Robust & Asymptotically Locally Optimal UAV-Trajectory Generation
Based on Spline Subdivision
Ruiqi Ni1, Teseo Schneider2, Daniele Panozzo2, Zherong Pan3, Xifeng Gao1

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 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.

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 [39], [24], [28].

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 [39], [34], [30] 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 [28]. Relaxation: In [6], [3], 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* [35] and its variants can find globally optimal UAV-trajectories in a discrete space and stochastic gradient descend [13] will approach locally optimal solutions only if the descent 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 [19] 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. points, 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).

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

Our method contributes 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. 3) Our formulation achieves simultaneous optimality in trajectory shapes and time allocations.

II. RELATED WORK

Trajectory optimization for a general robotic system can be solved using dynamic programming [29], direct collocation [33], and path-integral control [13], [37]. 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 [20], 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 [36], [16], mixed-integer programming [3], discrete search [18], and fast-marching method [9]. 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 restrict 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 [10], homotopy class [14], and coverage [11], [5]. These constraints can be formulated into the mixed-integer programming via additional binary decision variables [3] or into discrete search algorithms such as A* by pruning trajectories that violate the constraints [18]. 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 [35], [18]. In [9], [31], [34], 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 [38], [8], 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 [27], [9] 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 [25]. The trajectories computed using numerical integration by [25] are locally optimal in either shape variables or time variables but not both at the same time. The latest works [30], [31] achieve shapetime joint optimality via bilevel optimization or weighted combination. We following [31] 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:

$$O(W, T) = \int_0^T c(p(t, W), t)dt + R(p(t, W), T),$$

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:

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

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 $E = (P, \mathcal{L}, T)$ where $P$ is a set of points, $\mathcal{L}$ is a set of line segments, and $T$ is a set of triangles. The distance between a point $x$ and the environment is denoted as $\text{dist}(x, E)$. 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:

$$\arg\min_{W, T} O(W, T) \quad \text{s.t.} \quad \forall t \in [0, T], \begin{cases} \|\dot{p}(t, W)\| \leq v_{max} \\ \|\ddot{p}(t, W)\| \leq a_{max} \\ \text{dist}(p(t, W), E) \geq d_0 \end{cases}$$

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 [9], [12].

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 the different Bézier pieces is achieved
by constraining the control points. The constraints are linear
and can thus be removed from the system based on Störck’s
construction [26].

IV. SUBDIVISION-BASED P-IPM

We propose an iterative algorithm, inspired by the recently
proposed incremental potential contact handling scheme [15],
to find locally optimal solutions of Equation 3 with guaran-
teed feasibility.

![Fig. 1: (a): Our method imposes distance constraints between the convex hull of control points and the obstacle. (b): Our constraints can be refined by subdivision.](image)

A. Method Overview

The key idea of [15] is to use a primal-only (in-
stead of primal-dual) interior point method to solve
collision-constrained nonlinear optimization problems. This
is achieved by transforming the collision-constraints into log-
barrier functions, and reducing the problem to non-linear, un-
constrained optimization. Although primal-dual approaches
have better numerical stability in many problems, their con-
vergence 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 rejecting iterations with a jump in homotopy class
or violation of collision-free constraints. It has been shown
in [15] 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 \( \text{dist}(\bullet, \bullet) \). Although SQP algorithms with gradient-
sampling [2] can handle piecewise-smooth constraints such as
\( \text{dist}(p(t, W), \mathcal{E}) \geq d_0 \), they degrade the deterministic
convergence guarantee to a probabilistic one in the sampling
limit. In [15], [12], the authors note that two triangle meshes
\( \mathcal{E} \) and \( \mathcal{E}' \) are \( d_0 \)-apart if and only if any line segment pair
in \( \mathcal{L} \mathcal{L}' \) \( \equiv \mathcal{L} \times \mathcal{L}' \) and point-triangle pair in \( \mathcal{P} \mathcal{T}' \) \( \equiv \mathcal{P} \times \mathcal{T}' \)
or \( \mathcal{P}' \mathcal{T} \) \( \equiv \mathcal{P}' \times \mathcal{T} \) are \( d_0 \)-apart. As a result, the log-barrier
of \( \text{dist}(\bullet, \bullet) \) 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 search. With a fine-enough
discretization granularity of signed distance field in [9],
or that of sampled waypoints in [18], 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 [1], [32], 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 \( A_i W \), 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 [7]: we denote control points of the first piece
as \( D_1 w \) and the control points of the second piece as \( D_2 w \),
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 matri-
ces 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 1, given the environment \(E\) (represented using either a point cloud or a triangle mesh) and \(E'\), we can define our log-barrier function as:

\[
B(W, E) = \sum_{A \in \mathcal{H}} \left( \sum_{l \in E} c\log(\text{dist}(l, l') - d_0) + \sum_{p \in P} c\log(\text{dist}(p, p') - d_0) + \sum_{t \in T, p \in P} c\log(\text{dist}(t, p') - d_0) \right)
\]

where \(\text{dist}\) is the mollified, differentiable distance between a segment-segment or line-triangle pair, and \(c\log\) is the clamped log-barrier function:

\[
c\log(x) = \left\{ \begin{array}{ll} \frac{(x-x_0)^2}{x_0} - \ln(x_0) & 0 < x \leq x_0 \\ 0 & \text{else} \end{array} \right.
\]

with \(x_0\) being the activation range. Note that we use a slightly modified version of \(c\log\) from the original paper [15]. Our modification preserves the second-order continuity slightly modified version of \(c\log\) from the original paper with.

where \(\lambda\) 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 \(\text{SPD}(\bullet)\) 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:

\[
O(W', T) + \lambda B(W', E) \leq O(W, T) + \lambda B(W, E) + \frac{c\alpha \nabla W \left[ O(W, T) + \lambda B(W, E) \right]^T d}{\parallel \nabla W \parallel^2 d_0 + \epsilon_0}
\]

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 \(E\). This implies we can bound the maximal distance between any curve’s point and \(E\) 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 \(\epsilon\)-optimality is to always subdivide when 1) \(\Delta(w) > \epsilon\) and 2) the distance between the convex hull of \(w\) and \(E\) is smaller than \(d_0 + x_0\), where \(\epsilon\) 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 \(E\) are not subdivided to save computation. Putting these components together, we show in our extended report [23] that, under mild assumptions, P-IPM will converge to the local optima of the original semi-infinite oracle (Equation 3) as the coefficients of log-barrier functions (\(\lambda\)) and the clamp range of \(c\log\) functions (\(x_0\)) tends to zero.

### Algorithm 1: NeedSub(\(w\))

1. Return dist(Hull(\(w\)), \(E\)) < \(d_0 + x_0\) ∧ diameter(Hull(\(w\))) > \(\epsilon\)

### Algorithm 2: Sub(\(w, A, H\))

1. \(< P', L', T' > == \emptyset, \emptyset, \emptyset >\)
2. if NeedSub(\(w\)) then
3. \(H \leftarrow H \setminus \{A\} \cup \{D_1 A\} \cup \{D_2 A\}\)
4. \(< P', L', T' > == \emptyset, P(w), L(w), T(w) >\)
5. else
6. \(\{P', L', T' > == \emptyset, \emptyset, \emptyset >\)
7. Return \(< P', L', T' >\)

### Algorithm 3: TrajSub(\(p(t, W), H\))

1. \(< P', L', T' > == \emptyset, \emptyset, \emptyset >\)
2. for \(A \in H\) do
3. \(< P', L', T' > == \emptyset, P', L', T' > \cup \text{Sub}(p(t, W), A, H)\)
4. Return \(< P', L', T' >\)

### Algorithm 5: Search(\(< W, T >, d)\)

Input: \(\gamma, \epsilon_1 \in (0, 1)\)

1. \(\text{while True do}\)
2. \(\alpha \leftarrow 1, < W', T' > == < W, T > + \alpha d\)
3. Safe\leftarrow True
4. for \(A \in H\) do
5. if dist(Hull(\(AW\) ∪ \(AW'\)), \(E\)) < \(d_0\) then
6. Safe\leftarrow False
7. if Safe ∧ Wolfe’s first condition then
8. Return \(< W', T' >\)
9. else
10. \(\alpha \leftarrow \gamma \alpha, < W', T' > == < W, T > + \alpha d\)

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:

\[
\arg\min_W O(W, T) + \lambda B(W, E),
\]

\[\text{Equation 5}\]
Fig. 2: 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).

D. Inexact P-IPM

A practical problem with Algorithm 4 is that we have to add all the edges in \( L(w) \) and all triangles in \( T(w) \) of each convex hull of the control polygon to Equation 4. Even with a bounding volume hierarchy acceleration, summing up so many terms is still time-consuming. To overcome this issue, we propose an inexact, yet preserving the guarantees of local optimality, version of P-IPM. For every Bézier piece, we only keep all points and one edge. As compared with the exact counterpart, the inexact version significantly reduces the cost of log-barrier function evaluation by a factor of \( O(M^3) \).

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 \( \alpha \) satisfying the first Wolfe’s condition due to the conservative collision check. To ensure convergence to a local minimum, a simply strategy is to keep subdividing whenever the line-search fails. In our extended report [23], we show that, after several slight modifications to the line search Algorithm 5, the inexact P-IPM is guaranteed to converged to the KKT point after a finite number of subdivisions, assuming exact arithmetic.

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 [30], [9] 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 [30], [9], we argue that time-optimal will not be sacrificed because our formulation allows a larger solution space for the relative length of the \( N \) Bézier curves. In prior work [9] 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 the relative length to change arbitrarily. As illustrated in Figure 2, 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 \( S_1w \) and \( S_2w \), 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 \( < P(S_1w), L(S_1w), T(S_1w) > \) for velocity and \( < P(S_2w), L(S_2w), T(S_2w) > \) for acceleration. We can now define our log-barrier function approximating the velocity and acceleration limits as:

\[
B_T(W, T) = \sum_{i=1}^{N} \left[ \sum_{p_1 \in F(S_1w)} \log(v_{\text{max}} - \|p_1\|_2/(T/N)) + \sum_{p_2 \in F(S_2w)} \log(a_{\text{max}} - \|p_2\|_2/(T^2/N^2)) \right] \mid_{w, A, W}.
\]

P-IPM now minimizes \( O(W, T) + \lambda B(W, \varepsilon) + \lambda B_T(W, T) \) 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 \( S_1w \) is bounded by \( T/v_{\text{max}}/N \) and Bézier curve with control points \( S_2w \) is bounded by \( T^2a_{\text{max}}/N^2 \) 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 \( \Delta(S_1w) > \varepsilon \) or \( \Delta(S_2w) > \varepsilon \). 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: \( \lambda \) of the collision avoidance barrier term, \( x_0 \) of the activation range, \( d_0 \) of the clearance distance, \( v_{\text{max}}, a_{\text{max}}, \varepsilon \) of the subdivision threshold and \( \varepsilon_g \) of the stopping criterion. We use \( \lambda = 10, x_0 = 0.1, d_0 = 0.1, v_{\text{max}} = 2.0 m/s, a_{\text{max}} = 2.0 m/s^2, \varepsilon = 0.1, \varepsilon_g = 10^{-3} \) 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 [22] for convex hull collision detection.

A. Comparisons

We compare our approach with the state-of-the-art gradient based method [8] and corridor based method [9] on a set of scenes that are publicly downloadable from SketchFab [4]. 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), \( v_{\text{max}}, a_{\text{max}} \). The implementation of [9] and [8] are run on dense point clouds sampled 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. [9], [8], and ours.

As shown in Table I, our trajectories have the highest quality and far outperform the results of [9], [8]. Visual results are shown in Figure 3. Note that, since [8] doesn’t
Fig. 3: For the scenes listed in Table I, we compare UAV trajectories computed using our method (red), [8] (blue), and [9] (green) from the same initial guess. Both our method and [9] 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 [8] penetrates most of the environments.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Size</th>
<th>$L$</th>
<th>$T$</th>
<th>$C$</th>
<th>$L$</th>
<th>$T$</th>
<th>$C$</th>
<th>$L$</th>
<th>$T$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4M/2.7M</td>
<td>70.5</td>
<td>144.3</td>
<td>168.5</td>
<td>82.6</td>
<td>73.3</td>
<td>0.5</td>
<td>56.2</td>
<td>30.4</td>
<td>218.9</td>
</tr>
<tr>
<td>2</td>
<td>31K/58K</td>
<td>25.7</td>
<td>29.1</td>
<td>8.3</td>
<td>21.5</td>
<td>26.9</td>
<td>0.2</td>
<td>16.1</td>
<td>9.6</td>
<td>2.9</td>
</tr>
<tr>
<td>3</td>
<td>36K/68K</td>
<td>18.7</td>
<td>24.1</td>
<td>2.5</td>
<td>20.1</td>
<td>19.0</td>
<td>0.6</td>
<td>16.7</td>
<td>9.8</td>
<td>2.3</td>
</tr>
<tr>
<td>4</td>
<td>17K/34K</td>
<td>25.1</td>
<td>24.1</td>
<td>4.0</td>
<td>22.8</td>
<td>23.6</td>
<td>0.1</td>
<td>19.8</td>
<td>11.3</td>
<td>9.0</td>
</tr>
<tr>
<td>5</td>
<td>0.4M/0.7M</td>
<td>49.8</td>
<td>65.0</td>
<td>6.4</td>
<td>36.0</td>
<td>50.0</td>
<td>0.3</td>
<td>38.2</td>
<td>21.3</td>
<td>42.1</td>
</tr>
<tr>
<td>6</td>
<td>0.2M/0.3M</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>14.8</td>
<td>18.0</td>
<td>0.2</td>
<td>13.8</td>
<td>9.2</td>
<td>14.9</td>
</tr>
<tr>
<td>7</td>
<td>1K/16K</td>
<td>23.2</td>
<td>38.5</td>
<td>3.8</td>
<td>30.9</td>
<td>26.3</td>
<td>0.2</td>
<td>21.6</td>
<td>12.4</td>
<td>6.2</td>
</tr>
<tr>
<td>8</td>
<td>36K/68K</td>
<td>24.5</td>
<td>23.5</td>
<td>3.3</td>
<td>28.1</td>
<td>26.1</td>
<td>0.1</td>
<td>18.5</td>
<td>10.7</td>
<td>2.2</td>
</tr>
</tbody>
</table>

TABLE I: We profile the performance of [9], [8], and our method (denoted as *) 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 $\text{#V}/\text{#F}$.

- denotes a failure in the provided program. Red color is used to highlight trajectories where the UAV collides with the scene.

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 [9] requires only waypoints as input, we have to modify, in a trial-and-error 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.

Note that [8], [9] achieve real-time computations and good quality trajectories for simple environments, however they struggle as the environments get more complex. [8] tends to generate invalid trajectories with collisions between the UAV and the environment. [9] 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 4), we simulate the UAV flight trajectory in the challenging environment of a museum with a lot of narrow passages. The simulator is setup using [21] 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.

Fig. 4: 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 ($x/\text{dot}/y/\text{dot}/z/\text{dot}$[m/s]) of UAV for both the simulated and our trajectories.

VI. CONCLUSION & LIMITATIONS

We propose a new approach of local UAV-trajectory optimization with guaranteed feasibility and asymptotic convergence to the semi-infinite problem. The key to our success is the primal-only optimizer equipped with an adaptive subdivision scheme for line search and constraint refinement. The line search scheme preserves the homotopy class and the constraint refinement guarantees sufficient closedness to the semi-infinite oracle (Equation 3). Using the same framework, we show that our method can be extended to achieve time-optimality. Besides, our method requires a feasible initial guess, which can be easily computed using global search techniques.
REFERENCES


