Continuous Trajectory Optimization in Non-convex Environments: A Local ADMM Solver for Graphs of Convex Sets

Lukas Pries[1], Jon Arrizabalaga[2], Zachary Manchester[2] and Markus Ryll[1]

***Abstract*— Computing dynamically feasible trajectories in non-convex environments requires reasoning over both continuous motion and discrete spatial structure. In this work, we present a novel numerical solver that couples these two aspects through a customized implementation of the Alternating Direction Method of Multipliers (ADMM). Trajectories are parameterized as polynomials, allowing the primal update to be solved efficiently as a quadratic minimum-control-effort problem. To handle the non-convex geometry of the environment, we introduce a spatio-temporal allocation graph that encodes the assignment of trajectory segments to convex regions. The corresponding slack update reduces to a shortest-path search in this graph, allowing the solver to efficiently explore discrete options while maintaining continuous trajectory consistency within a unified iterative framework. We demonstrate the effectiveness of the approach on several challenging planning scenarios and discuss its potential for scalable trajectory optimization in robotics.**

I. INTRODUCTION

Simultaneous reasoning over discrete and continuous variables arises in many scientific and engineering domains, including task and motion planning, hybrid system control, and constrained decision-making. In these settings, discrete choices—such as selecting topological routes or contact modes—are tightly coupled with continuous variables like system states and control inputs. Consequently, efficient algorithms capable of jointly addressing this discrete-continuous coupling could have a profound impact on many robotic applications.

Historically, this problem has been considered computationally prohibitive, leading to a common decoupling of planning and control. In this paradigm, sampling- or search-based methods first determine a discrete plan, after which gradient-based optimization refines the continuous trajectory. While computationally convenient, this separation partitions the search space and prevents the optimization from reasoning about the global structure of the problem, often leading to suboptimal solutions.

To address this challenge, this paper introduces a novel numerical solver that leverages a customized Alternating Direction Method of Multipliers (ADMM) for continuous trajectory optimization over a union of convex sets. This allows us to decompose the joint optimization into primal, slack, and dual updates, each tailored to exploit the specific mathematical structure of the problem.

[1]Lukas Pries and Markus Ryll are with the Autonomous Aerial Systems Lab, Dep. of Aerospace and Geodesy, TU Munich, Germany, lukas.pries@tum.de, markus.ryll@tum.de

[2]Jon Arrizabalaga and Zachary Manchester are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA jonarri@mit.edu, zacm@mit.edu

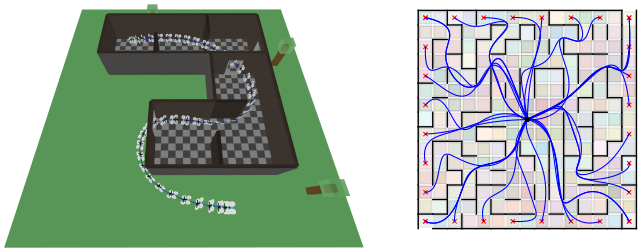


Fig. 1: Example trajectories generated by ACTOR: a quadrotor minimum-snap trajectory (left) and high-order continuous trajectories through a non-convex maze (right).

In our solver, which we denote as **ACTOR** (**A**DMM-based **C**ontinuous **T**rajectory **O**ptimize**R**), we introduce the concept of a *spatio-temporal allocation graph* based on a mixed-integer formulation of the non-convex safety constraints. The graph encodes all feasible convex realizations of the problem and provides the structure required for projection onto the non-convex feasible set, thereby enabling an ADMM-based solver. We frame the slack update as a *Shortest-Path Problem* (SPP) over the weighted allocation graph which can be carried out very efficiently due to its directed acyclic structure.

II. RELATED WORK

Trajectory optimization in non-convex environments is often formulated as a nonlinear program where collision avoidance constraints are expressed using signed-distance functions [1], [2]. While these methods can be solved with general gradient-based solvers [3], [4], they are susceptible to infeasible local minima and typically require good initializations. In practice, many approaches decouple the discrete path selection from continuous trajectory optimization by first computing a corridor using shortest-path heuristics and then optimizing the trajectory within the selected safe corridor [5], [6]. Alternative mixed-integer formulations represent the obstacle-free space as a general union of convex regions and encode region assignments using mixed-integer constraints, enabling joint reasoning over path and trajectory but introducing combinatorial complexity [7], [8]. Recent work on Graphs of Convex Sets (GCS) addresses this challenge through a reformulation into a Shortest-Path Problem (SPP) that allows efficient global optimization over the discrete structure of the environment [9]. However, a key limitation is that GCS generally permits only (convex) costs and bounds on velocity, while excluding any terms on higher-order derivatives that are critical in many applications [10]. In contrast, our method takes a direct approach to the mixed discrete–continuous formulation including higher-order smoothness cost and derivative bounds via an iterative ADMM scheme.

III. PROBLEM STATEMENT

In this section we introduce the problem formulation and the underlying assumptions that we make in order to solve the problem. We consider the following trajectory generation problem:

$$\begin{aligned} \min_{q} \quad & J(q) && \text{(efficiency)} && \text{(1a)} \\ \text{subject to} \quad & q(0) = q_0,\ q(T) = q_T && \text{(boundary)}, && \text{(1b)} \\ & q \in \mathcal{D} && \text{(feasibility)}, && \text{(1c)} \\ & q(t) \in \mathcal{S} && \text{(safety)}, && \text{(1d)} \end{aligned}$$

where $q : [0, T] \to \mathbb{R}^d$ represents a continuous, sufficiently differentiable trajectory.

a) Objective – Minimum Control Effort: We consider smooth trajectories that minimize the control effort required for the robot to follow these trajectories. The objective is therefore a weighted sum of integral quadratic penalties on the derivatives of the trajectory.

$$J(q) = \sum_{i=1}^{n} \alpha_i \int_0^T ||q^{(i)}(t)||^2 dt \tag{2}$$

b) Dynamic Feasibility – Continuous Trajectories: We consider differentially flat systems, including any controllable linear system [11] and nonlinear systems that are locally linearizable by dynamic feedback [12]. For these systems, any sufficiently differentiable trajectory of the flat outputs is dynamically feasible, provided their derivatives are sufficiently bounded to avoid input saturation. Accordingly, we only require $q : [0, T] \mapsto \mathbb{R}^d$ to be continuously differentiable and bounded up to order n_c:

$$\mathcal{D} := \left\{ q(t) \mid q \in \mathcal{C}^{n_c},\ q^{[n_c]}(t) \in \mathcal{B},\ \forall t \in [0, T] \right\}, \tag{3}$$

with $q^{[n_c]} = \left(\dot{q}, \ddot{q}, \ldots, q^{(n_c)}\right)^\top$ and the bounded set $\mathcal{B}$.

c) Safety – Union of Convex Sets: Without loss of generality, we represent the non-convex search space through a union of convex sets as

$$\mathcal{S} := \left\{ q(t) \mid q(t) \in \bigcup\nolimits_{k \in \mathcal{K}} \mathcal{Q}_k,\ \forall t \in [0, T] \right\}, \tag{4}$$

where $\mathcal{K}$ is a finite index set and each $\mathcal{Q}_k$ denotes a convex subset of the free space. We only require $\mathcal{Q}_k$ to be simple in the sense that Euclidean projections onto $\mathcal{Q}_k$ can be performed efficiently (e.g., ball sets, box sets, or polyhedra in low-dimensional spaces).

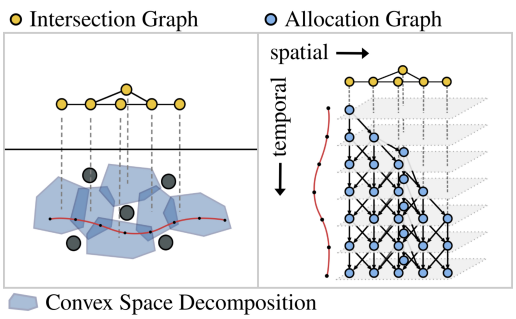


Fig. 2: The allocation graph assigns segments (red) to convex sets (blue) and is constructed from the convex space decomposition and its corresponding intersection graph.

IV. METHOD

The solver presented in this work relies on two main building blocks: (i) a piecewise-polynomial parameterization of the system dynamics and (ii) a graph-based representation of the feasible space through an allocation graph. This section introduces these components and formalizes the required definitions before presenting the solver in the next section.

A. Trajectories as Piecewise Polynomials

We represent the trajectory as a sequence of M polynomial segments with fixed duration T_m. All segments are modeled as n-th order standard Bézier curves whose control points $c \in \mathbb{R}^{M(n+1)\times d}$ serve as decision variables. In the following, we state the resulting expressions under this parameterization, while a detailed derivation is provided in the Appendix. The integral quadratic costs in (2) admit a quadratic form $c^\top \mathbf{Q} c$. The derivatives of a Bézier curve are linear functions of its control points and inter-segment continuity conditions and derivative bounds in (3) can be expressed as linear constraints $\mathbf{A}_{\text{eq}}\, c = \mathbf{b}$ and $\mathbf{A}_{\text{ineq}}\, c \in \mathcal{B}$, respectively. Furthermore, by the convex hull property of Bézier curves, we can enforce that a segment m lies within a safe set $\mathcal{Q}_k$ by requiring $\{c_m^i\}_{i=0}^n \subset \mathcal{Q}_k$. The flexible assignment of segments to sets via a mixed-integer (MI) formulation is introduced next.

B. Mixed-Integer Formulation

Let $\mathcal{M} := \{1, \ldots, M\}$ index the polynomial segments. We encode the assignment of each segment $m \in \mathcal{M}$ to a safe convex set $\mathcal{Q}_k$ using binary integer variables $b_{mk} \in \{0, 1\}$:

$$b_{mk} \implies c_m^i \in \mathcal{Q}_k \quad \forall i \in \{0, \ldots, n\} \tag{5a}$$

$$\sum\nolimits_{k=0}^{K} b_{mk} = 1 \tag{5b}$$

where the linear constraint (5b) ensures that segment m is contained in at least one set $\mathcal{Q}_k$. Note that the implication in (5a) is one-directional and (5b) only makes the assignment to one set explicit but allows a segment to be contained in multiple sets. We further introduce binary variables $y_{m,(k,l)} \in \{0, 1\}$ that indicate a transition of segment m in set $\mathcal{Q}_k$ to segment $m + 1$ in set $\mathcal{Q}_l$. The flow constraints $\sum_{l:(k,l)\in\mathcal{E}_\text{I}} y_{m,(k,l)} = b_{m,k}$ and $\sum_{k:(k,l)\in\mathcal{E}_\text{I}} y_{m,(k,l)} = b_{m+1,l}$ enforce consistency between assignments by restricting transitions to pairs of intersecting sets $(k, l) \in \mathcal{E}_\text{I}$ in the intersection graph G_I, which is defined in the next section.

C. Intersection Graph G_I

Let $\mathcal{S}_\text{I} := \bigcup_{k\in\mathcal{K}} \mathcal{Q}_k$ be the decomposition of the free space into a union of convex sets. We define the intersection graph $G_\text{I} = (\mathcal{V}_\text{I}, \mathcal{E}_\text{I})$ as follows: each convex set $\mathcal{Q}_k$ is associated with a vertex k in $\mathcal{V}_\text{I} = \mathcal{K}$, and edges exist between intersecting sets:

$$\mathcal{E}_\text{I} = \{(k, l) \in \mathcal{K} \times \mathcal{K} \mid \mathcal{Q}_k \cap \mathcal{Q}_l \neq \emptyset\}. \tag{6}$$

D. Allocation Graph G_A

In the following, we show that the previously introduced mixed-integer formulation can be reformulated into a concise graph structure which we denote as allocation graph $G_\text{A} = (\mathcal{V}_\text{A}, \mathcal{E}_\text{A})$. We let each variable b_{mk} represent a vertex (m, k) in $\mathcal{V}_\text{A} := \{(m, k) \mid m \in \mathcal{M},\ k \in \mathcal{K}\}$ and each flow variable

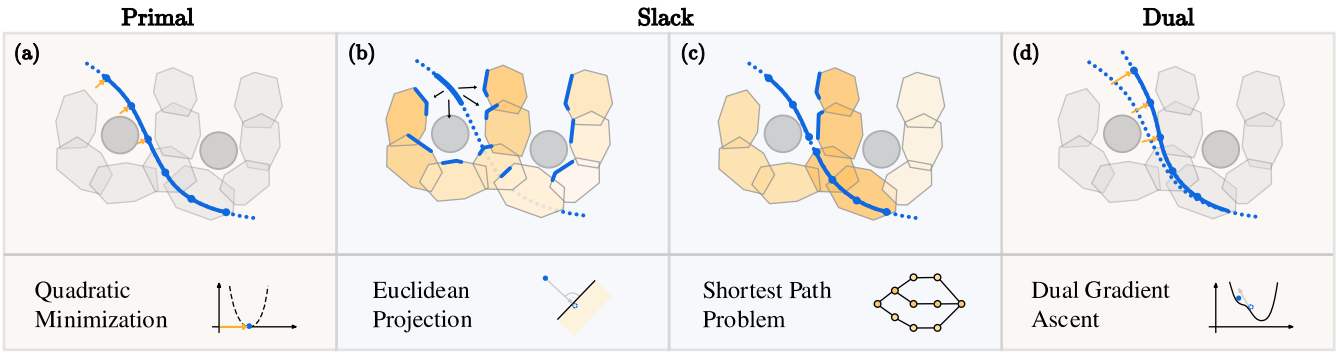


Fig. 3: Effects of ADMM updates on the trajectory. (a) The primal update yields a continuous trajectory. (b) The slack update projects each segment onto all convex sets. (c) A shortest-path search in the weighted allocation graph determines the segment allocation. (d) The dual update enforces consistency between primal and slack via gradient ascent steps.

$y_{m,(k,l)}$ an edge in $\mathcal{E}_\text{A} := \{((m,k),(m+1,l)) \mid (k,l) \in \mathcal{E}_\text{I}\}$. A visual example is given in Fig. 2. The strict temporal ordering of the trajectory segments $0 = t_0 < t_1 < \cdots < t_M = T$ induces a natural orientation of the edges from layer m to layer $m+1$. Thus, the resulting allocation graph is a directed acyclic graph (DAG).

We remark that each path $\mathcal{P} \in G_A$ from layer 1 to M corresponds to a feasible assignment. In particular, for a given path the safe set becomes convex and problem (7) reduces to a convex QP.

E. Problem Description

Combining the polynomial parameterization introduced in Section IV-A with the MI formulation through the allocation graph introduced in Section IV-D yields the following MIQP:

$$\min_{c,\,\mathcal{P}\in G_\text{A}} \quad \frac{1}{2} c^\top \mathbf{Q}\, c \tag{7a}$$

$$\text{s.t.} \quad \mathbf{A}_\text{eq}\, c = \mathbf{b}, \tag{7b}$$

$$\mathbf{A}_\text{ineq}\, c \in \mathcal{B}, \tag{7c}$$

$$c \in \mathcal{S}_\mathcal{P}, \tag{7d}$$

with $\mathcal{S}_\mathcal{P} := \{c \,|\, \{c_m^i\}_{i=0}^n \subset \mathcal{Q}_k,\ \forall (m,k) \in \mathcal{P}\}$.

V. The Actor Solver

In our solver, we employ ADMM to address problem (7) in a systematic way. The key idea is to decompose the problem into subproblems that isolate the difficult set constraint from the smooth quadratic objective. This allows each update step to reduce to a problem class that is well understood and computationally efficient. We present each subproblem in its final form and refer to the appendix for a detailed derivation from the Augmented Lagrangian (AL).

1) ***Primal*** *– Equality-constrained QP:* The primal update solves an equality-constrained QP of the following form:

$$\min_c \quad \frac{1}{2} c^\top \tilde{\mathbf{Q}} c + \tilde{q}^\top c \quad \text{s.t.} \quad \mathbf{A}_\text{eq}\, c = \mathbf{b} \tag{8}$$

where $\tilde{\mathbf{Q}}$ includes the original objective and quadratic penalty terms of the AL. Each iteration updates only the linear cost $\tilde{q}$ involving the Lagrange multipliers λ and slack variables z. The optimal solution c^+ can be obtained through sparse matrix multiplications by solving the linear system arising from the optimality conditions with a precomputed factorization.

2) ***Slack*** *– Projection & Shortest Path Problem:* The slack update takes the form of an Euclidean projection problem

$$\min_{z,\,\mathcal{P}\in G_\text{A}} \|\bar{z} - z\|_2^2 \quad \text{s.t.} \quad z \in \mathcal{X} \tag{9}$$

where $\bar{z}$ denotes the projection point, obtained from the primal solution c^+ via an affine mapping, and $\mathcal{X} := \mathcal{B} \times S_\mathcal{P}$ represents the feasible set corresponding to constraints (7c) and (7d). Intuitively, this step involves a projection of the primal variables onto the feasible set. Since terms are fully separable for both constraints, this step is split into individual problems for each set. While the projection onto the convex set $\mathcal{B}$ is straightforward, the projection onto $\mathcal{S}_\mathcal{P}$ requires further elaboration.

a) Safety Set ($\mathcal{S}_\mathcal{P}$): The separable structure of $\mathcal{S}_\mathcal{P}$ across segments-set assignments (m,k) and control points i admits the decomposition of projection problem (9) into a sum of independent contributions:

$$\min_{\mathcal{P}\in G_A} \sum_{(m,k)\in\mathcal{P}} \sum_{i=0}^{n} \min_{z_m^i \in \mathcal{Q}_k} \|\bar{z}_m^i - z_m^i\|_2^2, \tag{10}$$

which is equivalent to a *Shortest-Path Problem* (SPP) involving the accumulated projection costs for each segment-set pair in a possible path $\mathcal{P} = \{(m,k)\}_{m=1}^M$. As illustrated in Fig. 3b-c, we first compute the projection for each pair and subsequently search for the optimal assignment in the weighted allocation graph. Since the allocation graph is directed and acyclic, the SPP can be solved efficiently through M dynamic programming iterations.

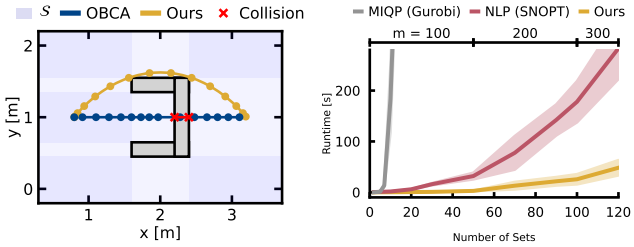
3) ***Dual*** *- Dual Gradient-Ascent Step:* After updating both primal variables c^+ and z^+ that minimize the Augmented Lagrangian, the dual update performs a gradient-ascent step on the Lagrange multipliers:

$$\lambda^+ = \lambda + \rho(c^+ - z^+). \tag{11}$$

This update gradually enforces consensus between variables c and z. Intuitively, it acts as a price update that shifts the primal solution towards feasibility as depicted in Fig. 3d.

A. Convergene and Completeness

The method iterates between primal, slack, and dual updates until a prescribed convergence tolerance is achieved. Unlike the convex case, there is no general guarantee that



(a) Infeasible Local Minimum. (b) Runtime Benchmark.

Fig. 4: Qualitative and quantitative results highlighting the benefits of ACTOR.

ADMM will converge to a globally optimal point for non-convex set constraints and it must be considered a local optimization method [13]. However, our method encodes all *feasible*[1] solutions of the MIQP (7) in the allocation graph and searches for the locally optimal one in each iteration. Thus, if the path $\mathcal{P}$ is frozen at any time, we recover the convex setting, and primal and dual residuals are guaranteed to converge to a feasible solution [14]. In practice, we observe that this happens naturally after a few iterations and does not have to be enforced explicitly (cf. Fig. 5).

VI. EXPERIMENTS

We demonstrate the effectiveness of ACTOR on a variety of numerical examples. Sections VI-A–VI-C provide qualitative comparisons that highlight the key advantages of ACTOR over existing approaches for Problem (1). For a comparison with GCS by Marcucci et al. [9], which considers a related problem setting, we refer to the Appendix VII-C. A runtime benchmark with respect to the problem size is presented in Fig. 4b.

A. Comparison with Direct Trajectory Optimization

Non-convex space constraints pose a fundamental challenge in trajectory optimization, as they induce infeasible local minima and multiple homotopy classes. To demonstrate the conceptual benefits of our method, we consider a minimal illustrative scenario with a non-convex U-shaped obstacle positioned between start and goal and a naive straight line initialization as depicted in Fig. 4a. We compare our method to Optimization-Based Collision Avoidance (OBCA) [15], a general approach for transcribing problem (1) into a non-linear program (NLP). A general NLP solver follows local descent directions and converges to an infeasible stationary point from which it cannot escape, failing to produce a feasible trajectory. In ACTOR, each gradient step is coupled with the allocation graph, which encodes global connectivity

[1]feasibility here only refers to the non-convex constraint $\mathcal{S}$

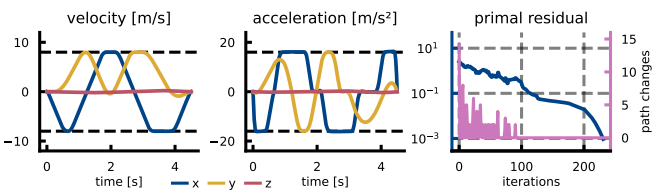


Fig. 5: Bounded velocity and acceleration profile (left) and convergence and vertex changes in the path between ADMM iterations (right) for the min-snap trajectory in Fig. 1.

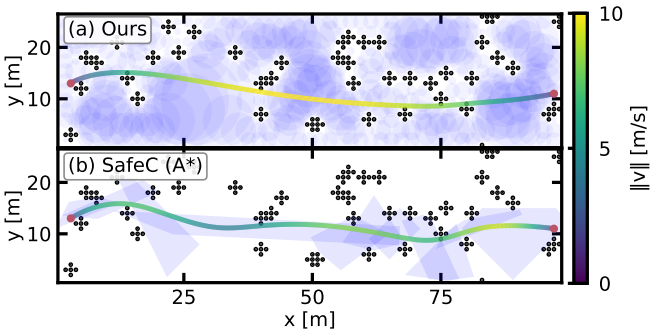


Fig. 6: Continuous trajectories in a point-cloud environment.

of the free space and systematically guides the iterates toward a feasible homotopy class. Importantly, this behavior extends beyond minimal examples: in the highly non-convex maze shown in Fig. 1, ACTOR reliably converges to feasible and locally optimal trajectories despite the presence of numerous competing homotopy classes.

B. Comparison with Decoupled Approaches

We adopt a point-cloud representation of the obstacles to highlight a key practical benefit of the proposed union-of-convex-sets formulation: free space can be decomposed in a simple and naive manner. Unlike decoupled methods, whose performance depends critically on carefully constructed free-space decompositions around a prescribed initial path, ACTOR can operate effectively on such naive decompositions. As shown in Fig. 6, the proposed graph formulation enables a much larger admissible solution space, allowing ACTOR to recover a substantially smoother trajectory than a corridor-based approach (SafeC), whose solution remains restricted to the pre-selected safe corridor.

C. Quadrotor Minimum-Snap Trajectory Planning

Finally, we showcase the full strength of ACTOR on a challenging quadrotor minimum-snap planning problem that combines non-convex spatial constraints with higher-order dynamic feasibility constraints on velocity and acceleration. As shown in Fig. 1 and Fig. 5, our method is able to find a smooth and feasible trajectory in such a highly constrained setting. In fact, we observe reliable convergence from a naive initial guess to a feasible optimum in these settings as depicted in Fig. 5. This experiment highlights the main advantage of ACTOR: jointly optimizing over continuous dynamics and discrete geometry, avoiding the restrictive decoupling of pipeline-based approaches.

VII. CONCLUSION

In this work, we presented ACTOR, a solver for continuous trajectory optimization in non-convex environments represented as a union of convex sets. The key contribution is a spatio-temporal allocation graph that enables the joint treatment of smooth system dynamics and discrete geometric constraints within a unified ADMM-based optimization framework. The resulting solver is robust to naive initialization, factorization-free, and exhibits linear scaling with respect to problem size. Experimental results demonstrate its effectiveness, highlighting the potential of this approach as a scalable framework for motion planning and trajectory optimization in increasingly complex environments.

APPENDIX

A. Polynomial Trajectory Parameterization

To solve problem (1) numerically, the trajectory $q(t)$ must be parameterized using a finite number of decision variables. Due to their smoothness and differentiability properties, polynomials provide a natural choice for this parameterization. Moreover, the feasibility and safety constraints defined in (3) and (4) can be enforced through linear relationships in the polynomial coefficients.

A single polynomial is generally insufficient to represent an entire trajectory. To increase flexibility, the trajectory is divided into consecutive segments, each parameterized by an individual polynomial. A general M-segment polynomial trajectory $p : [0, T] \mapsto \mathbb{R}^d$ with $T = \sum_{m=1}^{M} T_i$ is defined as

$$p(t) = \begin{cases} \sum_{i=0}^{n} c_1^i \, \beta_n^i(t/T_1), & t \in [0, T_1], \\ \sum_{i=0}^{n} c_2^i \, \beta_n^i(t/T_2), & t \in [0, T_2], \\ \quad \vdots & \quad \vdots \\ \sum_{i=0}^{n} c_M^i \, \beta_n^i(t/T_M), & t \in [0, T_M]. \end{cases} \tag{12}$$

where $\beta_n^i(\tau) = \binom{n}{i} \tau^i (1-\tau)^{n-i}$ denotes the Bernstein basis of order n and t is scaled by the duration T_m to yield a standard Bézier curve with $\tau \in [0, 1]$. The i^{th} control point of the m^{th} segment of the Bézier curve is parameterized by $c_m^i \in \mathbb{R}^d$.

a) Derivatives: For a Bézier curve $p_m(t) = \sum_{i=0}^{n} c_m^i \, \beta_n^i(t/T_m)$, its derivative can be expressed by a linear combination of corresponding lower-order control points:

$$\dot{p}_m(t) = \frac{n}{T_m} \sum_{i=0}^{n-1} \left(c_m^{i+1} - c_m^i \right) \beta_{n-1}^i(t/T_m), \tag{13}$$

for $t \in [0, T_m]$, which itself represents a Bézier curve with n control points ${c_m^i}' = n/T_m \left(c_m^{i+1} - c_m^i \right)$ and basis $\beta_{n-1}^i(\tau)$. Recursive application of this formula determines the control points of higher-order derivatives.

b) Continuity and Boundary Constraints: Continuous differentiability of the trajectory is enforced by imposing continuity constraints on the derivatives at segment boundaries:

$$p_m^{(j)}(T_m) = p_{m+1}^{(j)}(0), \tag{14}$$

for all $j \in \{0, \ldots, n_c\}$ and $m \in \{1, \ldots, M\}$, which relates control points of consecutive segments through linear relationships. For the m^{th} and $(m+1)^{\text{th}}$ segment, we present the expression for derivatives up to 2^{nd} order:

$$\begin{aligned} & c_m^n = c_{m+1}^0, \\ & (c_m^n - c_m^{n-1})/T_m = (c_{m+1}^1 - c_{m+1}^0)/T_{m+1}, \\ & (c_m^n - 2c_m^{n-1} + c_m^{n-2})/T_m^2 = (c_{m+1}^2 - 2c_{m+1}^1 + c_{m+1}^0)/T_{m+1}^2 \end{aligned}$$

The boundary constraints can be expressed in an equivalent way, replacing one side of the equation with the fixed boundary derivatives.

c) Safety Constraints: Bézier curves are widely used in constrained trajectory optimization due to their *convex hull property*, which guarantees that the curve lies entirely within the convex hull of its control points. Consequently, constraining the control points of each segment $\{c_m^i\}_{i=0}^n$ to lie within a convex set k ensures that the entire trajectory segment remains inside that set:

$$p_m(t) \in \mathcal{Q}_k \quad \rightarrow \quad \{c_m^i\}_{i=0}^n \subset \mathcal{Q}_k. \tag{15}$$

Given our safety constraint $p_m(t) \in \bigcup_{k \in \mathcal{K}} \mathcal{Q}_k$ defined over a union of convex sets, we will keep the assignment of segments to sets flexible via mixed-integer variables.

d) Dynamic Feasibility Constraints: The convex hull property can be used equivalently to ensure boundedness of the trajectory derivatives:

$$p_m^{(j)}(t) \in \mathcal{B}^j \quad \rightarrow \quad \underline{b}_j \leq (c_m^i)^{(j)} \leq \bar{b}_j, \tag{16}$$

for all $i \in \{0, \ldots, n-j\}$ and $j \in \{0, \ldots, n\}$, effectively constraining each derivative control point to remain in the bounded set $\mathcal{B}^j := \{x \in \mathbb{R}^d \,|\, \underline{b}_j \leq x \leq \bar{b}_j\}$.

e) Smoothness Objective: The integral of the squared n-th derivative over a segment can be expressed as a quadratic form in the polynomial coefficients:

$$\int_0^{T_m} \left[p_m^{(n)}(t) \right]^2 dt \quad \rightarrow \quad c_m^\top Q(T_m) c_m. \tag{17}$$

with $c_m = [c_m^0, c_m^1, \ldots, c_m^n]^\top$.

f) Time Allocation: For a given time-allocation $\mathbf{T} = [T_1, T_2, \ldots, T_m]$, both continuity and boundary constraints depend linearly on the control points $c = [c_0, c_1, \ldots, c_M]^\top$ and can be expressed in compact form as $\mathbf{A}_{\text{eq}} c = \mathbf{b}$. Likewise, derivative bounds can be written as linear set constraints $\mathbf{A}_{\text{ineq}} \in \mathcal{B}$. The quadratic objective (17) admits a block-diagonal representation $c^\top \mathbf{Q}\, c$, overall yielding a quadratic program (QP). The non-convex safety constraint is addressed in Section IV-B.

B. Detailed ADMM Derivation for ACTOR

The optimization problem in (7) involves a quadratic objective (7a) with affine equality (7b) and affine inequality constraints (7c) but, most importantly, a non-convex set constraint (7d) arising from the union of convex sets.

To enable a decomposition into simpler subproblems via ADMM, we introduce auxiliary (slack) variables that decouple the quadratic objective from the set constraints. Specifically, we rewrite (7) in an equivalent consensus form

$$\min_{c} \quad \frac{1}{2} c^\top \mathbf{Q} c + \mathbb{I}_{\mathbf{A}_{\text{eq}} c = \mathbf{b}}(c) + \mathbb{I}_{\mathcal{S}}(z_{\mathcal{S}}) + \mathbb{I}_{\mathcal{B}}(z_{\mathcal{B}}) \tag{18a}$$

$$\text{s.t.} \quad \mathbf{A}_{\text{ineq}}\, c - z_{\mathcal{B}} = 0 \tag{18b}$$

$$c - z_{\mathcal{S}} = 0. \tag{18c}$$

with slack variables $z = (z_{\mathcal{B}}, z_{\mathcal{S}})$ and consensus constraints (18b), (18c) for both the convex and non-convex set, respectively. Here, $\mathbb{I}_{\mathcal{X}}$ denotes the indicator function of a set $\mathcal{X}$, so that $\mathbb{I}_{\mathcal{X}}(x) = 0$ for $x \in \mathcal{X}$ and $\mathbb{I}_{\mathcal{X}}(x) = \infty$ for $x \notin \mathcal{X}$.

We proceed by formulating the Augmented Lagrangian (AL) of the equality-constrained problem (18) from which the individual ADMM steps can be derived:

$$\begin{aligned}\mathcal{L}_A(c,z,\lambda) = \frac{1}{2}c^T\mathbf{Q}c + \mathbb{I}_{\mathbf{A}_{\text{eq}}c=b}(c) + \mathbb{I}_{\mathcal{S}}(z_{\mathcal{S}}) + \mathbb{I}_{\mathcal{B}}(z_{\mathcal{B}}) \\ + \lambda_{\mathcal{B}}^T(\mathbf{A}_{\text{ineq}}c - z_{\mathcal{B}}) + \frac{\rho_{\mathcal{B}}}{2}||\mathbf{A}_{\text{ineq}}c - z_{\mathcal{B}}||_2^2 \\ + \lambda_{\mathcal{S}}^T(c - z_{\mathcal{S}}) + \frac{\rho_{\mathcal{S}}}{2}||c - z_{\mathcal{S}}||_2^2 \end{aligned} \tag{19}$$

where $(\lambda_{\mathcal{S}}, \rho_{\mathcal{S}})$ and $(\lambda_{\mathcal{B}}, \rho_{\mathcal{B}})$ correspond to the dual variables and penalty parameters of the equality constraints (18c) and (18b), respectively.

The *Method of Multipliers* performs primal-dual iterations on the Lagrangian by first minimizing w.r.t. the primal variables c and z before performing a Dual-Ascent step. If we alternate minimization over c and z, rather than simultaneously minimizing over both, we arrive at the three-step ADMM iteration:

$$\text{primal update}: \quad c^+ = \arg\min_c \mathcal{L}_A(c,z,\lambda), \tag{20a}$$

$$\text{slack update}: \quad z^+ = \arg\min_z \mathcal{L}_A(c^+,z,\lambda), \tag{20b}$$

$$\text{dual update}: \quad \lambda^+ = \lambda + \rho\nabla_\lambda \mathcal{L}_A(c^+,z^+). \tag{20c}$$

These steps can be iterated until a desired convergence tolerance is achieved.

In the following, we detail each subproblem and provide its corresponding solution.

*1) **Primal** – Equality-constrained QP:* The primal update results in an equality-constrained QP of the following form:

$$\min_c \quad \frac{1}{2}c^\top\tilde{\mathbf{Q}}c + \tilde{q}^\top c \quad \text{s.t.} \quad \mathbf{A}_{\text{eq}}\,c = \mathbf{b} \tag{21}$$

with

$$\begin{aligned}\tilde{\mathbf{Q}} &= \mathbf{Q} + \rho_{\mathcal{S}}\mathbf{I} + \rho_{\mathcal{B}}\mathbf{A}_{\text{ineq}}^\top\mathbf{A}_{\text{ineq}} \\ \tilde{q} &= \lambda_{\mathcal{S}} - \rho_{\mathcal{S}}z_{\mathcal{S}} + \mathbf{A}_{\text{ineq}}^\top(\lambda_{\mathcal{B}} - \rho_{\mathcal{B}}z_{\mathcal{B}}).\end{aligned}$$

whose solution can be obtained by solving the linear system arising from the optimality conditions $\mathbf{A}_{\text{eq}}\,c^+ = \mathbf{b}$, $\tilde{\mathbf{Q}}c^+ + \tilde{q} + \mathbf{A}_{\text{eq}}^\top\lambda_{\text{eq}}^+ = 0$. Since both $\tilde{\mathbf{Q}}$ and $\mathbf{A}_{\text{eq}}$ remain unchanged between iterations, the factorization can be performed offline and online evaluation of c^+ only involves sparse matrix multiplications.

*2) **Slack** – Projection & Shortest Path Problem:* The slack update (20b) takes the form of an Euclidean projection problem

$$\min_z ||\bar{z} - z||_2^2 \quad \text{s.t.} \quad z \in \mathcal{X} \tag{22}$$

where $\bar{z}$ denotes the projection point, obtained from the primal solution c^+ via an affine mapping, and $\mathcal{X} := \mathcal{B} \times S_{\mathcal{P}}$ represents the feasible set. Since terms for $z_{\mathcal{B}}$ and $z_{\mathcal{S}}$ are fully separable, this step is split into individual problems for each set.

a) Bounded Set ($\mathcal{B}$): For the convex set $\mathcal{B}$ this follows the standard ADMM formulation and the solution can be obtained in closed form through a simple linear projection $\Pi_{\mathcal{B}}$ onto the feasible set:

$$z_{\mathcal{B}}^+ = \Pi_{\mathcal{B}}(\bar{z}_{\mathcal{B}}) \tag{23}$$

with $\bar{z}_{\mathcal{B}} = \mathbf{A}_{\text{ineq}}c + \frac{1}{\rho_{\mathcal{B}}}\lambda_{\mathcal{B}}$. Evaluating $\Pi_{\mathcal{B}}$ can be performed efficiently by an element-wise min-max operation $\Pi_{\mathcal{B}}^i(z_i) = \min\{\max\{z_i, \underline{b}\}, \bar{b}\}$.

b) Safety Set ($\mathcal{S}$): For the variables $z_{\mathcal{S}}$, the update involves a projection onto a non-convex set which may not be unique. In our setting, we have encoded all combinations of feasible segment-set projections in the spatio-temporal allocation graph G_A that we introduced in (IV-D). In accordance with 20b, we seek the minimizer of

$$\min_{z,\,\mathcal{P}\in G_{\text{A}}} \quad ||\bar{z} - z||_2^2 \tag{24a}$$

$$\text{s.t.} \quad \{z_m^i\}_{i=0}^n \subset \mathcal{Q}_k, \quad \forall (m,k) \in \mathcal{P} \tag{24b}$$

with $\bar{z} = c$. We have omitted the subscript of $z_{\mathcal{S}}$ for clarity. Given a segment-set allocation $\{(m,k)\}_{m=1}^M$ through path $\mathcal{P}$, the objective and constraints in (24) are fully separable for each segment m. Further, for each segment it is also fully separable for each individual control point z_m^i. This admits the reformulation into:

$$\min_{\mathcal{P}\in G_A} \sum_{(m,k)\in\mathcal{P}} \sum_{i=0}^{n} \min_{z_m^i \in \mathcal{Q}_k} ||\bar{z}_m^i - z_m^i||_2^2, \tag{25}$$

which essentially becomes a *Shortest-Path Problem* (SPP) involving the sum over all projection costs for each segment-set pair in a possible path $\mathcal{P} \in G_A$. In Fig. 3, we visualize the cost for both segment-set pairs in (b) and paths in (c).

Since $\mathcal{Q}_k$ is convex, the projection of each control point onto a set k can be computed efficiently as in (23):

$$z_m^{i\,+} = \Pi_{\mathcal{Q}}(\bar{z}_m^i) \tag{26}$$

from which the projection costs for each segment-set pair $z_m^{k\,+} = [z_m^{0\,+}, z_m^{1\,+}, \ldots, z_m^{n\,+}]$ is computed as $l_{mk} = \sum_{i=0}^n ||\bar{z}_m^i - z_m^{i\,+}||_2^2$.

This allows us to simplify problem (25) as follows:

$$\min_{\mathcal{P}\in G_A} \sum_{(m,k)\in\mathcal{P}} l_{mk} \tag{27}$$

yielding a SPP on the weighted allocation graph G_A with vertex cost l_{mk}. Note that we can equivalently assign the vertex cost to all incoming edges of a vertex to represent this problem in a more standard form with costs assigned to the edges of a graph.

c) Shortest Path Problem on DAG: As derived in IV-D, the allocation graph G_A is a directed acyclic graph (DAG) with layers induced through segments m. Thus, the shortest path problem can be solved through M dynamic programming iterations in topological order. The recursion is initialized at the first layer by setting $V_{1k} = l_{1k}$ for $k \in \mathcal{K}_{\text{start}}$, and $V_{1k} = +\infty$ otherwise. For each $m = 2, \ldots, M$:

$$V_{mk} = l_{mk} + \min_{l:(l,k)\in\mathcal{E}_I} V_{m-1,l}, \qquad \forall k \in \mathcal{K}, \tag{28a}$$

$$\pi_{mk} = \arg\min_{l:(l,k)\in\mathcal{E}_I} V_{m-1,l}, \qquad \forall k \in \mathcal{K}, \tag{28b}$$

where V_{mk} is the value function and π_{mk} the minimizing predecessor of each node. After the forward pass, the optimal path can be obtained by simple backtracking from the goal node via $k_{m-1} = \pi_{m,k_m}$.

Finally, the optimal projection is obtained by selecting $z_{\mathcal{S}}^{+} = [z_1^{k+}, z_2^{k+}, \ldots, z_M^{k+}]$ based on the optimal path $\mathcal{P} = \{(m,k)\}_{m=1}^{M}$.

3) ***Dual*** *- Gradient Ascent Step:* After updating both primal variables c and z that minimize the AL, the dual update (20c) performs a gradient-ascent step on the Lagrange multipliers with step-size ρ. This update gradually enforces consensus between variables c and z.

C. Comparison with GCS

We first discuss the key differences between our problem formulation and the Graphs of Convex Sets (GCS) approach of Marcucci et al. [9]. In GCS, each convex set is associated with a single trajectory segment, and both spatial and temporal variables are optimized jointly for each segment. This joint optimization enables flexibility in both domains but introduces nonlinear dependencies between trajectory coefficients and time. In contrast, our approach employs fixed-duration segments and optimizes exclusively over spatial parameters and the discrete allocation to individual sets. Temporal flexibility is achieved implicitly by allowing a variable number of segments to be assigned to each sets.

The underlying graph structures further distinguish the two approaches. GCS directly transcribes problem (1) into a *global* shortest-path problem (SPP) over a graph induced by the convex sets. In contrast, ACTOR encodes mixed-integer variables for the assignment of segments to sets in the graph. The SPP arises as a subproblem within an iterative ADMM scheme to efficiently compute *local* projections onto the non-convex free space.

A fundamental difference lies in the treatment of higher-order derivatives. In GCS, costs and constraints involving acceleration or higher-order derivatives become non-convex due to their nonlinear dependence on time. As a result, GCS is limited to cost that are linear in time and linear or quadratic in the path (e.g. path length or path energy), ensuring compatibility with the convex relaxation of the SPP. In contrast, ACTOR optimizes over spatial control points of fixed-duration segments, for which integral costs and constraints on higher-order derivatives remain convex (cf. Section VII-A). This allows direct optimization of smoothness objectives.

This limitation is particularly relevant in practice, as control inputs for many robotic systems correspond to higher-order derivatives, such as torque commands for robotic manipulators, acceleration commands in autonomous vehicles and thrust commands of a quadrotor that relate to snap derivatives of a trajectory. While GCS introduces surrogate regularization terms to prevent higher-order derivatives from growing excessively in magnitude, these approximations are generally less tight and do not directly minimize smoothness cost.

To illustrate this difference, we consider the quadrotor example provided in the public GCS implementation [16]. Using the cost formulation proposed in [9], we progressively increase the regularization weights on higher-order derivatives until the solver becomes numerically instable. Figure 7 shows the resulting trajectories and their corresponding acceleration profiles for varying regularization levels. While stronger regularization effectively reduces the magnitude of higher-order derivatives, it simultaneously degrades trajectory optimality. In contrast, ACTOR directly minimizes smoothness costs and produces trajectories that are both smoother and exhibit significantly lower derivative magnitudes for the same problem setup.

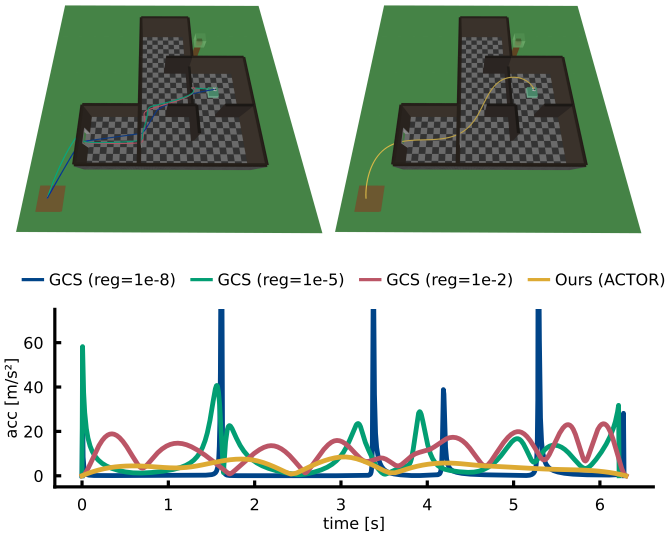


Fig. 7: Top: Quadrotor minimum-snap trajectories generated by GCS (for different regularization weights) and by ACTOR. Bottom: Corresponding acceleration profiles.

References

[1] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*, IEEE, 2009, pp. 489–494.

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

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

[4] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[5] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: The 16th International Symposium ISRR*, Springer, 2016, pp. 649–666.

[6] Y. Ren, F. Zhu, G. Lu, Y. Cai, L. Yin, F. Kong, J. Lin, N. Chen, and F. Zhang, "Safety-assured high-speed navigation for mavs," *Science Robotics*, vol. 10, no. 98, eado6187, 2025.

[7] 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.

[8] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 922–938, 2021.

[9] T. Marcucci, M. Petersen, D. Von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, eadf7843, 2023.

[10] D. von Wrangel and R. Tedrake, "Using graphs of convex sets to guide nonconvex trajectory optimization," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2024, pp. 9863–9870.

[11] P. Brunovskỳ, "A classification of linear controllable systems," *Kybernetika*, vol. 6, no. 3, pp. 173–188, 1970.

[12] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[13] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, "A simple effective heuristic for embedded mixed-integer quadratic programming," *International journal of control*, vol. 93, no. 1, pp. 2–12, 2020.

[14] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[15] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.

[16] T. Marcucci, M. Petersen, D. Von Wrangel, and R. Tedrake, *Motion planning around obstacles with convex optimization*, `https://github.com/RobotLocomotion/gcs-science-robotics`, Accessed: 2026-03-05, 2023.