic optimality guarantee. Our method represents the UAV-trajectory using smooth curves, encoded as piecewise Bézier curves. For each Bézier piece, we formulate the collision-free constraints between its control polygon and the environmental geometry. We show that these constraints can be formulated as a summation of distance functions between geometric primitives (e.g. point-triangle and edge-edge pairs). Each distance function is differentiable when restricted to the feasible domain and the restriction can be achieved using a primal interior point method (P-IPM). By using fewer geometric primitive pairs for the collision avoidance, we further propose an *inexact* P-IPM algorithm that achieves a speedup up to 15×, comparing to P-IPM. The convergence guarantees, under infinite precision, are provided for both the P-IPM and *inexact* P-IPM methods.

As compared with infeasible recovery methods that maintain both primal and dual variables, our primal-only formulation provides guaranteed solution feasibility throughout the entire process of optimization. This feature allows our planner to answer anytime re-planning queries as in [21] because the planner can be terminated at any iteration and return a feasible solution. And compared with relaxation-based methods, our method can asymptotically minimize the sub-optimality gap due to the control-polygon approximation of the true trajectory.

[1]Ruiqi Ni and Xifeng Gao are with Department of Computer Science, Florida State University (rn19g@my.fsu.edu and gao@cs.fsu.edu). [2]Teseo Schneider is with the Department of Computer Science, University of Victoria (teseo@uvic.ca). [3]Daniele Panozzo is with the Department of Computer Science, New York University (panozzo@nyu.edu). [4]Zherong Pan is with the Department of Computer Science, University of Illinois Urbana-Champaign (zherong@illinois.edu).
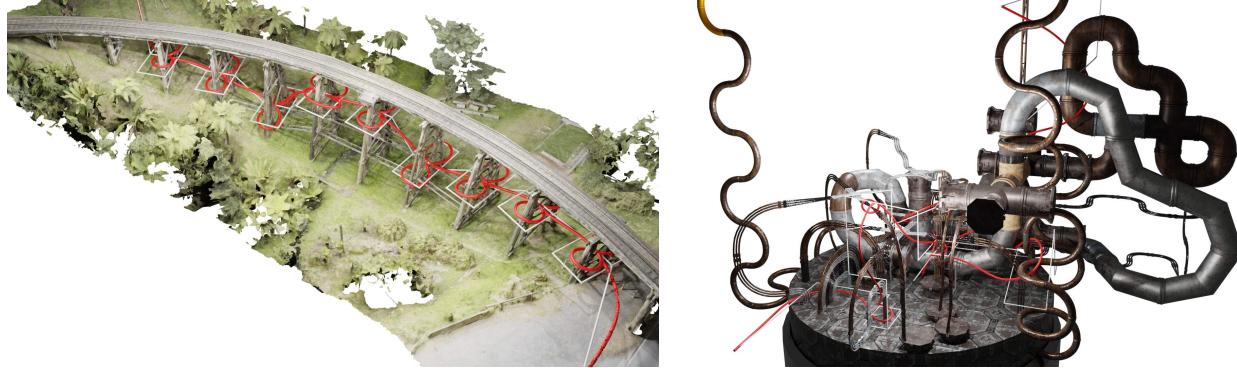
Fig. 1: Optimized (red) trajectories by our method for a bridge (left) and a machinery used in factories (right) which are challenging environments containing complex geometrical and topological features. Initial trajectories are in white color. During the optimization, our approach maintains the safe distance clearance, the homotopy class of the trajectory, and the dynamic actuation limits, and takes 1.2hrs and 0.25hrs to finish the solve respectively.

This is achieved by using an adaptive Bézier subdivision scheme with user-controllable termination criterion.

Our methods contribute to the solving of UAV planning problem in three ways: 1) Combined with RRT-style approaches, we can robustly optimize a feasible initial guess. 2) Starting from a feasible initial guess, our primal solver is guaranteed to preserve the homotopy class of the trajectory. This feature is useful, for example, for inspection planners where the order of inspection cannot be altered and Figure 1 demonstrates that our approach can be potentially used for complex environment inspections. 3) Our formulation achieves simultaneous optimality in trajectory shapes and time allocations.

This paper is an extension of our previous work [28] with 1) more technical details of the inexact P-IPM approach, 2) a complete convergence analysis showing that, within a finitely many iterations, our inexact algorithm convergences to a solution with the same quality as P-IPM, and 3) additional experiments that compare the performances of both approaches, and demonstrate the effectiveness of our approach on challenging settings. To foster researches along this direction, we release the source code of our implementation at https://github.com/ruiqini/traj-opt-subdivision, and also make public of the dataset used for our experiments on Google Drive.

II. RELATED WORK

Trajectory optimization for a general robotic system can be solved using dynamic programming [35], direct collocation [39], and path-integral control [17], [43]. While widely applicable, these methods require a search in a high-dimensional space and are thus computationally demanding and sensible to numerical failures. For the special case of UAV dynamics, a more robust and efficient approach has been proposed in [24], where the algorithm optimizes a reference position trajectory (global search) and then recovers the control inputs making use of differential flatness (local optimization). Our method follows the same strategy but offers the additional advantage of feasibility guarantees and locally optimality of the original non-convex optimization problem.

Global Search provides an initial trajectory that routes the UAV from a start to a goal position and usually optimizes the trajectory by minimizing a state- or time-dependent objective function. Global search can be accomplished using sampling-based motion planner [42], [20], mixed-integer programming [6], discrete search [22], and fast-marching method [12]. All these methods are approximating the globally optimal UAV-trajectory under some assumptions: optimal sampling-based motion planner approaches the optimal solution after a sufficient number of samples have been drawn; mixed-integer programming and discrete search restricts the decision space to a disjoint-convex or discrete subset, respectively; fast-marching methods discretize both the trajectory and the environment on a uniform grid. Global search is essential not only for providing an initial guess to the local optimization, but also for satisfying additional constraints. Typical constraints include visitation order [14], homotopy class [18], and coverage [15], [8]. These constraints can be formulated into the mixed-integer programming via additional binary decision variables [6] or into discrete search algorithms such as A* by pruning trajectories that violate the constraints [22]. All these methods are orthogonal to our work and any global search method with the collision-free guarantee can be used as the initialization step of our method.

Local Optimization complements the global search by refining the trajectories in a neighborhood of the initial guess. This step lifts the restricted search space assumption in the global search and further minimizes the objective function in the continuous decision space, until a local optimal solution is attained. In addition, the local step ensures that the solution satisfies kinematics and dynamics constraints so that the trajectory is executable on hardware. All prior works formulate the local step as a continuous or discrete optimization problem. The continuous problem can only be solved in the obstacle-free cases with a closed-form solution [41], [22]. In [12], [37], [40], obstacle-free constraints are relaxed to be convex and the resulting discrete optimization is guaranteed to be solvable using primal-dual interior point method (PD-IPM). In [44], [11], obstacle-free constraints are considered in its original, non-convex form, which is solved by PD-IPM but without

feasibility guarantee. In this work, we show that the feasibility guarantee can be provided using P-IPM.

Time-Allocation is an essential part of local optimization formulated as part of the dynamic constraints. Early works [32], [12] first optimize the shape of the trajectory and then allocate time for each discrete segment, and the resulting trajectory is sub-optimal with respect to the time variables. It has been shown that, for a fixed trajectory, optimal time-allocation can be achieved using bang-bang solutions computed via numerical integration [30]. The trajectories computed using numerical integration by [30] are locally optimal in either shape variables or time variables but not both at the same time. The latest works [36], [37] achieve shape-time joint optimality via bilevel optimization or weighted combination. We following [37] and optimize a weighted combination of shape and time cost functions.

III. PROBLEM STATEMENT

We propose a new method for local optimization of UAV trajectories. A UAV moves along the center-of-mass trajectory $p(t, W)$ with $t \in [0, T]$ where T is the travel time and W is the set of decision variables. A local optimizer takes an initial trajectory as input and locally updates it to minimize a cost function:

$$\mathcal{O}(W, T) = \int_0^T c(p(t, W), t) dt + R(p(t, W), T), \quad (1)$$

where $c(p, t)$ is the state-dependent cost (e.g., dependent on velocity, acceleration, jerk, or snap) and $R(p, T)$ is the terminal cost. In the meantime, a feasible UAV trajectory should satisfy a set of kinematic, dynamics, and application-dependent constraints. In this paper, we assume that dynamics constraints take the form of velocity and acceleration limits:

$$\|\dot{p}(t, W)\| \le v_{\max} \quad \|\ddot{p}(t, W)\| \le a_{\max} \quad \forall t \in [0, T], \quad (2)$$

which are semi-infinite constraints in the variable t. Here $\{v, a\}_{\max}$ are the maximum allowable velocity and acceleration. We further assume only the collision-free kinematic constraint that can take different forms depending on how obstacles are represented. Prominent environmental representations are point clouds and triangle meshes. These two representations can be uniformly denoted as a tuple of discrete elements $\mathcal{E} = \langle \mathcal{P}, \mathcal{L}, \mathcal{T} \rangle$ where $\mathcal{P}$ is a set of points, $\mathcal{L}$ is a set of line segments, and $\mathcal{T}$ is a set of triangles. The distance between a point x and the environment is denoted as $\text{dist}(x, \mathcal{E})$. Throughout the paper, we slightly abuse notation and define $\text{dist}(\bullet, \bullet)$ as closest distance between a pair of geometric objects (e.g. points, line segments, triangles, convex hulls, point clouds, and triangle meshes). Our complete local trajectory optimization problem can be written as:

$$\underset{W, T}{\text{argmin}} \, \mathcal{O}(W, T) \quad \text{s.t.} \, \forall t \in [0, T] \begin{cases} \|\dot{p}(t, W)\| \le v_{\max} \\ \|\ddot{p}(t, W)\| \le a_{\max} \\ \text{dist}(p(t, W), \mathcal{E}) \ge d_0 \end{cases}, \quad (3)$$

where d_0 is an uncertainty-tolerating safety distance. Equation 3 is among the most challenging problem instances in operations research due to the semi-infinite constraints in time t and the non-smoothness of the function $\text{dist}(\bullet, \bullet)$ in general [12], [16].

A. Trajectory Representation

We represent $p(t, W)$ as a composite Bézier curve where each Bézier piece has a constant degree M. We note that this generic definition allow for curves of arbitrary continuity. The continuity across different Bézier pieces is achieved by constraining the control points. The constraints are linear and can thus be removed from the system based on Stärk's construction [31].

IV. SUBDIVISION-BASED P-IPM

We propose an iterative algorithm, inspired by the recently proposed incremental potential contact handling scheme [19], to find locally optimal solutions of Equation 3 with guaranteed feasibility.
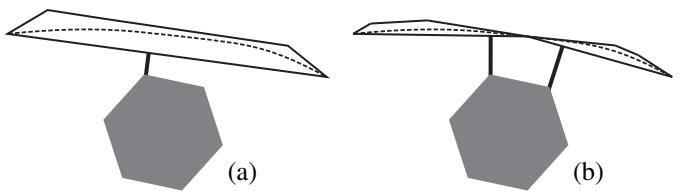

Fig. 2: (a): Our method imposes distance constraints between the convex hull of control points and the obstacle. (b): Our constraints can be refined by subdivision.

A. Method Overview

The key idea of [19] is to use a primal-only (instead of primal-dual) interior point method to solve collision-constrained nonlinear optimization problems. This is achieved by transforming all constraints including collision avoidance constraints into log-barrier functions, and reducing the problem to non-linear, unconstrained optimization. Although primal-dual approaches have better numerical stability in many problems, their convergence relies on sufficient smoothness and Mangasarian-Fromovitz constraint qualification (MFCQ) along the central path, which do not hold for collision constraints, often leading to failures in feasibility recovery. In contrast, given a feasible initial guess, line-search based, primal-only solvers are guaranteed (even in floating point implementations), to stay inside the feasible domain. The key to establish this invariance is the use of a robust line-search scheme that prevents solution from leaving the feasible domain, in our case the major challenge is rejecting iterations with a jump in homotopy class or violation of collision-free constraints. For other constraints, namely the acceleration and velocity limits, we show that they can be handled in a similar manner (Section IV-E). It has been shown in [19] that, for linearized trajectories, these safety checks can be performed using linear continuous collision detection (CCD). In this work, we extend this construction for motion planning with curved trajectories.

A safe line-search scheme ensures feasibility but does not guarantee local optimality, i.e. convergence to a first-order critical point. Assuming gradient-based optimizers are used, the additional requirement for local optimality is the objective function Equation 3 being differentiable, which is challenging

due to the non-smoothness of the function dist($\bullet, \bullet$). Although SQP algorithms with gradient-sampling [5] can handle piecewise-smooth constraints such as dist($p(t, W), \mathcal{E}) \geq d_0$, they degrade the deterministic convergence guarantee to a probabilistic one in the sampling limit. In [19], [16], the authors note that two objects, which could be represented by triangle meshes, $\mathcal{E}$ and $\mathcal{E}'$ are d_0-apart if and only if any line segment pair in $\mathcal{LL}' \triangleq \mathcal{L} \times \mathcal{L}'$ and point-triangle pair in $\mathcal{PT}' \triangleq \mathcal{P} \times \mathcal{T}'$ or $\mathcal{P}'\mathcal{T} \triangleq \mathcal{P}' \times \mathcal{T}$ are d_0-apart. As a result, the log-barrier of dist($\bullet, \bullet$) is equivalent to the sum of log-barrier of distance functions between primitive pairs, which can be made differentiable after an arbitrarily small perturbation.

We plan to use primal-only algorithms to solve UAV trajectory's local optimization problems. To this end, the three preconditions of the primal-only method must hold. The first condition of a feasible initial guess holds trivially by using an appropriate global searcher. With a fine-enough discretization granularity of signed distance field in [12], or that of sampled waypoints in [22], a feasible initial guess can be computed as long as a solution exists. For the second condition, although robust, exact continuous collision predicates exist for linear triangular meshes [3], [38], they have not been developed for polynomial curves. To bridge the gap, we propose to relax the exactness and resort to a conservative CCD scheme based on the subdivision of Bézier curve. For the third condition, i.e. the differentiability of the objective function, we propose to replace the trajectory with the union of convex hulls of the control polygons. These convex hulls are triangle meshes so the differentiable log-barrier between geometric primitives can be used. The accuracy of this approximation can be made arbitrarily high via adaptive subdivision.

B. P-IPM Framework

We use a column vector w to represent the set of control points of a Bézier curve. For the i-th piece Bézier curve of the trajectory, we denote its control points as A_iW, where A_i is a fixed transformation matrix of decision variables W into i-th curve's control points. A Bézier piece can be subdivided by a linear transformation of its control points into two pieces [10]: we denote control points of the first piece as D_1w and the control points of the second piece as D_2w, where $D_{1,2}$ are the fixed subdivision stencils constructed using the De-Casteljau's algorithm. Our adaptive subdivision scheme (Algorithm 3) will recursively apply these stencils until a stopping criterion is met. And we keep a subdivision history $\mathcal{H}$ to maintain the subdivision transformation matrices for each subdivided Bézier curve piece, where $\mathcal{H}$ will be initialized with $\{A_i | i = 1, ..., N\}$. Our algorithm handles two meshes $\mathcal{E}$ and $\mathcal{E}'$. We use $\mathcal{E}$ to denote the mesh of the environment and $\mathcal{E}'$ denotes the mesh discretizing the UAV trajectory. After the subdivision, $\mathcal{E}'$ will be updated so that it is the union of convex hulls of the new set of control points. For each subdivided Bézier curve piece with control points w, we denote $P(w)$ as the set of $M + 1$ vertices of the convex hull, $L(w)$ as the set of $(M + 1)M/2$ edges connecting all pair of vertices of $P(w)$, and $T(w)$ as the set of $(M + 1)M(M - 1)/6$ triangles connecting any 3 vertices. As illustrated in Figure 2, given the environment $\mathcal{E}$ (represented using either a point cloud or a triangle mesh) and $\mathcal{E}'$, we can define our log-barrier function as:

$$B(W, \mathcal{E}) \triangleq \sum_{A \in \mathcal{H}} \left[\sum_{l \in \mathcal{L}, l' \in L(w)} \text{clog}(\text{dist}(l, l') - d_0) + \quad (4) \right.$$

$$\sum_{p \in \mathcal{P}, t' \in T(w)} \text{clog}(\text{dist}(p, t') - d_0) +$$

$$\left. \sum_{t \in \mathcal{T}, p' \in P(w)} \text{clog}(\text{dist}(t, p') - d_0) \right] \Bigg|_{w = AW},$$

where dist is the mollified, differentiable distance between a segment-segment or line-triangle pair, and clog is the clamped log-barrier function:

$$\text{clog}(x) = \begin{cases} -\frac{(x - x_0)^2}{x} \log\left(\frac{x}{x_0}\right) & 0 < x \leq x_0 \\ 0 & \text{elsewhere}, \end{cases}$$

with x_0 being the activation range. Note that we use a slightly modified version of clog from the original paper [19]. Our modification preserves the second-order continuity but is essential for the correctness of our inexact algorithm introduced in Section IV-D. Although there are many terms in Equation 4, all the terms with dist($\bullet, \bullet$) larger than $d_0 + x_0$ are zero. Therefore, we use a bounding volume hierarchy structure to efficiently prune all the zero terms.

Algorithm 1 NeedSub(w)

1: Return dist(Hull(w), $\mathcal{E}$) $< d_0 + x_0 \wedge$ diameter(Hull(w)) $> \epsilon$

Algorithm 2 Sub($w, A, \mathcal{H}$)

1: $< \mathcal{P}', \mathcal{L}', \mathcal{T}' > \leftarrow < \varnothing, \varnothing, \varnothing >$
2: **if** NeedSub(w) **then**
3:     $\mathcal{H} \leftarrow \mathcal{H} \backslash \{A\} \cup \{D_1 A\} \cup \{D_2 A\}$
4:     $< \mathcal{P}', \mathcal{L}', \mathcal{T}' > \leftarrow < \mathcal{P}', \mathcal{L}', \mathcal{T}' > \cup \text{Sub}(D_1 w, D_1 A, \mathcal{H})$
5:     $< \mathcal{P}', \mathcal{L}', \mathcal{T}' > \leftarrow < \mathcal{P}', \mathcal{L}', \mathcal{T}' > \cup \text{Sub}(D_2 w, D_2 A, \mathcal{H})$
6: **else**
7:     $< \mathcal{P}', \mathcal{L}', \mathcal{T}' > \leftarrow < P(w), L(w), T(w) >$
8: Return $< \mathcal{P}', \mathcal{L}', \mathcal{T}' >$

Algorithm 3 TrajSub($p(t, W), \mathcal{H}$)

1: $< \mathcal{P}', \mathcal{L}', \mathcal{T}' > \leftarrow < \varnothing, \varnothing, \varnothing >$
2: **for** $A \in \mathcal{H}$ **do**
3:     $< \mathcal{P}', \mathcal{L}', \mathcal{T}' > \leftarrow < \mathcal{P}', \mathcal{L}', \mathcal{T}' > \cup \text{Sub}(AW, A, \mathcal{H})$
4: Return $\mathcal{E}' \leftarrow < \mathcal{P}', \mathcal{L}', \mathcal{T}' >$

Algorithm 4 P-IPM

Input: W_0, T_0
1: $\mathcal{H} \leftarrow \{A_i | i = 1, ..., N\}, W \leftarrow W_0, T \leftarrow T_0$
2: **while** True **do**
3:     $\mathcal{E}' \leftarrow \text{TrajSub}(p(t, W), \mathcal{H})$
4:     Calculate $\nabla_{W,T} O, \nabla^2_{W,T} O$
5:     Calculate $d \leftarrow -\text{SPD}(\nabla^2_{W,T} O)^{-1} \nabla_{W,T} O$
6:     $< W, T > \leftarrow \text{Search}(< W, T >, d)$
7:     **if** $\|\nabla_{W,T} O\|_\infty \leq \epsilon_g$ **then**
8:         Return $< W, T >$

We are now ready to summarize our main algorithm of P-IPM in Algorithm 4, which is a Newton-type method applied to the following unconstrained optimization problem:

$$\underset{W}{\text{argmin}} \, O(W, T) + \lambda B(W, \mathcal{E}), \quad (5)$$

where λ is the relaxation coefficient controlling the exactness of constraint satisfaction. If the cost function is differentiable,

the Hessian has bounded eigenvalues, then P-IPM converges to a first-order critical point of Equation 5. Finally, the function SPD($\bullet$) adjusts the Hessian matrix by clamping the negative eigenvalues to a small positive constant through a SVD to ensure positive-definiteness.

C. Convergence Guarantee

The convergence guarantee of Algorithm 4 relies on the correct implementation of two functions: Search (Algorithm 5) and NeedSub (Algorithm 1). The search function updates W to W' and ensures that the Wolfe's first condition:

$$O(W', T) + \lambda B(W', \mathcal{E}) \leq O(W, T) + \lambda B(W, \mathcal{E}) +$$
$$c\alpha \nabla_W \left[O(W, T) + \lambda B(W, \mathcal{E}) \right]^T d,$$

holds and $p(t, W)$ is homotopically equivalent to $p(t, W')$, where $c \in (0, 1)$ is some positive constant. Instead of using CCD which requires numerically stable polynomial root finding, we use a more conservative check between the convex hull of geometric primitives before and after the update from W to W'. This procedure is outlined in Algorithm 5.

The NeedSub function guarantees that the solution of Equation 5 asymptotically converge to that of Equation 3. For P-IPM, whenever there is an active log-barrier term in Equation 4, the convex hull of some Bézier curve with control points w is at most $d_0 + x_0$ away from $\mathcal{E}$. This implies we can bound the maximal distance between any curve's point and $\mathcal{E}$ as $d_0 + x_0 + \Delta(w)$, where $\Delta(w)$ is the diameter of the convex hull of the control polygon or any upper bound between a curve's point and its convex hull. A simple strategy that guarantees ϵ-optimality is to always subdivide when 1) $\Delta(w) > \epsilon$ and 2) the distance between the convex hull of w and $\mathcal{E}$ is smaller than $d_0 + x_0$, where ϵ is a user-provided optimality threshold. Note that condition 2) inherently induces an adaptive subdivision scheme where all the sub-trajectories that are sufficiently faraway from $\mathcal{E}$ are not subdivided to save computation. In theory, under assumption Assumption 4.1 and $\epsilon_g = 0$, we show in Section Section VII that P-IPM will converge to the local optima of the original semi-infinite oracle (Equation 3) as the coefficients of log-barrier functions (λ) and the clamp range of clog functions (x_0) tends to zero. In practice, we show that P-IPM will terminate after a finite number of iterations under the following assumption:

Assumption 4.1: The objective function $\mathcal{O}$ is twice-differentiable and the sequence of w generated by iterations of P-IPM is uniformly bounded.

Proposition 4.2: Suppose Assumption 4.1 holds and $\epsilon_g > 0$, then gradient descend method using line search Algorithm 5 terminates within finitely many iterations.

Proof: See Section VIII.   ∎

D. Inexact P-IPM

A practical problem with Algorithm 4 is that we have to add all the $M + 1$ vertices $(M + 1)M/2$ edges in $L(w)$ and all the $(M + 1)M(M - 1)/6$ triangles in $T(w)$ of each convex hull of the control polygon to Equation 4. Even with a bounding volume hierarchy, summing up so many terms is still time-consuming. To overcome this issue, we propose an inexact,

---

Algorithm 5 Search($< W, T >, d$)

Input: $\gamma, c_1 \in (0, 1)$
1: $\alpha \leftarrow 1, < W', T' > \leftarrow < W, T > + \alpha d$
2: **while** True **do**
3:      Safe←True
4:      **for** $A \in \mathcal{H}$ **do**
5:          **if** dist(Hull($AW \cup AW'$), $\mathcal{E}$) $< d_0$ **then**
6:              Safe←False
7:      **if** Safe $\wedge$ Wolfe's first condition **then**
8:          Return $< W', T' >$
9:      **else**
10:         $\alpha \leftarrow \gamma \alpha, < W', T' > \leftarrow < W, T > + \alpha d$

---

yet preserving the guarantees of local optimality, version of P-IPM. For every Bézier piece, we only keep all $M + 1$ points in $P(w)$ and one edge that connects the starting and ending control points of the curve's control polygon. As compared with the exact counterpart, the inexact version significantly reduces the cost of log-barrier function evaluation by a factor of $O(M^2)$ since we drop the entire triangle set $T(w)$.

Note that most edges and all triangles are omitted in constructing Equation 4 but not in the Search function to ensure feasibility. This naive inexact P-IPM is not guaranteed to converge to the local minimum of Equation 5 because Algorithm 5 might not find a positive α satisfying the first Wolfe's condition due to the conservative collision check. To ensure convergence to a local minimum, a simple strategy is to keep subdividing whenever the line-search fails. By slight modifications to the line search Algorithm 5, in the following, we present an inexact P-IPM algorithm that is guaranteed to converged to the KKT point after a finite number of subdivisions.

For simplicity of presentation, we assume there is only one Bézier curve piece, $A(s)w$ with $s \in [0, 1]$, i.e. $N = 1$. After inexact P-IPM performs a series of subdivisions, the $[0, 1]$ segment will be divided into consecutive intervals:

$$0 \leq s_1 < s_2 < \cdots < s_{S-1} < s_S = 1.$$

Using this notation, we can rewrite the inexact log-barrier function as:

$$\tilde{B}(w, \mathcal{E}) \triangleq \sum_{i=1}^{S-1} (s_{i+1} - s_i)\text{clog}(\text{dist}(m'_{i,i+1}(w), \mathcal{E}) - d_0),$$

where we use $m'_{i,i+1}(w)$ to denote the line segment connecting the start and the end control points of curve piece with $s \in [s_i, s_{i+1}]$. Note that inexact log-barrier function is scaled by the range of each sub-curve $s_{i+1} - s_i$, which is unnecessary in its exact counterpart.

Two modifications to Algorithm 5 are needed to ensure finite termination. First, we need to use a more conservative safeguard of the following form:

$$\text{Safe} \leftarrow \text{dist}(\text{Hull}(\mathcal{P}'(w), \mathcal{P}'(w')), \mathcal{E}) > d_0 + \psi(s_{i+1} - s_i),$$

where $\psi(s)$ is some positive function for which we choose $\psi(s) = \epsilon_s s^\eta$, where ϵ_s, η are small positive constants. Second, we subdivide all the curve pieces that do not satisfy the safeguard condition if α is smaller than some ϵ_α. The modified

---

Algorithm 6 Subdivide-Unsafe(w, w')

1: **for** $i = 1, \cdots, S-1$ **do**
2:     ▷ Consider subdivided Bézier curve piece s_i, s_{i+1}
3:     **if** dist(Hull($\mathcal{P}'(w), \mathcal{P}'(w')), \mathcal{E}) < d_0 + \psi(s_{i+1} - s_i)$ **then**
4:         Insert $\frac{s_i + s_{i+1}}{2}$ between s_i and s_{i+1}

Algorithm 7 Inexact-Search($<w, T>, d, \epsilon_\alpha$)

Input: $\gamma, c_1 \in (0,1), \epsilon_s > 0, \eta \in (0, 1/3)$
1: $\alpha \leftarrow 1, <w', T'> \leftarrow <w, T> + \alpha d$
2: **while** True **do**
3:     Safe←True
4:     **for** $i = 1, \cdots, S-1$ **do**
5:         ▷ Consider subdivided Bézier curve piece s_i, s_{i+1}
6:         **if** dist(Hull($\mathcal{P}'(w), \mathcal{P}'(w')), \mathcal{E}) < d_0 + \psi(s_{i+1} - s_i)$ **then**
7:             Safe←False
8:     **if** Safe ∧ Wolfe's first condition **then**
9:         Return $<w', T'>, \epsilon_\alpha$
10:    **else if** Safe=False $\wedge \alpha < \epsilon_\alpha$ **then**
11:        Subdivide-Unsafe(w, w')
12:        $\epsilon_\alpha \leftarrow \gamma \epsilon_\alpha$
13:    **else**
14:        $\alpha \leftarrow \gamma \alpha, <w', T'> \leftarrow <w, T> + \alpha d$

---

line-search scheme is outlined in Algorithm 7. We also propose to save computation by reducing ϵ_α after each subdivision.

In summary, our inexact algorithm solves:

$$\underset{w}{\arg\min}\ O(w, T) + \lambda \tilde{B}(w, \mathcal{E}), \qquad (6)$$

using Newton's method Algorithm 8 with Algorithm 7 being the line-search scheme. Our algorithm is guaranteed to terminate within a finite number of subdivisions as summarized below:

Proposition 4.3: Assuming Assumption 4.1 and $\epsilon_g > 0$, then Algorithm 8 using line search Algorithm 7 terminates within finitely many iterations.

    Proof: See Section IX. ∎
The crucial observation behind the proof is that, after infinite uniform subdivisions, the barrier function $\tilde{B}$ would approach the following integral of clog in the sense of Riemann sum:

$$\hat{B}(w, \mathcal{E}) \triangleq \int_0^1 \text{clog}(\text{dist}(A(s)w, \mathcal{E}) - d_0) ds,$$

and the finite termination can be proved by bounding the error between $\tilde{B}$ and $\hat{B}$.

E. Time-Optimality

Time efficacy can be formulated as a terminal cost such as $R(p, T) = T$, combining the semi-infinite velocity and acceleration bounds. Unlike [36], [12] that warp each Bézier curve piece using a separate time parameter, we use a single, global time re-scaling. In other words, the UAV flies through each polynomial piece for T/N seconds. Although we use a single variable T as compared with N variables for each piece in [36], [12], we argue that time-optimality will not be sacrificed because our formulation allows a larger solution space for the relative length of the N Bézier curves. In prior work [12] for example, each Bézier curve is constrained to a separate convex subset of the freespace, which restricts the relative length of neighboring Bézier curves. By comparison, our method does not rely on these constraints and allow

---

Algorithm 8 Inexact-P-IPM

Input: $w_0, T_0, \epsilon_\alpha > 0$
1: $\mathcal{H} \leftarrow \varnothing, w \leftarrow w_0, T \leftarrow T_0$
2: **while** True **do**
3:     $\mathcal{E}' \leftarrow \text{TrajSub}(p(t, w), \mathcal{H})$
4:     Calculate $\nabla_{w,T} O, \nabla^2_{w,T} O$
5:     Calculate $d \leftarrow -\text{SPD}(\nabla^2_{w,T} O)^{-1} \nabla_{w,T} O$
6:     $<w, T>, \epsilon_\alpha \leftarrow \text{Search}(<w, T>, d, \epsilon_\alpha)$
7:     **if** $\|\nabla_{w,T} O\|_\infty \le \epsilon_g$ **then**
8:         Return $<w, T>$

Fig. 3: We optimize a curve involving a sharp turn in the middle using two different time-allocations. Left: Each curve is assigned the same amount of time. Right: The middle curve is assigned less time. For both cases, our optimizer generates similar trajectories (yellow dots are the end points of Bézier curves).

the relative length to change arbitrarily. As illustrated in Figure 3, we optimize two trajectories using different, fixed time allocations, and our method converges to almost identical solutions by changing the relative length of curves, which justifies the redundancy of curve-wise time variables.

We use the same principle as collision constraints to handle velocity and acceleration limits, by introducing a new set of log-barrier functions. For a subdivided curve piece with control points w, its first and second derivatives are two new Bézier curves, where their control points are $S_1 w$ and $S_2 w$, and $S_{1,2}$ are the fixed transformation matrix constructed by Bézier derivative. The convex hull of their control polygons and corresponding point, edge, triangle set are denoted as $\langle P(S_1 w), L(S_1 w), T(S_1 w) \rangle$ for velocity and $\langle P(S_2 w), L(S_2 w), T(S_2 w) \rangle$ for acceleration. We can now define our log-barrier function approximating the velocity and acceleration limits as:

$$B_T(W, T) = \sum_{i=1}^{N} \Bigg[\sum_{p_1 \in P(S_1 w)} \text{clog}(v_{max} - \|p_1\|_2/(T/N)) + \qquad (7)$$
$$\sum_{p_2 \in P(S_2 w)} \text{clog}(a_{max} - \|p_2\|_2/(T^2/N^2)) \Bigg]\Bigg|_{w = A_i W}.$$

P-IPM now minimizes $O(W, T) + \lambda B(W, \mathcal{E}) + \lambda B_T(W, T)$ to achieve joint optimality in terms of trajectory shape and time. Due to the convex hull property of Bézier curves, the finite value of $B_T(W, T)$ implies that Bézier curve with control points $S_1 w$ is bounded by $T v_{max}/N$ and Bézier curve with control points $S_2 w$ is bounded by $T^2 a_{max}/N^2$ and subdivision can make the approximation arbitrarily exact. If desired, a separate subdivision rule can be used to control the exactness, e.g. always subdivide when $\Delta(S_1 w) > \dot{\epsilon}$ or $\Delta(S_2 w) > \ddot{\epsilon}$. Finally, note that we need a feasible initial guess for T and a sufficiently large T is always feasible.

V. EXPERIMENTS

Our implementation uses C++11 and all results are computed using a single thread on a workstation with a 3.5 GHz Intel Core i9 processor. Our algorithm has the following parameters: λ of the collision avoidance barrier term, x_0 of the activation range, d_0 of the clearance distance, v_{max}, a_{max}, ϵ of the subdivision threshold and ϵ_g of the stopping criterion. We use $\lambda = 10$, $x_0 = 0.1$, $d_0 = 0.1$, $v_{max} = 2.0 m/s$,

Scene	Size	$L^{[12]}$	$T^{[12]}$	$C^{[12]}$	$L^{[11]}$	$T^{[11]}$	$C^{[11]}$	L^{P-IPM}	T^{P-IPM}	C^{P-IPM}	L^{IP-IPM}	T^{IP-IPM}	C^{IP-IPM}
1	1.4M/2.7M	70.5	144.3	168.5	82.6	73.3	0.5	56.3	30.5	3.4K	56.2	30.4	218.9
2	31K/58K	25.7	29.1	8.3	21.5	26.9	0.2	16.1	9.6	13.0	16.1	9.6	2.9
3	36K/68K	18.7	24.1	2.5	20.1	19.0	0.6	16.7	9.8	5.4	16.7	9.8	2.3
4	17K/34K	25.1	24.1	4.0	22.8	23.6	0.1	19.8	11.3	15.0	19.8	11.3	9.0
5	0.4M/0	49.8	65.0	6.4	50.0	50.0	0.3	39.0	21.7	400.4	38.2	21.3	42.1
6	0.2M/0.3M	-	-	-	14.8	18.0	0.2	14.3	9.7	176.0	13.8	9.2	14.9
7	81K/16K	23.2	38.5	3.8	30.9	26.3	0.2	21.6	12.4	12.6	21.6	12.4	6.2
8	36K/68K	24.5	23.5	3.3	28.1	26.1	0.1	18.5	10.7	13.0	18.5	10.7	2.2

TABLE I: We profile the performance of [12], [11], our P-IPM method, and our Inexact P-IPM method in terms of trajectory length L, arrival time T, and computational cost C. The size of each scene, in terms of its numbers of vertices and faces, is listed in the form of #V/#F. – denotes a failure in the provided program. Red color is used to highlight trajectories where the UAV collides with the scene.
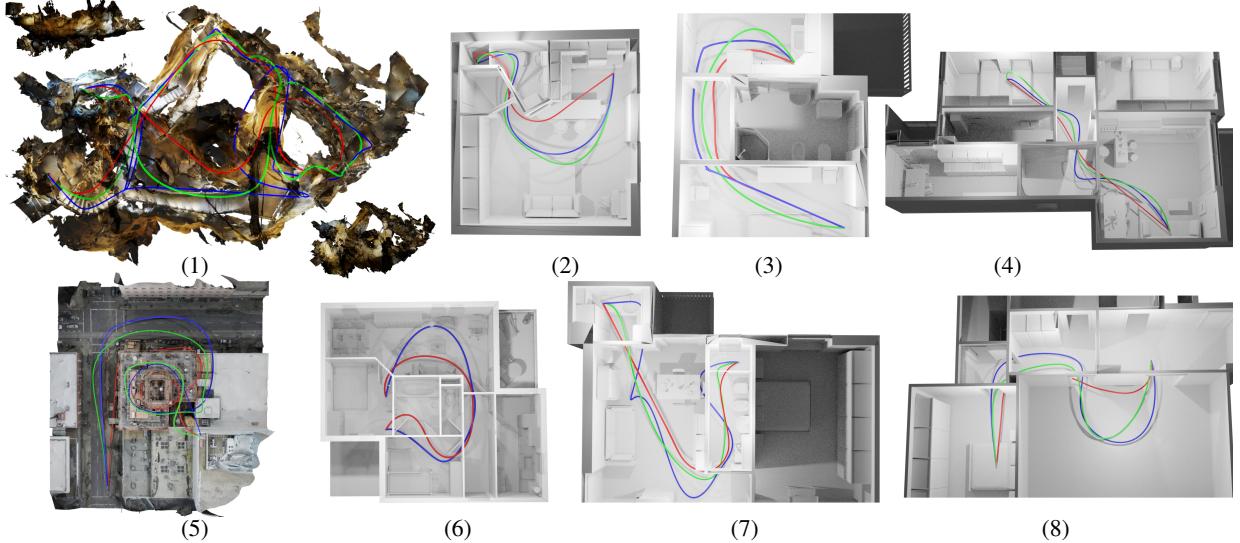

Fig. 4: For the scenes listed in Table I, we compare UAV trajectories computed using our method (red), [11] (blue), and [12] (green) from the same initial guess. Both our method and [12] can generate feasible trajectories and our method finds a smaller objective function (e.g. smoother trajectory and smaller travel time as shown in Table I), while [11] penetrates most of the environments.

$a_{\max} = 2.0 m/s^2$, $\epsilon = 0.1$, $\epsilon_g = 10^{-3}$ for all experiments. To match the energy setting of the works we compare with, we use the composite Bézier where each piece is degree 8 and the continuity between adjacent pieces is C^2, and set c in the objective function as jerk energy. We use improved GJK method [26] for convex hull collision detection.

A. Comparisons

We compare the two versions of our approach with the state-of-the-art gradient based method [11] and corridor based method [12] on a set of scenes, represented by either point clouds or triangle meshes, that are publicly downloadable from SketchFab [7]. We use the public implementations of both methods and tune our implementation to match the parameter settings of their methods, such as the orders of curves representing the trajectory, the energy term to measuring the smoothness of the trajectory (i.e. snap, jerk), $v_{\max}$, and $a_{\max}$. The implementation of [12] and [11] only handles point cloud data. Therefore, we use densely sampled point clouds from the triangular meshes representing our scenes. Because of the complexity of the scenes listed in Table I, we manually tune waypoints to ensure that there will be a valid initial trajectory generated by the global search for all the methods, i.e. [12], [11], and ours.

We shown several benchmarks in Table I, where most environments are represented using triangle meshes except for the 5th scene, which is a dense point cloud. While both the P-IPM and Inexact P-IPM methods produce trajectories with approximately the same lengths and travel times, the Inexact P-IPM method computes much faster than the P-IPM approach, with speedup ranging from 1.6× to 15.5×. If not specifically mentioned, we perform the comparisons and simulations using only the Inexact P-IPM method since it provides similarly the robustness and convergence guarantees.

As demonstrated in Table I, our trajectories have the highest quality and far outperform the results of [12], [11]. Visual results are shown in Figure 4. Note that, since [11] doesn't have a collision-free guarantee even with a valid initial trajectory, their generated trajectories often penetrate the environments as highlighted in red in Table I. On the contrary, our method not only ensures a collision-free trajectory at every step during the optimization, but also guarantees that the trajectory is never closer to the obstacles than a user-specified safe distance. Although [12] requires only waypoints as input, we have to modify, in a trial-and-fail way, either the waypoints or the grid resolution used by their approach so that their method can find a collision-free initialization to ensure that their trajectory optimization proceeds.

Fig. 5: Left: a simulation screenshot of the UAV (red) flies along a simulated trajectory that has neglectable differences from our optimized trajectory (green) in a museum. Right: each plot shows the overlaid positions (x/y/z[m]) and the speeds (xdot/ydot/zdot[m/s]) of UAV for both the simulated and our trajectories.

Note that, while our method obtained competitive computing efficiency to [12] for some scenes, [11], [12] typically achieve real-time computations and good quality trajectories for simple environments. However they struggle as the environments gets more complex. [11] tends to generate invalid trajectories with collisions between the UAV and the environment. [12] instead has a quickly growing runtime, since they have to reduce the failure rate of their initial trajectory generation by increasing the grid resolution, resulting in a similar efficiency as ours.

B. Simulated UAV Flight in Challenging Environments

In the attached video (a screenshot is shown in Figure 5), we simulate the UAV flight trajectory in the challenging environment of a museum with a lot of narrow passages. The simulator is setup using [25] on the hardware platform of a CrazyFlies nano-quadrotor. In the museum environment (represented using a dense triangle mesh with 1 million faces) we show that our optimization easily supports the generation of trajectories passing exactly through user-specified waypoints. As demonstrated in the video, our optimization can generate a high quality trajectory that has unnoticeable differences from the simulated path, allowing smooth UAV flying even during sharp turns.

VI. CONCLUSION & LIMITATIONS

We propose a new approach of local UAV-trajectory optimization with guaranteed feasibility and asymptotic convergence to the semi-infinite problem (Equation 3). The key to our success is the primal-only optimizer, P-IPM, equipped with an adaptive subdivision scheme for line search and constraint refinement. The line search scheme preserves the homotopy class and the constraint refinement ensures that our solution approaches that the of semi-infinite oracle with sufficiently small error. Using the same framework, we show that our method can be extended to achieve time-optimality.

We further present an inexact P-IPM approach that achieves up to 10× speedup than its exact counterpart. We also present the convergence analysis showing that both exact and inexact P-IPM algorithms terminate after a finite number of iterations, and they converge to a solution of the same quality, i.e., a solution with sufficiently small gradient norm.

Our method has several limitations, leading to several avenues of future research. First, our method requires a feasible (collision-free) initial guess, which can be computed using sampling-based planners, e.g., RRT*[41]. Second, our method requires the environment to be represented geometrically, which is a common assumption of all optimization-based UAV trajectory planner. We have shown in Section V that our method can handle several different forms of geometric representations, such as point clouds and triangle meshes, without preprocessing. Finally, although our inexact P-IPM is already up to one order of magnitude faster than the exact baseline, our method is still offline. In the future, we plan to explore randomized or first-order numerical algorithms for further speedup.

REFERENCES

[1] T. G. Berry and R. R. Patterson, "The uniqueness of bézier control points," Computer Aided Geometric Design, vol. 14, no. 9, pp. 877–879, 1997.

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

[3] T. Brochu, E. Edwards, and R. Bridson, "Efficient geometrically exact continuous collision detection," ACM Trans. Graph., vol. 31, no. 4, July 2012. [Online]. Available: https://doi.org/10.1145/2185520.2185592

[4] C. Carathéodory, "Über den variabilitätsbereich der fourier'schen konstanten von positiven harmonischen funktionen," Rendiconti Del Circolo Matematico di Palermo (1884-1940), vol. 32, no. 1, pp. 193–217, 1911.

[5] F. E. Curtis and M. L. Overton, "A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization," SIAM Journal on Optimization, vol. 22, no. 2, pp. 474–500, 2012.

[6] 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.

[7] A. Denoyel, C. Pinson, and P.-A. Passet, SketchFab, 2020 (accessed October 25, 2020). [Online]. Available: https://sketchfab.com/

[8] C. Di Franco and G. Buttazzo, "Energy-aware coverage path planning of uavs," in 2015 IEEE international conference on autonomous robot systems and competitions. IEEE, 2015, pp. 111–117.

[9] W. Ding, W. Gao, K. Wang, and S. Shen, "Trajectory replanning for quadrotors using kinodynamic search and elastic optimization," in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 7595–7602.

[10] G. E. Farin and D. Hansford, The Essentials of CAGD, 1st ed. USA: A. K. Peters, Ltd., 2000.

[11] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 3681–3688.

[12] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 344–351.

[13] J. Gravesen, "Adaptive subdivision and the length and energy of bézier curves," Computational Geometry, vol. 8, no. 1, pp. 13–31, 1997.

[14] D. Grundel and D. Jeffcoat, "Formulation and solution of the target visitation problem," in AIAA 1st Intelligent Systems Technical Conference, 2004, p. 6212.

[15] J. A. Guerrero and Y. Bestaoui, "Uav path planning for structure inspection in windy environments," Journal of Intelligent & Robotic Systems, vol. 69, no. 1-4, pp. 297–311, 2013.

[16] K. Hauser, "Semi-infinite programming for trajectory optimization with nonconvex obstacles," in International Workshop on the Algorithmic Foundations of Robotics. Springer, 2018, pp. 565–580.

[17] 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.

[18] S. Kim, K. Sreenath, S. Bhattacharya, and V. Kumar, "Optimal trajectory generation under homology class constraints," in 2012 IEEE 51st IEEE Conference on Decision and Control (CDC), 2012, pp. 3157–3164.

[19] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics," ACM Trans. Graph., vol. 39, no. 4, July 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392425

[20] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," The International Journal of Robotics Research, vol. 35, no. 5, pp. 528–564, 2016.

[21] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling, ser. ICAPS'05. AAAI Press, 2005, p. 262–271.

[22] 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.

[23] O. Mechali, L. Xu, M. Wei, I. Benkhaddra, F. Guo, and A. Senouci, "A rectified rrt* with efficient obstacles avoidance method for uav in 3d environment," in 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2019, pp. 480–485.

[24] 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.

[25] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," IEEE Robotics Automation Magazine, vol. 17, no. 3, pp. 56–65, 2010.

[26] M. Montanari, N. Petrinic, and E. Barbieri, "Improving the gjk algorithm for faster and more reliable distance queries between convex objects," ACM Transactions on Graphics (TOG), vol. 36, no. 3, pp. 1–17, 2017.

[27] B. Mordukhovich and T. Nghia, "Constraint qualifications and optimality conditions for nonconvex semi-infinite and infinite programs," Mathematical Programming, vol. 139, no. 1-2, pp. 271–300, 2013.

[28] R. Ni, T. Schneider, D. Panozzo, Z. Pan, and X. Gao, "Robust & asymptotically locally optimal uav-trajectory generation based on spline subdivision," International Conference on Robotics and Automation (ICRA), 2021.

[29] 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.

[30] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," IEEE Transactions on Robotics, vol. 30, no. 6, pp. 1533–1540, 2014.

[31] H. Prautzsch, W. Boehm, and M. Paluszny, Bezier and B-Spline Techniques. Berlin, Heidelberg: Springer-Verlag, 2002.

[32] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in Robotics Research. Springer, 2016, pp. 649–666.

[33] R. T. Rockafellar, Convex analysis. Princeton university press, 2015.

[34] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," The International Journal of Robotics Research, vol. 33, no. 9, pp. 1251–1270, 2014.

[35] N. Slegers, J. Kyle, and M. Costello, "Nonlinear model predictive control technique for unmanned air vehicles," Journal of guidance, control, and dynamics, vol. 29, no. 5, pp. 1179–1188, 2006.

[36] W. Sun, G. Tang, and K. Hauser, "Fast uav trajectory optimization using bilevel optimization with analytical gradients," in 2020 American Control Conference (ACC). IEEE, 2020, pp. 82–87.

[37] G. Tang, W. Sun, and K. Hauser, "Enhancing bilevel optimization for uav time-optimal trajectory using a duality gap approach," in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 2515–2521.

[38] M. Tang, R. Tong, Z. Wang, and D. Manocha, "Fast and exact continuous collision detection with bernstein sign classification," ACM Transactions on Graphics (TOG), vol. 33, no. 6, pp. 1–8, 2014.

[39] O. Von Stryk, "Numerical solution of optimal control problems by direct collocation," in Optimal control. Springer, 1993, pp. 129–143.

[40] Z. Wang, X. Zhou, C. Xu, J. Chu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 4836–4843, 2020.

[41] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," arXiv, pp. arXiv–1205, 2012.

[42] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013, pp. 5054–5061.

[43] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016, pp. 1433–1440.

[44] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 3529–3536, 2019.

[45] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," The International Journal of Robotics Research, vol. 32, no. 9-10, pp. 1164–1193, 2013.

VII. CONVERGENCE OF EXACT P-IPM TO SIP

In this section, we analyze the convergence of Equation 5 to a KKT point of SIP Equation 3. For simplification, we ignore the velocity and acceleration limits, the case with these limits is similar. Here, we consider one piece Bézier trajectory, so w is the same as W. We denote row vector of Bernstein basis as $A(s)$ with $s \in [0,1]$, then the Bézier curve is $A(s)w$.

A. Perturbed Optimality

We first provide a definition of (inexact) KKT point for the SIP problem following [27]. We rewrite Equation 3 as a smooth SIP by augmenting the index set:

$$\operatorname*{argmin}_{w} O(w) \quad \text{s.t.} \ \forall t \in [0,T], q \in \mathcal{E} \quad \text{dist}(p(t), q) \geq d_0, \quad (8)$$

which is equivalent to Equation 3 (without the velocity and acceleration limits). $\text{dist}(p(t), q)$ is the distance between two points, which is a smooth function when $p(t) \neq q$. We abbreviate $O(w, T)$ as $O(w)$ from now on. The exact KKT condition at w requires an index set $I(w) = \{\langle t, q \rangle \mid \text{dist}(p(t, w), q) = d_0\}$ on which:

$$
\begin{aligned}
0 = &\nabla_w O(w) + \sum_{\langle t, q \rangle \in I} \lambda_{\langle t, q \rangle} \nabla_w \text{dist}(p(t, w), q) \\
&\text{s.t. } \lambda_{\langle t, q \rangle} \leq 0.
\end{aligned}
\tag{9}
$$

Equation 9 is the exact KKT condition. But in practice, a solution satisfying Equation 9 will never be reached because it requires a possibly infinite index set $I(w)$. Instead, we will show that our practical algorithm produces a solution that satisfies a perturbed KKT condition as defined below. We will show later that the perturbed KKT condition approaches Equation 9 in the limit.

Definition 7.1: A perturbed KKT condition, denoted as $\text{KKT}(\epsilon_1, \epsilon_2)$ where $\epsilon_1 > 0$ and $\epsilon_2 > 0$, could be defined on an inexact index set $\tilde{I}(w) = \{\langle t, q \rangle \mid d_0 - \epsilon_1 \leq \text{dist}(p(t, w), q) \leq d_0 + \epsilon_1\}$ on which:

$$
\begin{aligned}
0 = &\nabla_w O(w) + \sum_{\langle t, q \rangle \in \tilde{I}} \lambda_{\langle t, q \rangle} \nabla_w g_{\langle t, q \rangle}(w) \\
&\text{s.t. } \lambda_{\langle t, q \rangle} \leq 0 \\
&\| \nabla_w g_{\langle t, q \rangle}(w) - \nabla_w \text{dist}(p(t, w), q) \| \leq \epsilon_2,
\end{aligned}
$$

where $g_{\langle t, q \rangle}$ is some function of w that satisfy the above conditions. In our setup, $g_{\langle t, q \rangle}(w)$ will be the distance offset by d_0 as shown in the next section.

Remark 1: We have abused notation in the above KKT condition and used a summation over an (possibly) infinite set: $\langle t, q \rangle \in \tilde{I}$, which is well-defined due to the Carathéodory's theorem [4]. In other words, we could always choose a finite subset of $\tilde{I}$ and a corresponding set of Lagrangian multipliers such that the KKT condition still holds.

B. Solution Sequence

We then show that Algorithm 4 returns a solution that satisfies the perturbed KKT condition, assuming $\epsilon_g = 0$. With each set of parameters λ, x_0, ϵ, and from initial guess w_0, Algorithm 4 will return a solution denoted as $w(\lambda, x_0, \epsilon, w_0)$. By introducing another parameter $\beta \in (0, 1)$, we can construct an infinite sequence $\{w_k \mid k \geq 1$ such that $w_k \triangleq w(\beta^k \lambda, \beta^k x_0, \beta^k \epsilon, w_{k-1})$. If we can solve P-IPM to local minimum, then the following function diminishes at w_k:

$$
\begin{aligned}
0 = &\nabla_w O(w_k) + \sum_{i \in S_k} \lambda_i^k \nabla g_i(w_k) \\
&\lambda_i^k = \beta^k \lambda \nabla \text{clog}(g_i(w_k)) \leq 0,
\end{aligned}
\tag{10}
$$

where $g_i(w)$ are functions of form $\text{dist}(\bullet, \bullet) - d_0$, and S_k is the set of indices of non-zero log-barrier terms. From the definition of clog, it is obvious that $0 \leq g_i(w_k) \leq \beta^k x_0$ for all $i \in S_k$.

Based on Assumption 4.1, it's obvious that $\{w_k \mid k \geq 1\}$ and $\{p(t, w_k) \mid t \in [0, T], k \geq 1\}$ are uniformly bounded. Therefore, the following lemma is obvious, which essentially

shows that w_k progressively approach KKT with a diminishing error:

Lemma 7.2: Assuming Assumption 4.1, $\text{dist}(x, y)$ is a Lipschitz-continuous function in x with constant L_1, and $\nabla_x \text{dist}(x, y)$ is another Lipschitz-continuous function in x with constant L_2; then w_k satisfies $\text{KKT}(L_1 \beta^k \epsilon + \beta^k x_0, L_2 \beta^k \epsilon)$.

Proof: We consider the special form of $g_i(w_k) = \text{dist}(\bullet, \bullet) - d_0$. Although $\bullet$ can be a point, a line segment, or a triangle, there must be a point on the sub-trajectory's convex hull p_i^k and a point $q_i^k \in \mathcal{E}$ that realize g_i, i.e. $g_i(w_k) = \text{dist}(p_i^k, q_i^k) - d_0$ and $\nabla g_i(w_k) = \nabla_w \text{dist}(p_i^k, q_i^k)$. Further, due to our subdivision condition, there must be a point on the exact trajectory $p(t_i^k, w_k)$ such that $\|p_i^k - p(t_i^k, w_k)\| < \beta^k \epsilon$. In other words, each $g_i(w_k)$ is associated with three variables: a point on sub-trajectory's convex hull p_i^k, a point on the exact trajectory $p(t_i^k, w_k)$, and a point on the environment q_i^k. We have the following results:

$$
\begin{aligned}
&- L_1 \beta^k \epsilon + d_0 \\
\leq &- \|\text{dist}(p(t_i^k, w_k), q_i^k) - \text{dist}(p_i^k, q_i^k)\| + \text{dist}(p_i^k, q_i^k) \\
\leq &\text{dist}(p_i^k, q_i^k) \leq \text{dist}(p(t_i^k, w_k), q_i^k) \\
\leq &\|\text{dist}(p(t_i^k, w_k), q_i^k) - \text{dist}(p_i^k, q_i^k)\| + \text{dist}(p_i^k, q_i^k) \\
\leq &L_1 \beta^k \epsilon + \beta^k x_0 + d_0
\end{aligned}
$$

$$
\begin{aligned}
&\| \nabla g_i(w_k) - \nabla_w \text{dist}(p(t_i^k, w_k), q_i^k) \| \\
= &\| \nabla_w \text{dist}(p_i^k, q_i^k) - \nabla_w \text{dist}(p(t_i^k, w_k), q_i^k) \| \leq L_2 \beta^k \epsilon,
\end{aligned}
$$

so w_k satisfies $\text{KKT}(L_1 \beta^k \epsilon + \beta^k x_0, L_2 \beta^k \epsilon)$ by choosing $\tilde{I}(w_k) = \{\langle t_i^k, q_i^k \rangle\}$, $g_{\langle t_i^k, q_i^k \rangle} = g_i$, and $\lambda_{\langle t_i^k, q_i^k \rangle} = \lambda_i^k$. ∎
Next, we consider a convergent subsequence $\{w_{k(j)} \mid j \geq 1\} \to w_*$ (there must be at least one due to Assumption 4.1). Although the exactness of KKT satisfaction at $w_{k(j)}$ can be made arbitrarily close to zero, we emphasize that w_* might not satisfy KKT condition of Equation 8, of which a counterexample can be easily constructed at some w_* without the linear independence constraint qualification (LICQ). Therefore, to establish first-order optimality at w_*, we need to design additional qualifications at local optima. Specifically we assume the boundedness of approximate Lagrangian multipliers:

Assumption 7.3: For a constant L_3 independent of k, we have $\|\nabla \text{clog}(g_i(w_k))\| \leq L_3$.

C. Subdivision Limit

We now switch gears and show an intermediary result that, after subdivision, the number of log-barrier terms grows reciprocally with β^k. This result is due to the fact that the arc length of a Bézier curve is upper bounded by length of its control polygon according to [13], which will be used to prove the KKT satisfaction at w_*.

Lemma 7.4: Under Assumption 4.1, there exists a constant L_4 independent of k, such that for any convergent subsequence $\{w_{k(j)} \mid j \geq 1\} \to w_*$, the number of log-barrier terms involved in Equation 9 is upper bounded by $L_4/\beta^{k(j)}$, i.e. $|S_{k(j)}| < L_4/\beta^{k(j)}$.

Proof: To show this result, we need to adopt a similar technique as [13]. We use the same notations as those in the main paper, i.e. the UAV trajectory is represented by N Bézier curves each of order M with parameter $s \in [0, 1]$. For simplicity and without a loss of generality, we assume $N = 1$. We introduce the arc-length of a Bézier curves as:

$$\Sigma(w) = \int_0^1 \|\dot{A}(s)w\| ds.$$

Notice that $\Sigma(w)$ is subdivision invariant. In other words, summing up $\Sigma(w)$ for each sub-curve after subdivision yields the arc-length of the entire curve. By the convergence of $w_{k(j)}$ and the continuity of $\Sigma(w)$, we can further choose a sufficiently large number j_Σ such that $\Sigma(w_{k(j)}) < 2\Sigma(w_*)$ for all $j > j_\Sigma$.

Next, we need to bound the arc-length of each sub-curve. By Cauchy-Schwarz inequality, we have:

$$\begin{aligned}
\Sigma(w) = \int_0^1 \|\dot{A}(s)w\| ds &\geq \sqrt{\int_0^1 \|\dot{A}(s)w\|^2 ds} \\
&= \sqrt{w^T \left[\int_0^1 \dot{A}(s)^T \dot{A}(s) ds\right] w} \triangleq \sqrt{w^T H w} \\
&\triangleq \sqrt{w^T D^T \bar{H} D w} \geq \sigma_{\min}(\sqrt{\bar{H}}) \|Dw\|.
\end{aligned} \tag{11}$$

Here D is the stencil that extracts the edges of control polygon, defined as:

$$D \triangleq \begin{pmatrix} I & -I & & & \\ & I & -I & & \\ & & \ddots & \ddots & \\ & & & I & -I \end{pmatrix}.$$

Since the derivative of a Bézier curve is another Bézier curve with control points being Dw, we must have $H = D^T \bar{H} D$ for some positive semi-definite matrix $\bar{H}$. We further emphasize that $\sigma_{\min}(\sqrt{\bar{H}}) > 0$, i.e. $\bar{H}$ is strictly positive definite. This is because the derivative of a Bézier curve of order M is another Bézier curve of order $M - 1$, with control points being Dw. In addition, a Bézier curve is uniquely defined by its control polygons [1], so that if $w^T D^T \bar{H} D w = 0$, then $Dw = 0$, which further implies that $\bar{H}$ has a full rank.

We then bound the term $\|Dw\|$, which can be done using our subdivision rule. We first notice that $\|Dw\|$ is the total length of control polygon and we know that the total length of control polygon must be larger than the diameter of a control polygon $\Delta(A, w)$. (This is because the diameter of a polytope is realized on two vertices, the distance between which is smaller than the path connecting all vertices.) We consider a subdivision that is done on $A(s)w$ to derive two sub-curves with control points: $D_{1,2}w$, respectively. If $\|DD_1w\| \leq \beta^{k(j)}\epsilon \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2))$, then

$$\begin{aligned}
\Delta(A, w) &\leq \|Dw\| \leq \|DD_1w\| / \sigma_{\min}(D_1) \\
&\leq \beta^{k(j)}\epsilon \frac{\min(\sigma_{min}(D_1), \sigma_{\min}(D_2))}{\sigma_{\min}(D_1)} \leq \beta^{k(j)}\epsilon,
\end{aligned}$$

and a similar result holds for $\|DD_2w\|$. However, it is impossible for $\Delta(A, w) \leq \beta^{k(j)}\epsilon$ because our subdivision rule will prevent further subdivision. As a result, we must have

that every curve that is subdivided to the limit satisfies the following condition:

$$\|Dw\| \geq \beta^{k(j)}\epsilon \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2)).$$

Combining this result with Equation 11, we have the following lower bound of the arc-length of each subdivided Bézier curve piece:

$$\beta^{k(j)}\epsilon \sigma_{\min}(\sqrt{\bar{H}}) \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2)). \tag{12}$$

Finally, we can define L_4 as follows:

$$\begin{aligned}
L_4 \triangleq &\left\lceil \frac{2\Sigma(w_*)}{\epsilon \sigma_{\min}(\sqrt{\bar{H}}) \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2))} \right\rceil \\
&\left[|\mathcal{L}| \frac{(M+1)M}{2} + |\mathcal{P}| \frac{(M+1)M(M-1)}{6} + |\mathcal{T}|(M+1) \right],
\end{aligned}$$

and prove by contradiction. If we can find infinitely many j such that $|S_{k(j)}| \geq L_4/\beta^{k(j)}$. Note that the distance between a convex hull and the environment boils down to finitely many distance between geometric primitive pairs, where the number of terms are upper bounded by:

$$|\mathcal{L}| \frac{(M+1)M}{2} + |\mathcal{P}| \frac{(M+1)M(M-1)}{6} + |\mathcal{T}|(M+1).$$

By the Pigeonhole principle, the number of subdivided Bézier curve pieces is at least:

$$\left\lfloor \frac{1}{\beta^{k(j)}} \left\lceil \frac{2\Sigma(w_*)}{\epsilon \sigma_{\min}(\sqrt{\bar{H}}) \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2))} \right\rceil \right\rfloor. \tag{13}$$

By Equation 12 and Equation 13, the total arc-length satisfies $\Sigma(w_{k(j)}) \geq 2\Sigma(w_*)$ for infinitely many j, which contradicts the convergence of $\Sigma(w_{k(j)})$ to $\Sigma(w_*)$. ∎

D. KKT Condition at w_*

We can finally prove by contradiction that KKT condition (Equation 9) holds at w_*. It is obvious that w_* is a feasible point of Equation 8. Suppose otherwise that KKT condition does not hold at w_*, then there must be a unit direction v such that:

$$\begin{aligned}
v^T \nabla_w O(w_*) &= \epsilon_3 < 0 \\
v^T \nabla_w \text{dist}(p(t, w_*), q) &\geq 0 \quad \forall \langle t, q \rangle \in I(w_*).
\end{aligned} \tag{14}$$

Remark 2: To show that Equation 14 holds whenever the KKT condition (Equation 9) fails, let us consider the convex cone $\mathbb{C} = \{\sum_{\langle t,q \rangle \in I} \lambda_{\langle t,q \rangle} \nabla_w \text{dist}(p(t, w), q) | \lambda_{\langle t,q \rangle} \leq 0\}$. The failure of Equation 9 implies that $-\nabla_w O$ does not belong to $\mathbb{C}$. If $\mathbb{C}$ is closed, we can then project $-\nabla_w O$ back to the cone along some normal vector v, which is the vector we are looking for in Equation 14 (see e.g., [33]).

Remark 3: Informally, we take three steps to show that $\mathbb{C}$ in Remark 2 is closed. First, we define another set $\bar{\mathbb{C}} = \{\nabla_w \text{dist}(p(t, w), q) | \langle t, q \rangle \in I\}$, which is obviously bounded. We argue that $\bar{\mathbb{C}}$ is also closed. This is because a Bézier curve can only touch a triangle mesh in finitely many closed segments, $\nabla_w \text{dist}$ is a continuous function, and a finite union of closed sets is again closed. We conclude that $\bar{\mathbb{C}}$ is compact. Second, we have $\mathbb{C} = \text{Cone}(\text{Conv}(-\bar{\mathbb{C}}))$ and the convex hull of a compact set is compact in $\mathbb{R}^n$ so $\text{Conv}(-\bar{\mathbb{C}})$ is compact and convex. Finally, the cone of a compact convex set is closed so $\mathbb{C}$ is closed.

We prove that v must be a descend direction at $w_{k(j)}$ for sufficiently large j, contradicting the fact that $w_{k(j)}$ is a local minimum of P-IPM. Note that for any point $p(t_i^{k(j)}, w_{k(j)})$, it is not possible for $\text{dist}(p(t_i^{k(j)}, w_{k(j)}), q_i^{k(j)}) = d_0$ as the log-barrier will be infinite otherwise, i.e. we cannot reach any point in the index set of w_*, but we can bound their distance using the following lemma:

Lemma 7.5: Under Assumption 4.1, given any $\epsilon_4 > 0$, there is a large enough $j(\epsilon_4)$, such that for all $j \geq j(\epsilon_4)$ and $\text{clog}(g_i(w_{k(j)})) > 0$, we can find some $\langle t, q \rangle \in I(w_*)$ satisfying:

$$\|p(t_i^{k(j)}, w_{k(j)}) - p(t, w_*)\| \leq \epsilon_4.$$

Proof: We proof by contradiction. Suppose otherwise, there exists some $\epsilon_4 > 0$ such that for any j_0, we can always find some $j > j_0$ and $\text{clog}(g_{i(j)}(w_{k(j)})) > 0$ where $\|p(t_{i(j)}^{k(j)}, w_{k(j)}) - p(t, w_*)\| > \epsilon_4$ for all $\langle t, q \rangle \in I(w_*)$. Then we can construct an infinite sub-sequence of form $\{p(t_{i(j)}^{k(j)}, w_{k(j)})\}$ in which every point is at least ϵ_4 away from the index set $I(w_*)$. Due to Assumption 4.1, there will be a convergent subsequence of $\{p(t_{i(j)}^{k(j)}, w_{k(j)})\}$, the limit of which denoted as $p(t_*, w_*)$. We have $\text{dist}(p(t_*, w_*), \mathcal{E}) = 0$ so $p(t_*, w_*)$ belongs to the index set. Further, $p(t_*, w_*)$ is at least ϵ_4 away from the index set, a contradiction. ∎

We can now present our main result, which involves bounding each term in Equation 10:

Proposition 7.6: Under Assumption 4.1 and Assumption 7.3, for any convergent subsequence $\{w_{k(j)} \mid j \geq 1\} \to w_*$, a unit vector satisfying Equation 14 is a descend direction at $w_{k(j)}$ for sufficiently large j.

Proof: We have bounded w under Assumption 4.1 and $t \in [0, T]$ is also bounded. As a result, a point on the curve, namely $p(t, w)$, is a continuous function on the bounded domain, which is also Lipschitz continuous. For the same reason, $A(t)$ is Lipschitz continuous. We denote the Lipschitz constant for these two functions as L_5. We start our proof by bounding the constraint gradient ∇g_i. We divide the directional derivative of ∇g_i along v into three terms, each equipped with a diminishing bound:

$$
\begin{aligned}
&v^T \nabla g_i(w_{k(j)}) \\
=&v^T(\nabla g_i(w_{k(j)}) - \nabla_w \text{dist}(p(t_i^{k(j)}, w_{k(j)}), q_i^{k(j)}))+ \\
&\text{(first term)} \\
&v^T(\nabla_w \text{dist}(p(t_i^{k(j)}, w_{k(j)}), q_i^{k(j)}) - \nabla_w \text{dist}(p(t, w_*), q))+ \\
&\text{(second term)} \\
&v^T \nabla_w \text{dist}(p(t, w_*), q) \quad \text{(third term)} \\
\geq& -L_2 \beta^{k(j)} \epsilon - L_2 \|p(t_i^{k(j)}, w_{k(j)}) - p(t, w_*)\| \\
\geq& -L_2 \beta^{k(j)} \epsilon - L_2 \epsilon_4,
\end{aligned}
$$

For the first term above, we use our subdivision rule and continuity of $\nabla_w \text{dist}$. For the second term, we use Lemma 7.5 by choosing $j > j(\epsilon_4)$. The third term is bigger than zero due to v violating the KKT condition (Equation 14). Next, we can invoke Lemma 7.4 and show that:

$$v^T \sum_{i \in S_{k(j)}} \lambda_i^{k(j)} \nabla g_i(w_{k(j)}) \leq L_3 L_4 \lambda (L_2 \beta^{k(j)} \epsilon + L_2 \epsilon_4).$$

Finally, our algorithm only consider twice-differentiable objective function O. Under assumption Assumption 4.1, $\nabla_w O$

is Lipschitz continuous in the bounded domain of w with constant L_6. As a result, we have:

$$
\begin{aligned}
v^T \nabla_w O(w_{k(j)}) =& v^T(\nabla_w O(w_{k(j)}) - \nabla_w O(w_*)) + v^T \nabla_w O(w_*) \\
\leq& \epsilon_3 + L_6 \|w_{k(j)} - w_*\|.
\end{aligned}
$$

Combining all the results above, we conclude that:

$$
\begin{aligned}
&v^T \nabla_w \big[O(w_{k(j)} + \lambda B(w_{k(j)}, \mathcal{E})\big] \\
\leq& \epsilon_3 + L_6 \|w_{k(j)} - w_*\| + L_3 L_4 \lambda (L_2 \beta^{k(j)} \epsilon + L_2 \epsilon_4),
\end{aligned}
$$

which can be made arbitrarily close to ϵ_3 by choosing large enough j, contradicting the fact that $w_{k(j)}$ is a local minimum of P-IPM. ∎

We can now claim our main result:

Corollary 7.7: Any accumulation point of the sequence $\{w_k\}$ satisfies the KKT condition of SIP (Equation 9) under Assumption 4.1 and Assumption 7.3.

VIII. FINITE TERMINATION OF EXACT P-IPM

In the above convergence analysis of exact P-IPM to SIP, we assume that $\epsilon_g = 0$. However, a practical algorithm needs to have finite termination. In this section, we show that as long as $\epsilon_g > 0$, the exact P-IPM Algorithm 4 always converges in a finite number of iterations. Finite termination happens under two conditions:

- The while loop in line-search Algorithm 5 is finite.
- The outer loop in Algorithm 4 is finite.

Although the finite termination of Newton's method has been shown in [2], we have used a special safeguard in our line search (Line 5 in Algorithm 5), which is a new setting whose convergence needs to be shown.

A. Finite Termination of Line Search

Lemma 8.1: Algorithm 5 terminates after finitely many iterations, if started from a w with finite objective function value.

Proof: Consider any point $p \in \text{Hull}(\mathcal{P}'(w))$, which is a convex combination of its $N + 1$ vertices. If we update w to $w' = w + \alpha d$ along some descendent direction d, then p will be moved to p' with $\text{dist}(p, p') \leq \alpha L_1 L_5 \|d\|$. If we choose:

$$\alpha \leq \frac{\text{dist}(\text{Hull}(\mathcal{P}'(w)), \mathcal{E}) - d_0}{2 L_1 L_5 \|d\|}, \tag{15}$$

then we have:

$$
\begin{aligned}
&\text{dist}(l'_{pp'}, \mathcal{E}) \\
\geq& \text{dist}(p, \mathcal{E}) - \text{dist}(p, p') \\
\geq& \frac{\text{dist}(\text{Hull}(\mathcal{P}'(w)), \mathcal{E}) + d_0}{2} > d_0,
\end{aligned}
$$

where $l'_{pp'}$ is a line segment connecting p and p'. Since p is an arbitrary point in the convex hull, we thus have:

$$\text{dist}(\text{Hull}(\mathcal{P}'(w), \mathcal{P}'(w')), \mathcal{E}) > d_0,$$

which implies that the safeguard will pass as long as Equation 15 holds. Due to the finite objective function value, the righthand side of Equation 15 is positive. Then by the differentiability of the objective function, a positive α satisfying Wolfe's first condition can be found. ∎

B. Finite Termination of Outer Loop

We have the following corollary of this assumption based on Assumption 4.1:

Corollary 8.2: Suppose Assumption 4.1 holds (w is bounded). For each w generated by Algorithm 4, we have:

$$\text{dist}(\text{Hull}(\mathcal{P}'(w)), \mathcal{E}) \geq d_0 + L_7,$$

with some $L_7 > 0$ independent of iteration number.

Proof: If no such L_7 can be found, then $\text{dist}(\text{Hull}(\mathcal{P}'(w)), \mathcal{E}) - d_0 < L_7$ for any sufficiently small L_7. If we choose a sequence of $L_7 \to 0$, we have the objective function $O(w, T) + \lambda B(w, \mathcal{E}) \geq \lambda \text{clog}(L_7) \to \infty$. This contradicts the monotonic decrease of the objective function. ∎

Corollary 8.3: Suppose Assumption 4.1 holds. For each w generated by Algorithm 4, we have:

$$\left\| \frac{\partial O(w, T)}{\partial w} + \lambda \frac{\partial B(w, \mathcal{E})}{\partial w} \right\| < L_8,$$

with some $L_8 > 0$ independent of iteration number.

Proof: Since w is bounded and O is twice-differentiable, its gradient is bounded. By a similar argument as Lemma 7.4, we have that the number of subdivided Bézier curve pieces is at most:

$$\left\lceil \frac{\|Dw\| \sigma_{\max}(\sqrt{\overline{H}})}{\epsilon \sigma_{\min}(\sqrt{\overline{H}}) \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2))} \right\rceil$$

Combining these two results, we have:

$$\left\| \frac{\partial O(w, T)}{\partial w} + \lambda \frac{\partial B(w, \mathcal{E})}{\partial w} \right\|$$
$$\leq \left\| \frac{\partial O(w, T)}{\partial w} \right\| + \lambda \left\lceil \frac{\|Dw\| \sigma_{\max}(\sqrt{\overline{H}})}{\epsilon \sigma_{\min}(\sqrt{\overline{H}}) \min(\sigma_{\min}(D_1), \sigma_{\min}(D_2))} \right\rceil$$
$$\left[|\mathcal{L}| \frac{(M+1)M}{2} + |\mathcal{P}| \frac{(M+1)M(M-1)}{6} + |\mathcal{T}|(M+1) \right]$$
$$\left| \frac{\partial \text{clog}(d)}{\partial d} \right|_{d=d_0+L_7} \left\| \frac{\partial \text{dist}(p(t,w), \bullet)}{\partial w} \right\|,$$

where $\bullet$ means any geometric primitive and we have used Corollary 8.2 to bound $\|Dw\|$. Further, it can be shown that the gradient norm $\partial \text{clog}(d)/\partial d$ is monotonically decreasing in d, so it can be lower-bounded to the gradient norm at $d_0 + L_7$ due to Corollary 8.2. $\partial O(w, T)/\partial w$ is bounded by L_6 due to Assumption 4.1. $\partial \text{dist}(p(t, w))/\partial w$ is bounded because:

$$\left\| \frac{\partial \text{dist}(p(t,w))}{\partial w} \right\| \leq \left\| \frac{\partial \text{dist}(p(t,w))}{\partial p(t,w)} \right\| \left\| \frac{\partial p(t,w)}{\partial w} \right\| \leq L_2 L_5.$$
∎

Proof of Proposition 4.2: Similar to the proof of Lemma 8.1, we consider every point $p \in \text{Hull}(\mathcal{P}'(w))$, which is a convex combination of its $N + 1$ vertices. If we update w to $w' = w + \alpha d$, then p will be moved to p' with $\text{dist}(p, p') \leq \alpha L_1 L_5 L_8$, where we have used Corollary 8.3 to bound d. By Corollary 8.2 we have $\text{dist}(l'_{pp'}, \mathcal{E}) > d_0 + L_7$, so triangle inequality implies that:

$$\text{dist}(l'_{pp'}, \mathcal{E}) \geq \text{dist}(p, \mathcal{E}) - \text{dist}(p, p') \geq d_0 + L_7 - \alpha L_1 L_5 L_8.$$

Therefore, our line search will always pass the safeguard test, if we choose $\alpha \leq L_7/(L_1 L_5 L_8)$. The finite termination follows by choosing a stepsize sequence with each term smaller than $L_7/(L_1 L_5 L_8)$ and [2, Proposition 1.2.4]. ∎

Note that we have only analyzed gradient descend method (i.e. $\text{SPD}(\nabla^2_{w,T} O) = I$), the case with Newton's method is almost identical, assuming the Hessian matrix has bounded singular values.

IX. FINITE TERMINATION OF INEXACT P-IPM

The convergence analysis for inexact P-IPM is more obscure. In this setting, the safeguard algorithm measures the distance based on convex hull, while the objective function simplifies the convex hull to a line segment. This inconsistent geometric representation might invalidate finite termination in the line-search Algorithm 5. In the following sections, we prove that the inexact P-IPM algorithm has finite termination. The core to our proof is the derivation of the error between $\tilde{B}$ and $\hat{B}$ can be bounded.

Remark 4: Our definition of $\tilde{B}$ and $\hat{B}$ is not differentiable for general environment mesh $\mathcal{E}$. A differentiable version can be derived by breaking up $\mathcal{E}$ into edges and triangles. For simplicity of proof, we assume that $\mathcal{E}$ is a single edge throughout this section. The proof can be applied to general environment meshes by adding separate terms to both $\tilde{B}$ and $\hat{B}$ for each edge and triangle.

A. Well-Defined Integral of Log-Barrier

We have used a different version of clog from its original definition in [19], for which the reason is to ensure the well-definedness of $\hat{B}$. A well-defined log-barrier function should take infinite value when any part of the Bézier curve intersects $\mathcal{E}$. We present a counter example in Figure 6 to show that the original log function in [19] does not pertain this property. Indeed, the value of $\hat{B}$ in Figure 6 can be evaluated analytically as follows:

$$\int_0^1 -(|s - 0.5| - 0.5)^2 \log \frac{|s - 0.5|}{0.5} ds = \frac{11}{72} < \infty,$$

where we choose $x_0 = 0.5$. For our modified clog function, we prove well-definedness in the following lemma:

Lemma 9.1: If $\text{dist}(A(s)w, \mathcal{E}) = d_0$ for some $s \in (0, 1)$, then $\hat{B} = \infty$.

Proof: Consider the closed interval $[s - \epsilon_5, s + \epsilon_5] \subset (0, 1)$ for some sufficiently small ϵ_5. The value of dist is upper bounded by:

$$0 \leq \text{dist}(A(s')w, \mathcal{E}) \leq L_1 L_5 \epsilon_5 + d_0,$$

where $s' \in [s - \epsilon_5, s + \epsilon_5]$. The sub-integral of $\hat{B}$ in range $[s - \epsilon_5, s + \epsilon_5]$ is thus lower bounded by:

$$\hat{B}(w, \mathcal{E}) \geq \int_{s-\epsilon_5}^{s+\epsilon_5} \text{clog}(\text{dist}(A(s)w, \mathcal{E}) - d_0) ds$$
$$\geq 2 L_1 L_5 \epsilon_5 \text{clog}(L_1 L_5 \epsilon_5)$$
$$= -2(L_1 L_5 \epsilon_5 - x_0)^2 \log(\frac{L_1 L_5 \epsilon_5}{x_0}),$$

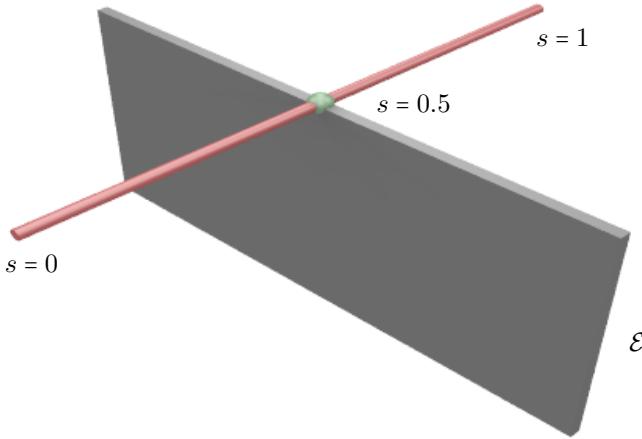the righthand side of which tends to ∞ as $\epsilon_5 \to 0$. ∎

Fig. 6: In this counter example, the red line is the UAV's Bézier curve and the gray horizontal wall is the environment, touching at exactly one green point. However, $\hat{B}$ takes a finite value if clog from [19] is used.

B. Adaptive Subdivision

A potential issue in using $\hat{B}$ for our analysis is that $\tilde{B}$ approaches $\hat{B}$ only under uniform subdivisions (i.e. subdividing every curve piece from s_i to s_{i+1}). However, we use adaptive subdivisions. To bridge this gap, we need the following definition for further analysis:

$$\hat{B}_{\epsilon_6}(w,\mathcal{E}) \triangleq \sum_{i=1}^{S-1}$$
$$\begin{cases} \int_{s_i}^{s_{i+1}} \text{clog}(\text{dist}(A(s)w,\mathcal{E}) - d_0)ds & s_{i+1} - s_i < \epsilon_6 \\ (s_{i+1} - s_i)\text{clog}(\text{dist}(m'_{i,i+1}(w),\mathcal{E}) - d_0) & s_{i+1} - s_i \geq \epsilon_6. \end{cases}$$

Intuitively, $\hat{B}_{\epsilon_6}(w,\mathcal{E})$ combines $\tilde{B}$ and $\hat{B}$ based on the length of each subdivided curve piece. A pivotal property of $\hat{B}$ is that it is invariant to subdivision. Similarly, we can show that $\hat{B}_{\epsilon_6}$ is invariant to subdivision after sufficiently large number of calls to Algorithm 6:

Lemma 9.2: For any fixed ϵ_6, after sufficiently but finitely many calls to Algorithm 6, $\hat{B}_{\epsilon_6}$ is invariant to subdivision.

Proof: It takes finitely many calls to Algorithm 6 to bring every $s_{i+1} - s_i < \epsilon_6$. ∎

We can now bound the error between $\tilde{B}$ and $\hat{B}_{\epsilon_6}$ by controlling ϵ_6 as follows:

Lemma 9.3: Taking Assumption 4.1, if w is generated by some iteration of Algorithm 8, then we have the following bound:

$$|\tilde{B} - \hat{B}_{\epsilon_6}| \leq \epsilon_6 \left[L_9 \frac{\log(\psi(\epsilon_6))}{\psi(\epsilon_6)^2} + L_{10} \right] L_1 L_5 = \mathcal{O}(\epsilon_6^{1-2\eta}\log(\epsilon_6)),$$

for some negative constant L_9 and positive constant L_{10}. The bound is diminishing as $\epsilon_6 \to 0$ if $\eta < 1/2$.

Proof: If w is generated by some iteration of Algorithm 8, then every Bézier curve piece must have passed the safeguard check inside the line search. For a curve piece satisfying $s_{i+1} - s_i \geq \epsilon_6$, $\tilde{B}$ and $\hat{B}_{\epsilon_6}$ are the same on that segment. For a curve piece satisfying $s_{i+1} - s_i < \epsilon_6$, instead, we have from the safeguard condition that:

$$\text{dist}(A(s)w,\mathcal{E}) - d_0 \geq \psi(s_{i+1} - s_i) \quad \forall s \in [s_i, s_{i+1}].$$

For any two parameters $s_1, s_2 \in [s_i, s_{i+1}]$, we have from the mean value theorem that:

$$|\text{clog}(\text{dist}(A(s_1)w,\mathcal{E}) - d_0) - \text{clog}(\text{dist}(A(s_2)w,\mathcal{E}) - d_0)|$$
$$= \left| \int_{\text{dist}(A(s_1)w,\mathcal{E})-d_0}^{\text{dist}(A(s_2)w,\mathcal{E})-d_0} \frac{\partial \text{clog}(x)}{\partial x} dx \right|$$
$$\leq \left| \frac{\partial \text{clog}(d)}{\partial d} \right|_{\psi(s_{i+1}-s_i)} \left| |\text{dist}(A(s_1)w,\mathcal{E}) - \text{dist}(A(s_2)w,\mathcal{E})| \right.$$
$$\leq \left| \frac{\partial \text{clog}(d)}{\partial d} \right|_{\psi(s_{i+1}-s_i)} |A(s_1)w - A(s_2)w|L_1$$
$$\leq |s_1 - s_2| \left| \frac{\partial \text{clog}(d)}{\partial d} \right|_{\psi(s_{i+1}-s_i)} L_1 L_5. \qquad (16)$$

It can be verified that as $d \to 0$, the dominating term of $\partial \text{clog}(d)/\partial d$ is $\log(d)/d^2$. Therefore, we can find negative constant L_9 and positive constant L_{10} such that:

$$\left| \frac{\partial \text{clog}(d)}{\partial d} \right| \leq L_9 \frac{\log(d)}{d^2} + L_{10} \quad \forall d > 0,$$

which can be plugged into Equation 16 to derive:

$$|\text{clog}(\text{dist}(A(s_1)w,\mathcal{E}) - d_0) - \text{clog}(\text{dist}(A(s_2)w,\mathcal{E}) - d_0)|$$
$$\leq |s_1 - s_2| \left[L_9 \frac{\log(\psi(s_{i+1} - s_i))}{\psi(s_{i+1} - s_i)^2} + L_{10} \right] L_1 L_5.$$

Therefore, the difference between $\tilde{B}$ and $\hat{B}_{\epsilon_6}$ can be bounded on the curve piece as:

$$\left| \int_{s_i}^{s_{i+1}} \text{clog}(\text{dist}(A(s)w,\mathcal{E}) - d_0)ds \right.$$
$$\left. - (s_{i+1} - s_i)\text{clog}(\text{dist}(m'_{i,i+1}(w),\mathcal{E}) - d_0) \right|$$
$$\leq (s_{i+1} - s_i)^2 \left[L_9 \frac{\log(\psi(s_{i+1} - s_i))}{\psi(s_{i+1} - s_i)^2} + L_{10} \right] L_1 L_5$$
$$\leq (s_{i+1} - s_i)\epsilon_6 \left[L_9 \frac{\log(\psi(\epsilon_6))}{\psi(\epsilon_6)^2} + L_{10} \right] L_1 L_5.$$

The difference between $\tilde{B}$ and $\hat{B}_{\epsilon_6}$ is maximized when all the curve pieces satisfy $s_{i+1} - s_i < \epsilon_6$. Combined with the fact that $\sum_{i=1}^{S-1}(s_{i+1}-s_i) = 1$, we derive the bound to be proved. Further, if $\eta < 1/2$, then the leading term of $|\tilde{B} - \hat{B}_{\epsilon_6}|$ is $\epsilon_6^{1-2\eta}\log(\epsilon_6)$, which is diminishing as $\epsilon_6 \to 0$. ∎

Following a similar reasoning, we can also bound the different in gradient:

Lemma 9.4: Taking Assumption 4.1, if w is generated by some iteration of inexact P-IPM, then we have the following bound:

$$\|\nabla_w \tilde{B} - \nabla_w \hat{B}_{\epsilon_6}\|$$
$$\leq \epsilon_6 \left[L_9 \frac{\log(\psi(\epsilon_6))}{\psi(\epsilon_6)^2} + L_{10} \right] (L_2 L_5 \max_{s \in [0,1]} \|A(s)\| + L_5) +$$
$$\epsilon_6 \left[L_{11} \frac{\log(\psi(\epsilon_6))}{\psi(\epsilon_6)^3} + L_{12} \right] (L_2 L_5 \max_{s \in [0,1]} \|A(s)\|)$$
$$= \mathcal{O}(\epsilon_6^{1-3\eta}\log(\epsilon_6)),$$

for some negative constant L_{11} and positive constant L_{12}. The bound is diminishing as $\epsilon_6 \to 0$ if $\eta < 1/3$.

Proof: For a curve piece satisfying $s_{i+1} - s_i \leq \epsilon_6$ and two parameters $s_1, s_2 \in [s_i, s_{i+1}]$, we have from the triangle inequality that:

$$\|\nabla_w \mathrm{clog}(\mathrm{dist}(A(s_1)w, \mathcal{E}) - d_0) - \nabla_w \mathrm{clog}(\mathrm{dist}(A(s_2)w, \mathcal{E}) - d_0)\|$$

$$= \left\| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_1} A(s_1) - \frac{\partial \mathrm{clog}(d)}{\partial d_2} \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) \right\|$$

$$= \left\| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_1} A(s_1) - \frac{\partial \mathrm{clog}(d)}{\partial d_1} \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) + \right.$$

$$\left. \frac{\partial \mathrm{clog}(d)}{\partial d_1} \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) - \frac{\partial \mathrm{clog}(d)}{\partial d_2} \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) \right\|$$

$$\leq \left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \right| \left\| \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_1} A(s_1) - \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) \right\| +$$

$$\left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} - \frac{\partial \mathrm{clog}(d)}{\partial d_2} \right| \left\| \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) \right\|,$$

where $d_1 = \mathrm{dist}(p_1, \mathcal{E})$, $p_1 = p(s_1, w) = A(s_1)w$, and we have the same for d_2, p_2. There are two terms in the above equation to be bounded. For the first term, we have:

$$\left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \right| \left\| \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_1} A(s_1) - \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) \right\|$$

$$\leq \left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \right| \left\| \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_1} - \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} \right\| \|A(s_1)\| +$$

$$\left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \right| \left\| \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} \right\| \|A(s_1) - A(s_2)\|$$

$$\leq \left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \right|_{\psi(s_{i+1}-s_i)} L_2 L_5 (s_{i+1} - s_i) \max_{s \in [0,1]} \|A(s)\| +$$

$$\left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} \right|_{\psi(s_{i+1}-s_i)} L_5 (s_{i+1} - s_i)$$

$$\leq (s_{i+1} - s_i) \left[L_9 \frac{\log(\psi(s_{i+1}-s_i))}{\psi(s_{i+1}-s_i)^2} + L_{10} \right]$$

$$(L_2 L_5 \max_{s \in [0,1]} \|A(s)\| + L_5),$$

where we have used Lemma 9.3 and the property of the eikonal equation that the gradient-norm of a distance function is upper bounded by one or $\|\partial \mathrm{dist}(p, \mathcal{E})/\partial p_2\| \leq 1$. To bound the second term, we need to use mean value theorem again. It can be verified that as $d \to 0$, the dominating term of $\partial^2 \mathrm{clog}(d)/\partial d^2$ is $\log(d)/d^3$. Therefore, we can find negative constant L_{11} and positive constant L_{12} such that:

$$\left| \frac{\partial^2 \mathrm{clog}(d)}{\partial d^2} \right| \leq L_{11} \frac{\log(d)}{d^3} + L_{12} \quad \forall d > 0.$$

We are now ready to invoke the mean value theorem and bound the second term as:

$$\left| \frac{\partial \mathrm{clog}(d)}{\partial d_1} - \frac{\partial \mathrm{clog}(d)}{\partial d_2} \right| \left\| \frac{\partial \mathrm{dist}(p, \mathcal{E})}{\partial p_2} A(s_2) \right\|$$

$$\leq \left| \int_{d_1}^{d_2} \frac{\partial^2 \mathrm{clog}(d)}{\partial d^2} dd \right| \max_{s \in [0,1]} \|A(s)\|$$

$$\leq \int_{d_1}^{d_2} \left[L_{11} \frac{\log(d)}{d^3} + L_{12} \right] dd \max_{s \in [0,1]} \|A(s)\|$$

$$\leq |d_1 - d_2| \max_{d \in [d_1, d_2]} \left[L_{11} \frac{\log(d)}{d^3} + L_{12} \right] \max_{s \in [0,1]} \|A(s)\|$$

$$\leq L_1 L_5 (s_{i+1} - s_i) \max_{d \in [d_1, d_2]} \left[L_{11} \frac{\log(d)}{d^3} + L_{12} \right] \max_{s \in [0,1]} \|A(s)\|$$

$$\leq (s_{i+1} - s_i) \left[L_{11} \frac{\log(\psi(s_{i+1}-s_i))}{\psi(s_{i+1}-s_i)^3} + L_{12} \right] (L_1 L_5 \max_{s \in [0,1]} \|A(s)\|).$$

The rest of the argument is identical to Lemma 9.3. If $\eta < 1/3$, then the leading term of $|\nabla_w \tilde{B} - \nabla_w \hat{B}_{\epsilon_6}|$ is $\epsilon_6^{1-3\eta} \log(\epsilon_6)$, which is diminishing as $\epsilon_6 \to 0$. ∎

C. Finite Termination of Line Search

It is obvious that if line search does not have finite termination, then Subdivide-Unsafe will be invoked for infinitely many times and ϵ_α will get arbitrarily small. We first notice that due to the positivity of ψ, $\hat{B}_{\epsilon_6}$ is finite:

Corollary 9.5: If we choose any fixed ϵ_6 and suppose w is some solution generated by an iteration of inexact P-IPM, then $\hat{B}_{\epsilon_6}(w, \mathcal{E}) < \infty$.

Proof: For each point $A(s)w$, we have: $\mathrm{dist}(A(s)w, \mathcal{E}) - d_0 > \min_{i=1,\cdots,S-1} \psi(s_{i+1} - s_i) > 0$. ∎

We can prove by contradiction that if there are infinitely many unsafe Bézier curve pieces, than $\hat{B}_{\epsilon_6}$ cannot be finite, which is an indication of finite termination of Algorithm 7.

Lemma 9.6: If we choose any fixed ϵ_6 and suppose there are infinitely many calls to Algorithm 6 within Algorithm 8, then $\hat{B}_{\epsilon_6}$ is unbounded.

Proof: Suppose we have infinitely many calls to Algorithm 6, then there must be an unsafe Bézier curve piece satisfying:

$$s_{i+1} - s_i \leq \frac{1}{2^k} < \epsilon_6,$$

for any, sufficiently large, positive integer k. Consider a point $p = A(s)w$ on that unsafe curve piece $s \in [s_i, s_{i+1}]$, which is supposed to be updated to $p' = A(s)w'$ after the line search. We can choose $s \in (s_i, s_{i+1})$ such that:

$$\mathrm{dist}(l'_{pp'}, \mathcal{E}) < d_0 + \psi(s_{i+1} - s_i)$$

$$\Rightarrow \mathrm{dist}(p, \mathcal{E}) \leq \mathrm{dist}(l'_{pp'}, \mathcal{E}) + \mathrm{dist}(p, p')$$

$$\leq d_0 + \psi(s_{i+1} - s_i) + L_2 L_5 \alpha \|d\|.$$

Now consider a small range $[s - \epsilon_5, s + \epsilon_5] \subset (s_i, s_{i+1})$ in which we have:

$$\mathrm{dist}(A(s')w, \mathcal{E}) \leq \mathrm{dist}(p, \mathcal{E}) + L_1 L_5 \epsilon_5$$

$$\leq d_0 + \psi(s_{i+1} - s_i) + L_2 L_5 \alpha \|d\| + L_1 L_5 \epsilon_5,$$

where $s' \in [s - \epsilon_5, s + \epsilon_5]$. We can then bound $\hat{B}_{\epsilon_6}$ from below as:

$$\hat{B}_{\epsilon_6}(w, \mathcal{E}) \geq \int_{s-\epsilon_5}^{s+\epsilon_5} \mathrm{clog}(\mathrm{dist}(A(s')w, \mathcal{E}) - d_0) ds'$$

$$\geq 2\epsilon_5 \mathrm{clog}(\psi(s_{i+1} - s_i) + L_2 L_5 \alpha \|d\| + L_1 L_5 \epsilon_5)$$

$$\geq 2\epsilon_5 \mathrm{clog}(\frac{\epsilon_s}{2^{\eta k}} + L_2 L_5 \epsilon_\alpha \|d\| + L_1 L_5 \epsilon_5).$$

Taking $k \to \infty$, $\epsilon_5 \to 0$, $\epsilon_\alpha \to 0$, and we have $\hat{B}_{\epsilon_6}(w) \to \infty$, contradicting Corollary 9.5. ∎

The finite termination of Algorithm 7 is a direct corollary of the two results above:

Corollary 9.7: Algorithm 7 has finite termination.

Proof: If Algorithm 7 does not have finite termination, then $\hat{B}_{\epsilon_6}$ is unbounded by Lemma 9.6, contradicting Corollary 9.5. ∎

D. Finite Termination of Outer Loop

Similar to the analysis of exact P-IPM, we show finite termination for gradient descend method (i.e. $\mathrm{SPD}(\nabla_{w,T}^2 O) = I$), instead of Newton's method, for simplicity. Unlike exact P-IPM, however, the inexact line search Algorithm 7 is not

guaranteed to reduce function value, because the function value might increase due to Algorithm 6. Instead, we analyze the total number of subdivisions throughout Algorithm 8 in two cases.

Case I: If Algorithm 8 uses finitely many unsafe subdivisions, then we can find a last outer iteration calling Algorithm 6. Afterwards, we are in a similar situation to Proposition 4.2.

Lemma 9.8: Assuming Assumption 4.1, if Algorithm 8 makes finitely many calls to Algorithm 6, then Algorithm 8 terminates after finitely many iterations.

Proof: If there are only finitely many calls to Algorithm 6, then we can denote w as the solution generated by the outer iteration that makes the last call to Algorithm 6. After this iteration, ϵ_a takes some positive value. From this iteration onward, ϵ_a will never change and ϵ_a is always safe, because there will be another call to Algorithm 6 otherwise. Therefore, we have finite termination by [2, Proposition 1.2.4]. ∎

Case II: If Algorithm 8 uses infinitely many unsafe subdivisions, then $\hat{B}_{\epsilon_6}$ increases but $\tilde{B}$ decreases during an update from w to w'. But Lemma 9.3 and Lemma 9.4 shows that $\hat{B}_{\epsilon_6}$ and $\tilde{B}$ can be arbitrarily close, from which we can derive a contradiction.

Lemma 9.9: Assuming Assumption 4.1, Algorithm 8 makes finitely many calls to Algorithm 6.

Proof: Let's use the following abbreviations of the objective functions:

$$\tilde{f}(w) \triangleq O(w, T) + \lambda \tilde{B}(w, \mathcal{E})$$
$$\hat{f}_{\epsilon_6}(w) \triangleq O(w, T) + \lambda \hat{B}_{\epsilon_6}(w, \mathcal{E}).$$

If there are infinitely many calls to Algorithm 6, then $\|\nabla_w \tilde{f}\| > \epsilon_g$ because the outer loop would terminate immediately otherwise. Consider a update from w to w' which satisfies the Wolfe's condition, we have:

$$\tilde{f}(w') \leq \tilde{f}(w) - c\alpha \|\nabla_w \tilde{f}\|^2 \leq \tilde{f}(w) - c\alpha \|\nabla_w \tilde{f}\| \epsilon_g.$$

We can bound the corresponding change in $\hat{f}_{\epsilon_6}$ using Lemma 9.4 as follows:

$$
\begin{aligned}
\hat{f}_{\epsilon_6}(w') &= \hat{f}_{\epsilon_6}(w) + \int_w^{w'} \langle \nabla_w \hat{f}_{\epsilon_6}(\omega), d\omega \rangle \\
&= \hat{f}_{\epsilon_6}(w) + \int_w^{w'} \langle \nabla_w \hat{f}_{\epsilon_6}(\omega) - \nabla_w \tilde{f}(\omega) + \nabla_w \tilde{f}(\omega), d\omega \rangle \\
&\leq \hat{f}_{\epsilon_6}(w) + \int_w^{w'} \|\nabla_w \hat{f}_{\epsilon_6}(\omega) - \nabla_w \tilde{f}(\omega)\| \|d\omega\| + \tilde{f}(w') - \tilde{f}(w) \\
&\leq \hat{f}_{\epsilon_6}(w) + \mathcal{O}(\epsilon_6^{1-3\eta} \log(\epsilon_6)) \|w' - w\| - c\alpha \|\nabla_w \tilde{f}\| \epsilon_g \\
&= \hat{f}_{\epsilon_6}(w) + \alpha \|\nabla_w \tilde{f}\| \left[\mathcal{O}(\epsilon_6^{1-3\eta} \log(\epsilon_6)) - c\epsilon_g \right],
\end{aligned}
$$

where we can choose fixed, sufficiently small ϵ_6 so that:

$$\hat{f}_{\epsilon_6}(w') \leq \hat{f}_{\epsilon_6}(w).$$

Moreover, by Lemma 9.2, $\hat{f}$ is invariant to subdivision after finitely many calls to Algorithm 6. We conclude that, after finitely many iterations, $\hat{f}_{\epsilon_6}$ monotonically decreases, contradicting Lemma 9.6. ∎

Proof of proposition 4.3: The outer loop of Algorithm 8 has finite termination by combining Lemma 9.8 and Lemma 9.9. ∎