IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

Robust Multi-Robot Trajectory Optimization Using Alternating Direction Method of Multiplier

Ruiqi Ni[1], Zherong Pan[2], and Xifeng Gao[2]

***Abstract*—We propose a variant of alternating direction method of multiplier (ADMM) to solve constrained trajectory optimization problems. Our ADMM framework breaks a joint optimization into small sub-problems, leading to a low iteration cost and decentralized parameter updates. Starting from a collision-free initial trajectory, our method inherits the theoretical properties of primal interior point method (P-IPM), i.e., guaranteed collision avoidance and homotopy preservation throughout optimization, while being orders of magnitude faster. We have analyzed the convergence and evaluated our method for time-optimal multi-UAV trajectory optimizations and simultaneous goal-reaching of multiple robot arms, where we take into consider kinematics-, dynamics-limits, and homotopy-preserving collision constraints. Our method highlights an order of magnitude's speedup, while generating trajectories of comparable qualities as state-of-the-art P-IPM solver.**

***Index Terms*—ADMM, trajectory optimization, multi-robot and motion planning**

I. Introduction

This paper focuses on trajectory optimization problems, a fundamental topic in robotic motion planning. Although the problem finds countless domains of applications, their pivotal common feature could be illustrated through the lens of two applications: goal-reaching of multiple UAVs and articulated robot arms. UAV trajectory optimization has been studied vastly [1]. Due to their small size and differential-flat dynamics [2], point-mass models can be used and Cartesian-space trajectories are linear functions of configuration variables. Furthermore, the quality of a UAV trajectory could be measured via convex metrics such as jerk or snap, casting trajectory optimization as convex programs. However, when flying in obstacle-rich environments and among other UAVs, non-convex, collision constraints must be considered [3]. Failing to satisfy these constraints can render a generated trajectory completely useless. Handling articulated robot arms poses an even more challenging problem, where the linear dynamic assumption must be replaced with a nonlinear forward kinematic function that maps from configuration- to Cartesian-space, rendering all the Cartesian-space constraints non-convex. In summary, trajectory optimizer should pertain three properties: **(versatility)** handle non-convex constraints and kinematic models; **(robustness)** guarantee to satisfy all the constraints throughout optimization; **(efficacy)** rapidly refine feasible initial trajectories into nearby, locally optimal solutions.

We observe that prior trajectory optimization techniques exhibit remarkable performs under certain assumptions but have partial coverage of the three features above. For example, off-the-shelf primal-dual optimizers can solve general constrained programs and have been applied to trajectory optimization [4, 3, 5]. However, these methods violate robustness by allowing a feasible trajectory to leave the constraint manifold. Similarly, penalty methods [6, 7, 8] have been used for trajectory optimization by replacing hard constraints with soft energies, which cannot guarantee robustness. On the other hand, we proposed a new optimizer in our prior work [9] for UAV trajectory planning with perfect versatility and robustness, where all the constraints are converted into primal-only log-barrier functions with finite duality gap. As a result, all the constraints are satisfied throughout the optimization with the Continuous Collision Detection (CCD) bounded line search step. With the improved robustness, however, comes a significant sacrifice in efficacy. For the same benchmarks, our primal-only methods can take $3-5\times$ more computations to converge as compared with primal-dual counterparts. This is due to the log-barrier functions introducing arbitrarily large gradients near the constraint boundaries. As a result, an optimizer needs to use a costly line-search after each iteration to ensure a safe solution that satisfies all the stiff constraints. The gradient-flows of such objective functions are known as stiff dynamics, for which numerical time-integration can have ill-convergence as studied [10].

Main Results: We propose a variant of ADMM-type solver that inherits the versatility and robustness from [9], while we achieve orders of magnitude higher performance. Intuitively, ADMM separates non-stiff and stiff objective terms into different sub-problems using slack variables, so that each sub-problem is well-conditioned. Moreover, since sub-problems are independent and involve very few decision variables, an ADMM iteration can be trivially parallelized and incurs a much lower cost. Existing convergence analysis, however, only guarantees that ADMM converges for convex problems or non-convex problems with linear or affine constraints [11]. We present improved analysis which shows that our ADMM variant converges for both UAV and articulated trajectory planning problems under nonlinear collision constraints, kinematic- and dynamic-limits. We have applied our method to large-scale multi-UAV trajectory optimization and articulated multi-robot goal-reaching problems as defined in Section III. During our evaluations (Section V), we observe tens of times' speedup over Newton-type methods. Our algorithms are detailed in Section IV.

Manuscript received: November 16, 2021; Revised February 5, 2022; Accepted March 4, 2022.

This paper was recommended for publication by Editor Stephen J. Guy upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by NSF-IIS-1910486.

[1] Department of Computer Science, Florida State University, rn19g@my.fsu.edu. [2] Lightspeed & Quantum Studios, Tencent America, {zherong.pan.usa, gxf.xisha}@gmail.com.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

II. Related Work

We review related trajectory generation techniques and cover necessary backgrounds in operations research.

Trajectory Generation aims at computing robot trajectories from high-level goals and constraints. Their typical scenarios of applications involve navigation [12], multi-UAV coordination [13], human-robot interaction [14], tele-operation [15], trajectory following [16], etc., where frequent trajectory update is a necessity to handle various sources of uncertainty. Due to the limited computational resources, early works use pre-computations to reduce the runtime cost. For example, Panagou, Shimoda *et al.* [12, 17] modulate a vector field to guide agents in a collision-free manner, while assuming point robots and known environments. Closed-form solutions such as [18, 19] exist but are limited to certain types of dynamic systems or problem paradigms. More recently, Belghith *et al.*, Karaman *et al.* [20, 21] have established anytime-variants of sampling-based roadmaps that continually improve an initial feasible solution during execution.

Trajectory Optimization refines robot trajectories given a feasible or infeasible initial guess. Trajectory optimization dates back to [22, 4], but has recently gained significantly attention due to the maturity of nonlinear programming solvers. These methods have robots' goal of navigation formulated as objective functions, while taking various safety requirements as (non)linear constraints. They achieve unprecedented success in real-time control of high-dimensional articulated bodies [23] and large swarms of UAVs [3]. In particular, trajectory optimizer can also be used for trajectory generation by setting the initial trajectory to be a trivial solution [23, 24, 25]. On the down side, most trajectory optimization techniques suffer from a lack of robustness. Many works formulate constraints as soft objective functions [26, 23] or use primal-dual interior point methods to handle non-convex constraints [4, 27, 25], which does not guarantee constraint satisfaction. On the other hand, some techniques [28, 29] restrict the solution space to a disjoint convex subset so that efficient solvers are available, but these methods are limited to returning sub-optimal solutions. Instead, Ni *et al.* [9] starts from a strictly feasible initial guess and uses a primal-only method to transform non-convex constraints into log-barrier functions with finite duality gap. Using a line-search with CCD as safe-guard, it is guaranteed to satisfy all constraints and no restrictions in solution space are needed. But optimizer in this case can make slow progress, being blocked by the large gradient of log-barrier functions. This dilemma between efficacy and robustness has been studied as stiff dynamic systems [10], for which dedicated techniques are developed for different applications such as numerical continuation [30]. Unlike these methods, we show that first-order methods can be combined with barrier methods to achieve significant speedup.

Alternating Direction Method of Multiplier is a first-order optimization framework originally designed in convex-programming paradigm [31]. ADMM features a low iteration cost and moderate accuracy of solutions, making it a stellar fit for trajectory optimization where many iterations can be performed within a short period of time. ADMM uses slack variables to split the problem into small subproblems and approximately maintains the consistency between slack and original variables by updating the Lagrangian multipliers. Although theoretical convergence guarantee was only available under convex assumptions, ADMM has been adapted to solve non-linear fluid dynamics control [32], collision-free UAV trajectory generation [24], and bipedal locomotion [33], where good empirically performances have been observed. It was not until very recently that the good performances of ADMM in nonlinear settings have been theoretically explained by [34, 35, 11]. Unfortunately, even these latest analysis cannot cover many robotic applications such as [24]. Our new ADMM algorithm correctly divides the responsibility between the slack and original variables, where the slack variables handle non-stiff function terms and the original variables handle stiff ones. By using line-search to ensure strict function decrease, we can prove convergence (speed) of our algorithm to a robust solution.

III. Trajectory Optimization Paradigm

We motivate our analysis using a 2D collision-free trajectory optimization problem as illustrated in Figure 1, while our algorithm is applied to both 2D and 3D workspaces alike. Consider a point robot traveling in a 2D workspace along a piecewise linear trajectory discretized using $N+1$ linear segments and $N+2$ vertices. We further assume the start and goal positions are fixed, leaving the intermediary N points as decision variables. This trajectory can be parameterized by a vector $x \in \mathbb{R}^{2N}$ where $x_i \in \mathbb{R}^2$ is the ith vertex. For simplicity in this example, our goal is for the trajectory to be as smooth as possible. Further, the robot must be collision-free and cannot intersect the box-shaped obstacle in the middle and we assume that the four vertices of the box are $Z = \{z_1, z_2, z_3, z_4\}$. These collision constraints can be expressed as:

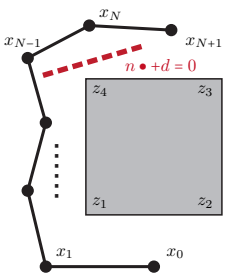

Fig. 1: Illustrative problem with the red dashed line being the separating plane.

$$\mathrm{d}(\mathrm{hull}(X_i), \mathrm{hull}(Z)) \geq 0 \quad \forall i = 1, \cdots, N-1,$$

where $X_i = \{x_i, x_{i+1}\}$ is the ith line segment ($X_i \in \mathbb{R}^4$ in this illustrative toy example), $\mathrm{hull}(\bullet)$ denotes the convex hull and d is Euclidean the distance between two convex objects. A collision-free trajectory optimization problem can be formulated as:

$$\begin{aligned} \underset{x}{\operatorname{argmin}} \ & \sum_i X_i^T L X_i \\ \text{s.t. } & \mathrm{d}(\mathrm{hull}(X_i), \mathrm{hull}(Z)) \geq 0 \quad \forall i = 1, \cdots, N-1, \end{aligned} \tag{1}$$

where L is the Laplacian stencil measuring smoothness and $X_i^T L X_i$ measures the squared length of ith line segment. However, Equation 1 only considers geometric or kinematic constraints and a robot might not be able to traverse the optimized trajectory due to the violation physical constraints. In many problems, including autonomous driving [36] and UAV path planning [37], a simplified physical model can be incorporated that only considers velocity and acceleration

limits. We could approximate the velocity and acceleration using finite-difference as:

$$V_i \triangleq x_{i+1} - x_i \quad A_i \triangleq x_{i+2} - 2x_{i+1} + x_i,$$

and formulate the time-optimal, collision-free trajectory optimization problem as:

$$\begin{aligned} \underset{x,\Delta t}{\operatorname{argmin}} \ & \sum_i X_i^T L X_i / \Delta t^2 + w\Delta t \\ \text{s.t. } & \mathrm{d}(\mathrm{hull}(X_i), \mathrm{hull}(Z)) \geq 0 \quad \forall i = 1, \cdots, N-1 \\ & \|V_i\| \leq v_{\max}\Delta t \quad \forall i = 1, \cdots, N-1 \\ & \|A_i\| \leq a_{\max}\Delta t^2 \quad \forall i = 1, \cdots, N-2, \end{aligned} \tag{2}$$

where we use $\sum_i X_i^T L X_i / \Delta t^2$ to measure trajectory smoothness with time, e.g. Dirichlet energy, and we use a coefficient w to balance between optimality in terms of trajectory length and arrival time. Here, $v, a_{\max}$ are the upper bounds of velocities and accelerations. Although the above example is only considering a single robot and piecewise linear trajectories, extensions to several practical problem settings are straightforward, as discussed below.

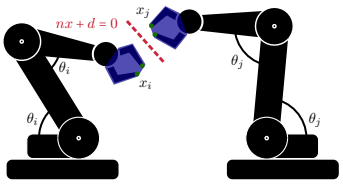

Fig. 2: The configuration of two robot arms includes joint angles θ_i, θ_j. The set of Cartesian points on the robot, x_i, x_j (green), are mapped from θ_i, θ_j using forward kinematic functions. To avoid collisions between the two robot arms, a convex hull is computed for each robot link (blue), and a separating plane (red) is used to ensure the pair of convex hulls stay on different sides.

A. Time-Optimal Multi-UAV Trajectory Optimization

We first extend our formulation to handle multiple UAV trajectories represented using composite Bézier curves with N pieces of order M. In this case, the decision variable x is a set of $N(M-2)+3$ control points or $x \in \mathbb{R}^{3(N(M-2)+3)}$ (note that the two neighboring curves share 3 control points to ensure second order continuity and we refer readers to [9] for more details). For the ith piece of Bézier curve, the velocity and acceleration are defined as:

$$X_i \triangleq \begin{pmatrix} x_{i(M-2)-M+3} \\ \vdots \\ x_{i(M-2)+3} \end{pmatrix} \quad V_i(s) \triangleq \dot{I}(s)X_i \quad A_i(s) \triangleq \ddot{I}(s)X_i,$$

where X_i are the $M+1$ control points of the ith Bézier curve piece, and $I(s)$, $\dot{I}(s)$ and $\ddot{I}(s)$ are the Bézier curve's interpolation stencil for position, velocity and acceleration, respectively, with $s \in [0,1]$ being the natural parameter. It can be shown that $V_i(s), A_i(s)$ are also Bézier curves of orders $M-1$ and $M-2$, respectively. The velocity and acceleration limits must hold for every $s \in [0,1]$, for which a finite-dimensional, conservative approximation is to require all the control points of $\dot{I}(s)$ and $\ddot{I}(s)$ are bounded by $v_{\max}$ and $a_{\max}$. With a slight abuse of notation, we reuse $I, \dot{I}, \ddot{I}$ without parameter s to denote the matrices extracting the control points of $I(s)X_i, \dot{I}(s)X_i, \ddot{I}(s)X_i$, respectively. In other words, we define the vectors $V_i \triangleq \dot{I}X_i$ and $A_i \triangleq \ddot{I}X_i$ as the control points of the ith Bézier curve, then the form of velocity and acceleration limits are identical to Equation 2.

For multiple UAVs, however, we need to consider the additional collision constraints between different trajectories. To further unify the notations, we concatenate the control points of different UAVs into a single vector x, i.e., two Bézier curve pieces might correspond to different UAVs, and we introduce collision constraints between different UAVs:

$$\mathrm{d}(\mathrm{hull}(X_i), \mathrm{hull}(X_j)) \geq 0 \quad \forall i, j \in \text{different UAV}.$$

Our final formulation of time-optimal multi-UAV trajectory optimization takes the following form:

$$\begin{aligned} \underset{x,\Delta t}{\operatorname{argmin}} \ & \sum_i \mathcal{O}(X_i, \Delta t) \\ \text{s.t. } & \mathrm{d}(\mathrm{hull}(X_i), \mathrm{hull}(Z)) \geq 0 \quad \forall i = 1, \cdots, N-1 \\ & \mathrm{d}(\mathrm{hull}(X_i), \mathrm{hull}(X_j)) \geq 0 \quad \forall i, j \in \text{different UAV} \\ & \|V_i\| \leq v_{\max}\Delta t \quad \forall i = 1, \cdots, N-1 \\ & \|A_i\| \leq a_{\max}\Delta t^2 \quad \forall i = 1, \cdots, N-2, \end{aligned} \tag{3}$$

where we generalize the objective function with smoothness and time optimality to take an arbitrary, possibly non-convex form, $\mathcal{O}(X_i, \Delta t)$, which is a function of a single piece of sub-trajectory X_i. Almost all the objective functions in trajectory optimization applications can be written in this form. For example, smoothness can be written as the sum of total curvature, snap, or jerk of each piece, and the end-point cost is only related to the last piece.

B. Goal-Reaching of Articulated Robot Arms

The position of UAV at any instance s on the ith Bézier curve is $I(s)X_i$, which is a linear function of decision variable X_i and thus x. However, more general problem settings require non-linear relationships, of which a typical case is articulated robot arms as illustrated in Figure 2. Consider the problem of multiple interacting robot arms in a shared 3D workspace. Each arm's Cartesian-space configuration at the ith time instance is represented by a triangle mesh with a set of vertices concatenated into the vector x_i. However, we need to maintain the corresponding configuration θ_i, where $|\theta_i|$ is the degrees of freedom of each arm (DOF). Our decision variable is $\theta \in \mathbb{R}^{\mathrm{DOF} \times N}$, each x_i is a derived variable of θ_i via the forward kinematics function $x_i \triangleq \mathrm{FK}(\theta_i)$, and a linear interpolated Cartesian-space trajectory is: $X_i(\theta) = \{x_i, x_{i+1}\} = \{FK(\theta_i), FK(\theta_{i+1})\}$. We further define the velocity and acceleration in configuration space as:

$$V_i(\theta) \triangleq \theta_{i+1} - \theta_i \quad A_i(\theta) \triangleq \theta_{i+2} - 2\theta_{i+1} + \theta_i.$$

In summary, the multi-arm goal-reaching problem can be formulated as:

$$\begin{aligned} \underset{\theta,\Delta t}{\operatorname{argmin}} \ & \sum_i \mathcal{O}(X_i(\theta), \Delta t) \\ \text{s.t. } & \mathrm{d}(\mathrm{hull}(X_i(\theta)), \mathrm{hull}(Z)) \geq 0 \quad \forall i = 1, \cdots, N-1 \\ & \mathrm{d}(\mathrm{hull}(X_i(\theta)), \mathrm{hull}(X_j(\theta))) \geq 0 \quad \forall i, j \in \text{different arm} \\ & \|V_i(\theta)\| \leq v_{\max}\Delta t \quad \forall i = 1, \cdots, N-1 \\ & \|A_i(\theta)\| \leq a_{\max}\Delta t^2 \quad \forall i = 1, \cdots, N-2. \end{aligned} \tag{4}$$

Equation 4 takes a more general form than all the previous problems, and we would propose our variant of ADMM algorithm assuming this formulation. If our method is applied

to UAV trajectory optimization, we can plug in the degenerate relationship $\text{FK}(\theta_i) = \theta_i$.

***Remark** 3.1:* By using separating planes to formulate collision constraints, we assume each robot link is convex. This treatment allows us to use only one separating plane between a pair of robot links. If concave features of robot links must be modeled accurately, then robot link must be further decomposed into convex parts, and a separating plane must be used between each pair of parts, leading to more separating planes and iterations.

IV. ADMM-Type Trajectory Optimization

ADMM is a variant of the Augmented Lagrangian Method (ALM) that does not update the penalty parameter. The main advantage of ADMM is that, by introducing slack variables, each substep consists of either a small problem involving the non-stiff part of the objective function or a large problem involving the stiff part of the objective function or constraints. As a result, the ADMM solver allows larger timestep sizes to be taken for the non-stiff part, leading to faster convergence. Although ADMM has only first-order convergence rate, it can quickly approximate a locally optimal solution with moderate accuracy, which is sufficiently for trajectory optimization.

ADMM handles inequality constraints by reformulating them as indicator functions, but we are handling possibly non-convex constraints for which projection operators, which is associated with indicator functions, do not have closed-form solutions. Instead, we follow our prior work [9] and rely on a log-barrier relaxations with non-zero duality gap. For example, if we have a hard constraint $g(x) \geq 0$ where g is some differentiable function, then the feasible domain can be identified with the finite sublevel set of the log-barrier function: $-\log(g(x))$. We apply this technique to the velocity and acceleration limits. A similar technique can be used for collision constraints with the help of a separating plane, which is illustrated in Figure 1 as $n^T \bullet + d = 0$ (n is the plane normal, d is the position, and $\bullet$ is the arbitrary point on the plane). Since we only consider distance between convex hulls, two convex hulls are non-overlapping if and only if there is a separating plane such that the two hulls are on different sides. We propose to optimize the parameters of the separating plane (n, d) as additional slack variables. As a result, the collision constraints become convex when fixing n, d and optimizing the trajectory alone. Applying this idea to all the constraints and we can transform Equation 4 into the following unconstrained optimization:

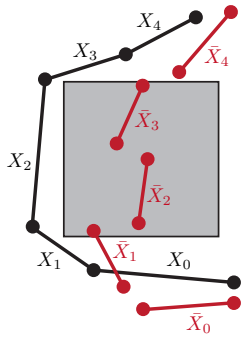

Fig. 3: ADMM maintain two sets of variables X_i and $\bar{X}_i$. X_i satisfies all constraints throughout optimization, while $\bar{X}_i$ can violate constraints during intermediary iterations. On convergence, however, both X_i and $\bar{X}_i$ satisfy all constraints.

$$\underset{\substack{\theta,\Delta t,d_i,d_{ij}\\ \|n_i\|,\|n_{ij}\|=1}}{\text{argmin}} \quad \mathcal{L}(\theta,\Delta t,n_i,d_i,n_{ij},d_{ij}) \triangleq \sum_i \mathcal{O}(X_i(\theta),\Delta t) -$$
$$\gamma \sum_i \log(v_{\max}\Delta t - \|V_i\|) - \gamma \sum_i \log(a_{\max}\Delta t^2 - \|A_i\|) -$$
$$\gamma \sum_i \left[\sum_{x\in X_i} \log(n_i x(\theta) + d_i) + \sum_{z\in Z} \log(-n_i z - d_i) \right] -$$
$$\gamma \sum_{ij} \left[\sum_{x\in X_i} \log(n_{ij} x(\theta) + d_{ij}) + \sum_{x\in X_j} \log(-n_{ij} x(\theta) - d_{ij}) \right], \quad (5)$$

where γ is the weight of log-barrier function that can be tuned for each problem to control the exactness of constraint satisfaction. We use the same subscript convention as Section III. Specifically, for ith trajectory piece X_i and the obstacle Z, we introduce a separating plane $n_i^T \bullet + d_i = 0$. For the pair of ith and jth trajectories pieces that might collide, we introduce a separating plane $n_{ij}^T \bullet + d_{ij} = 0$.

Equation 5 is a strongly coupled problem with six sets of decision variables, where the constraints (or log-barrier functions) and the objective $\mathcal{O}$ are added up. However, these two kinds of functions have very different properties. The log-barrier functions are "stiff" and do not have a Lipschitz constant, which could generate arbitrarily large blocking gradients near the constraint boundaries, but the objective $\mathcal{O}$ is well-conditioned, oftentimes having a finite Lipschitz constant. Our main idea is to handle these functions in separate subproblems.

Algorithm 1: AM

Input: $\theta^0, \Delta t^0, \left(n_i, d_i\right)^0, \left(n_{ij}, d_{ij}\right)^0$

for $k = 0, 1, \cdots$ **do** ▷ Update $\theta, \Delta t$

  $\theta^{k+1}, \Delta t^{k+1} \approx \underset{\theta,\Delta t}{\text{argmin}}\ \mathcal{L}(\theta, \Delta t, n_i^k, d_i^k, n_{ij}^k, d_{ij}^k)$

  for collision constraint between ith piece of trajectory and environment **do** ▷ Optimize separating plane

    $\left(n_i, d_i\right)^{k+1} \approx \underset{\|n_i\|=1, d_i}{\text{argmin}}\ \mathcal{L}(\theta^{k+1}, \Delta t^{k+1}, n_i, d_i, n_{ij}^k, d_{ij}^k)$

  for collision constraint between ith and jth piece of trajectory **do**

    $\left(n_{ij}, d_{ij}\right)^{k+1} \approx \underset{\|n_{ij}\|=1, d_{ij}}{\text{argmin}}$

    $\mathcal{L}(\theta^{k+1}, \Delta t^{k+1}, n_i^{k+1}, d_i^{k+1}, n_{ij}, d_{ij})$

A. Alternating Minimization (AM)

Before we describe our ADMM-type method, we review the basic alternating minimization scheme. AM has been used in trajectory optimization to handle time-optimality [38] and collision constraints [39]. A similar method can be applied to minimize Equation 5 that alternates between updating the separating plane n_i, d_i, n_{ij}, d_{ij} and the robot configurations θ as outlined in AM 1. AM can be used along with ADMM while being easier to analyze. Prior work [39] did not provide a convergence analysis and Wang *et al.* [38] setup the first order convergence for a specific, strictly convex objective function where each minimization subproblem is a single-valued map. In the appendices of our arxiv version [40], we establish the convergence of AM 1 for twice-differentiable objective functions with plane normals n_i, n_{ij} constrained to the unit circle/sphere. In the next section, we will combine AM and ADMM, specifically we update robot configurations θ using ADMM and update separating planes n_i, d_i, n_{ij}, d_{ij} using AM.

Algorithm 2: ADMM with Stiffness Decoupling

Input: $\theta^0, \Delta t^0, \Delta\bar{t}_i^0, \bar{X}_i^0, \lambda_i^0, \left(n_i, d_i\right)^0, \left(n_{ij}, d_{ij}\right)^0$

for $k = 0, 1, \cdots$ **do** ▷ Update $\theta, \Delta t, \bar{X}_i, \lambda_i$

  $\theta^{k+1}, \Delta t^{k+1} \approx \underset{\theta, \Delta t}{\text{argmin}}\ \mathcal{L}(\theta, \Delta t, \Delta\bar{t}_i, \bar{X}_i^k, \lambda_i^k, n_i^k, d_i^k, n_{ij}^k, d_{ij}^k)$

  for ith piece of trajectory **do**

    $\Delta\bar{t}_i^{k+1}, \bar{X}_i^{k+1} \approx \underset{\Delta\bar{t}_i, \bar{X}_i}{\text{argmin}}$

    $\mathcal{L}(\theta^{k+1}, \Delta t^{k+1}, \Delta\bar{t}_i, \bar{X}_i, \lambda_i^k, n_i^k, d_i^k, n_{ij}^k, d_{ij}^k)+$

    $\frac{\varrho}{2}\|\bar{X}_i - \bar{X}_i^k\|^2$

    $\lambda_i^{k+1} \leftarrow \lambda_i^k + \varrho(X_i(\theta^{k+1}) - \bar{X}_i^{k+1})$

    $\Lambda_i^{k+1} \leftarrow \Lambda_i^k + \varrho(\Delta t^{k+1} - \Delta\bar{t}_i^{k+1})$

  for collision-free constraint between ith piece of trajectory and environment **do** ▷ Optimize separating plane

    $\left(n_i, d_i\right)^{k+1} \approx \underset{\|n_i\|=1, d_i}{\text{argmin}}$

    $\mathcal{L}(\theta^{k+1}, \Delta t^{k+1}, \Delta\bar{t}_i^{k+1}, \bar{X}_i^{k+1}, \lambda_i^{k+1}, n_i, d_i, n_{ij}^k, d_{ij}^k)$

  for collision-free constraint between ith and jth piece of trajectory **do**

    $\left(n_{ij}, d_{ij}\right)^{k+1} \approx \underset{\|n_{ij}\|=1, d_{ij}}{\text{argmin}}$

    $\mathcal{L}(\theta^{k+1}, \Delta t^{k+1}, \Delta\bar{t}_i^{k+1}, \bar{X}_i^{k+1}, \lambda_i^{k+1}, n_i^{k+1}, d_i^{k+1}, n_{ij}, d_{ij})$

B. ADMM with Stiffness Decoupling

The key idea behind ADMM is to treat stiff and non-stiff functions separately by introducing slack variables. Specifically, we introduce slack variables $\bar{X}_i$ for each $i = 1, \cdots, N$ and transforms Equation 5 into the following equivalent form:

$$\begin{aligned}
&\underset{\substack{\theta, \Delta t, \Delta\bar{t}_i, \bar{X}_i \\ d_i, d_{ij}, \|n_i\|, \|n_{ij}\|=1}}{\text{argmin}} \quad \sum_i \left[\mathcal{O}(\bar{X}_i, \Delta\bar{t}_i)\right] - \\
&\gamma \sum_i \log(v_{\max}\Delta t - \|V_i\|) - \sum_i \gamma \log(a_{\max}\Delta t^2 - \|A_i\|) - \\
&\gamma \sum_i \left[\sum_{x \in X_i} \log(n_i x(\theta) + d_i) + \sum_{z \in Z} \log(-n_i z - d_i)\right] - \\
&\gamma \sum_{ij} \left[\sum_{x \in X_i} \log(n_{ij} x(\theta) + d_{ij}) + \sum_{x \in X_j} \log(-n_{ij} x(\theta) - d_{ij})\right] \\
&\text{s.t. } X_i(\theta) = \bar{X}_i \wedge \Delta t = \Delta\bar{t}_i.
\end{aligned} \tag{6}$$

By convention, we use a bar to indicate slack variables, i.e. $\bar{V}_i(s) = \dot{A}(s)\bar{X}_i$ and $\bar{A}_i(s) = \ddot{A}(s)\bar{X}_i$.

***Remark** 4.1:* We choose to have all slack variables reside in Cartesian space. As a result, if non-linear forward kinematics functions are used, ADMM must handle nonlinear constraint $X_i(\theta) = \bar{X}_i$.

ADMM proceeds by transforming the equality constraints in Equation 6 into augmented Lagrangian terms. We arrive at the following augmented Lagrangian function:

$$\begin{aligned}
&\mathcal{L}(\theta, \Delta t, \Delta\bar{t}_i, \bar{X}_i, \lambda_i, n_i, d_i, n_{ij}, d_{ij}) \triangleq \sum_i \mathcal{O}(\bar{X}_i, \Delta\bar{t}_i) - \\
&\gamma \sum_i \log(v_{\max}\Delta t - \|V_i\|) - \gamma \sum_i \log(a_{\max}\Delta t^2 - \|A_i\|) - \\
&\gamma \sum_i \left[\sum_{x \in X_i} \log(n_i x(\theta) + d_i) + \sum_{z \in Z} \log(-n_i z - d_i)\right] - \\
&\gamma \sum_{ij} \left[\sum_{x \in X_i} \log(n_{ij} x(\theta) + d_{ij}) + \sum_{x \in X_j} \log(-n_{ij} x(\theta) - d_{ij})\right] + \\
&\sum_i \frac{\varrho}{2}\|X_i(\theta) - \bar{X}_i\|^2 + \lambda_i^T(X_i(\theta) - \bar{X}_i) + \\
&\sum_i \frac{\varrho}{2}\|\Delta t - \Delta\bar{t}_i\|^2 + \Lambda_i^T(\Delta t - \Delta\bar{t}_i),
\end{aligned} \tag{7}$$

where ϱ is the penalty parameter, λ_i is the augmented Lagrangian multiplier for $\bar{X}_i$. We can now present our ADMM algorithm seeking stationary points of Equation 7. Each iteration of our ADMM 2 is a five-way update that alternates between $\{\theta, \Delta t\}, \{\Delta\bar{t}_i, \bar{X}_i\}, \{\lambda_i, \Lambda_i\}, \left(n_i, d_i\right), \left(n_{ij}, d_{ij}\right)$. Note that our objective function only appears in the $\{\Delta\bar{t}_i, \bar{X}_i\}$-subproblem, which does not involve any stiff, log-barrier functions. Therefore, ADMM 2 achieves stiffness decoupling.

***Remark** 4.2:* For each optimization subproblem of ADMM 2, we assume that the decision variable is initialized from last iteration. In the appendices of our arxiv version [40], we show that subproblems in AM 1 and ADMM 2 only need to be solved approximately. Specifically, we update $\theta, \Delta t, n_i, d_i, n_{ij}, d_{ij}$ using a single (Riemannian) line search step, and we update $\Delta\bar{t}, \bar{X}$ using a linearized function $\mathcal{L}$. For brevity, we denote these approximate oracles using an $\approx$ symbol.

***Remark** 4.3:* ADMM always maintains two representations of the trajectory, X_i and $\bar{X}_i$, where X_i is used to satisfy the collision constraints and $\bar{X}_i$ (the slack variable) focuses on minimizing the objective function $\mathcal{O}$ at the risk of violating the collision constraints, as illustrated in Figure 3. It is know that using slack variables can loosen constraint satisfaction. As a result, we choose to formulate collision constraints and other hard constraints on X_i instead of $\bar{X}_i$, so that all the constraints can be satisfied using a line-search step on X_i variables. For collision constraints in particular, we inherit the line-step technique from [9] that is safe-guarded by CCD. The CCD procedure ensures that there always exists a separating plane to split each pair of convex objects and the Lagrangian function $\mathcal{L}$ always takes a finite value. On convergence, however, the two representations coincide and both collision-free and local optimal conditions hold.

***Remark** 4.4:* When updating the separating plane, the normal vector must be constrained to have unit length. These constraints can be reparameterized as an optimization on $\mathcal{SO}(3)$. Specifically, given the current solution denoted as n_{ij}^{curr} with $\|n_{ij}^{curr}\| = 1$, we reparameterize n_{ij} by pre-multiplying a rotation matrix $R = \exp(r)$ by n_{ij}^{curr}, where we use the Rodriguez formula to parameterize a rotation matrix as the exponential of an arbitrary 3-dimensional vector r. Instead of using n_{ij} as decision variables, we let $n_{ij} = \exp(r)n_{ij}^{curr}$ and use r as our decision variables. Whichever value r takes, we can ensure $\|n_{ij}\| = 1$ (we refer reads to [42] for more details).

V. Evaluations

Our implementation uses C++11. Experiments are performed on a workstation with a 3.5 GHz Intel Core i9 processor. For experiments, we choose a unified set of parameters $v_{max} = 2m/s, a_{max} = 2m/s^2$ for UAVs (resp. $v_{max} = 0.1m/s, a_{max} = 0.1m/s^2$ for articulated bodies), $w = 10^8, \varrho = 0.1$ unless otherwise stated. We use the same weight, $\gamma = 10$, for all the log-barrier functions. Although fine-tuning γ separately for each log-barrier function can lead to better results, we find the same $\gamma = 10$ achieves reasonably

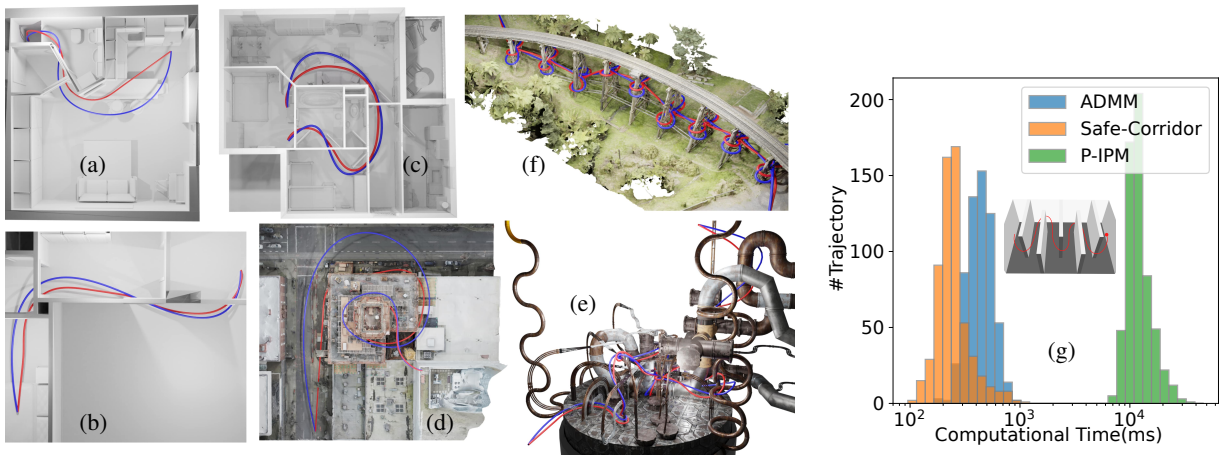

Fig. 4: Examples of trajectories generated by Safe-Corridor [41] (blue) and ADMM 2 (red) for a single UAV in complex environments: indoor flight (a-c), outdoor flight (a-f). (g): The distributions of computational time of ADMM 2, P-IPM [9], and Safe-Corridor [41] over 600 trajectories computed for a synthetic environment in which we compute 600 random initial trajectories using RRT-connect.

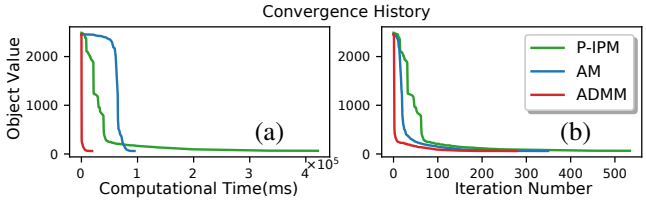

Fig. 5: Convergence history of various algorithms for the example in Figure 4 (f), comparing AM, ADMM, and Newton-type method (P-IPM) in terms of wall-time (a) and #Iterations (b).

Trajectory Length(m) / Flying Time(s)				
Example	AM	ADMM	P-IPM [9]	Safe-Corridor [41]
(a)	16.0/9.6	16.0/9.6	16.1/9.7	18.0/14.0
(b)	17.9/10.4	17.9/10.4	18.1/10.6	19.0/11.3
(c)	13.2/8.9	13.2/9.0	13.8/9.1	14.6/11.7
(d)	39.1/21.8	39.2/21.9	39.2/21.8	47.2/25.1
(e)	66.8/44.7	66.5/44.6	68.1/45.5	71.2/57.9
(f)	67.3/62.8	67.2/62.8	70.1/65.3	74.6/74.1
Computational Cost(ms)				
Example	AM	ADMM	P-IPM [9]	Safe-Corridor [41]
(a)	388	45	1.0K	41
(b)	133	34	2.8K	50
(c)	313	123	7.8K	79
(d)	8.7K	938	34.3K	51
(e)	39.0K	6.4K	175.0K	864
(f)	92.9K	18.9K	420.1K	5.6K

TABLE I: The quality of results in terms of trajectory length/flying time and computational cost of the three methods (ours, P-IPM [9], and Safe-Corridor [41]) for the six examples of Figure 4.

good results over all examples. We use a locally supported log-barrier function as done in our prior work [9], which is active only when the distance between two objects is less than $0.1m$ for UAVs (resp. $0.04m$ for articulated bodies). Further, we set our clearance distance to be $0.1m$ for UAVs (resp. $10^{-3}m$ for articulated bodies), which can be plugged into CCD used by our line-search algorithm. Our algorithm terminates when $\|\nabla_{\theta^{k+1}}\mathcal{L}\|_\infty < \epsilon$ and we choose $\epsilon = 10^{-2}$ for UAVs (resp. $\epsilon = 10^{-1}$ for articulated bodies). By comparing with our prior work [9], we demonstrate the ability of ADMM in terms of resolving stiff-coupling issues and boosting the overall perform. We further compare with prior works [8, 41] to highlight the robustness of our approach.

Fast Separating Plane Update: We found that iterative separating plane updates is a major computational bottleneck. Fortunately, mature collision detection algorithms such as the Gilbert–Johnson–Keerthi (GJK) algorithm [43] can quickly return the optimal separating direction. We emphasize that our separating planes minimizing the soft log-barrier penalties do not match the separating directions returned by GJK in general. However, we found that the separating plane return by GJK oftentimes leads to a reduction in the Lagrangian function, while being orders of magnitude faster to compute due to their highly optimized implementation. Therefore, we propose to use the GJK algorithm for updating the separating planes. After each iteration of of either AM 1 or ADMM 2, we check whether the Lagrangian function $\mathcal{L}$ is decreasing. If $\mathcal{L}$ increases, we fallback to our standard log-barrier functions so the overall algorithm conforms to our convergence guarantee.

Single-UAV: We first show six examples of trajectory optimization for a single UAV in complex environments as illustrated in Figure 4. For each example, we compare our method against two baselines [9, 41]. We initialize all three methods using the same feasible trajectory that is manually designed. Our AM 1 takes from 133 to $93K(ms)$ to convergence and our ADMM 2 takes 34 to $18.9K(ms)$, as compared with our prior work [9] taking $1K$ to $420K(ms)$. The convergence history for three of the algorithms are summarized in Figure 5. Although the convergence speed is comparable to P-IPM in terms of number of iterations, our two methods (AM and ADMM) clearly outperform in terms of computational time. By not restricting the trajectory to precomputed corridors as done in [28, 44, 45, 41, 46], our method allows a larger solution space and returns a shorter trajectory. We summarize the quality of trajectory as computed by three methods in Table I. Finally, we conduct large-scale experiments using a synthetic problem illustrated in Figure 4g, where we randomly compute feasible initial trajectories using RRT-connect for 600 times and compare the computational speed of ADMM 2 and [9, 41]. The resulting plot Figure 4g shows more than an order of magnitude's speedup over P-IPM using stiffness decoupling, with our trajectories having similar

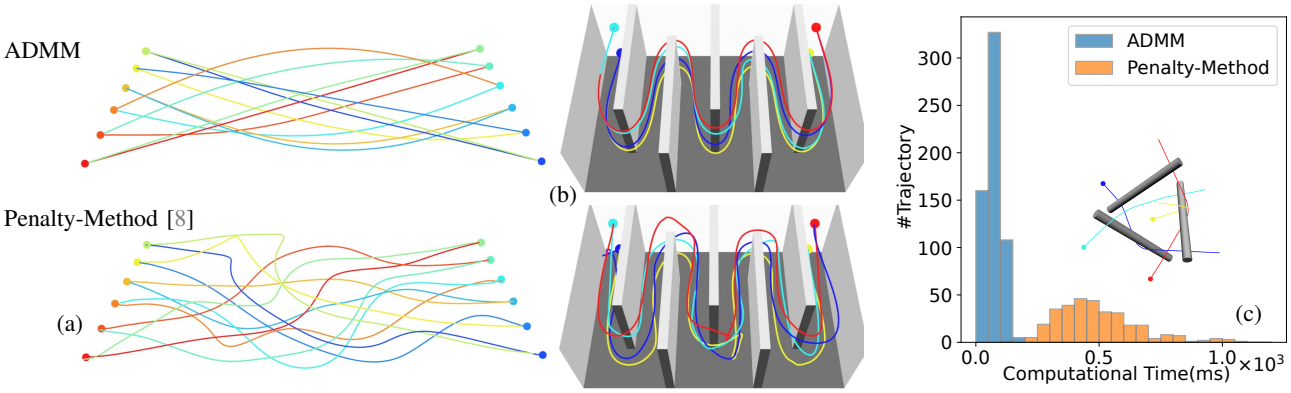

Fig. 6: Multiple UAV in complex environments, both with two groups of UAVs switching positions without (a) and with (b) obstacles. (c): The distributions of computational time of ADMM 2 and Penalty-Method [8] over 600 trajectories computed for a synthetic environment in which we randomly generate initial trajectories for the 4 UAVs in different homotopy classes.

quality (mean/variance trajectory length $14.7/2 \times 10^{-2}$ using ADMM versus $15.0/4 \times 10^{-2}$ using P-IPM, and mean/varance flying time $12.7/2 \times 10^{-5}$ using ADMM versus $12.8/4 \times 10^{-5}$ using P-IPM). Although the computational speed of Safe-Corridor [41] is slightly faster, our results give shorter trajectories (mean/variance trajectory length $15.9/8 \times 10^{-2}$ and mean/variance flying time $14.8/3 \times 10^{-2}$ using [41]).

Multi-UAV: We assume the trajectory of each UAV is represented by composite Bézier curves with degree $M = 5$. We use a bounding-volume hierarchy and only add collision constraints when the distance between two convex hulls are less than the activation distance of log-barrier function ($0.1m$). Note that this abrupt change in number of collision constraints will not hinder the convergence of ADMM because it can happen at most finitely many times. We compute initial trajectories using RRT connect. Our ADMM algorithm minimizes the jerk of each trajectory with time optimality as our objective function $O(\bar{X}_i, \Delta\bar{t}_i)$. Figure 6 shows two challenging problems. We compare our method with penalty method [8], which uses soft penalty terms to push trajectories out of the obstacles. This work is complementary to our method, which allows initial guesses to penetrate obstacles but cannot ensure final result to be collision-free. Instead, our method must start from a collision-free initial guess and maintain the collision-free guarantee throughout the optimization. In Table II, we compare the quality of solutions and computational cost of these two methods. We have also conducted a large-scale comparison with penalty method [8] as illustrated in Figure 6c, where we randomly generate 600 initial trajectories that cover different homotopy classes. As summarized in Figure 6c, our method generates trajectories with faster computing time and a 100% success rate, while [8] can only achieve a success rate of 53%. Our results have higher quality than [8], e.g. the mean/variance of trajectory length 77.8/17.8 (ours) vs 80.1/21.3 ([8]) and flying time 11.3/0.4 (ours) vs 20.6/8.0 ([8]).

Trajectory Length(m) / Flying Time(s) / Computational Cost(ms)		
Example	ADMM	Penalty-Method [8]
(a)	152.68/7.84/515	169.34/14.48/844.49
(b)	71.33/14.57/2527	89.82/20.73/584.89

TABLE II: The quality of results in terms of trajectory length/flying time and computational cost, comparing our method and Penalty-Method [8].

Articulated Robot Arm: We highlight the performance of our method via an example involving two arms. We approximate each robot link as a single convex object to reduce the number of separating planes. Our example is inspired by prior work [47], as illustrated in Figure 7, where we have two KUKA LWR robot arms (each with 8 links) switch positions of their end-effectors. From an initial trajectory computed with the length 6.38m using RRT-connect, our stiffness-decoupled ADMM method can easily minimize acceleration of end-effects obtaining a trajectory with the length 1.63m in 16s. The convergence history is shown in Figure 8.

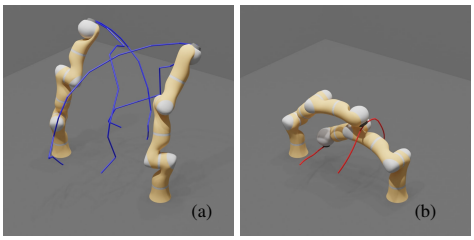

Fig. 7: Trajectory optimization for two KUKA LWR robot arms switching end-effector positions. (a): initial trajectory via RRT-connect; (b): optimized trajectory.

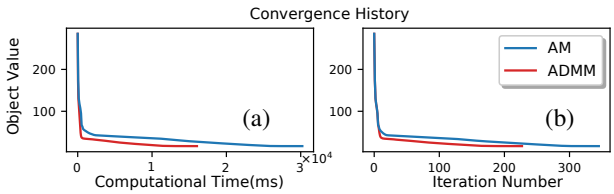

Fig. 8: Convergence history for the example in Figure 7, comparing AM and stiffness-decoupled ADMM in terms of wall-time (a) and #Iterations (b).

VI. Conclusion & Limitation

We propose a variant of ADMM-type solver for trajectory optimization. We observe that the limited efficacy of our prior work [9] is mainly due to the stiff log-barrier functions corresponding to various hard constraints. Therefore, we propose to decompose stiff and non-stiff objective function terms using slack variables, while using additional constraints to ensure their consistency. ADMM was originally applied to convex

optimizations and we establish its convergence guarantee under non-convex objectives and constraints that arise from UAV and articulated robot kinematics. Our experiments confirm that ADMM successfully resolves stiff-coupling issues and achieves tens of times' speedup over Newton-type algorithms. The major limitation of our method is the requirement of a strictly feasible initial trajectory and our convergence rate can also be dependent on the initial guess. As a result, our method cannot be used for receding-horizon settings. If the horizons are truncated, then there can be unforeseen obstacles, with which collisions are ignored, resulting in infeasible trajectories in future horizons. Finally, the convergence of ADMM-type solvers for general control of nonlinear dynamic systems [32] remains an open question.

REFERENCES

[1] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 65–100, 2010.

[2] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*, IEEE, 2010, pp. 5420–5425.

[3] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ international conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 1917–1922.

[4] J. T. Betts and W. P. Huffman, "Path-constrained trajectory optimization using sparse sequential quadratic programming," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 1, pp. 59–68, 1993.

[5] C. Park, J. Pan, and D. Manocha, "Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS'12, Atibaia, São Paulo, Brazil: AAAI Press, 2012, 207–215.

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

[7] 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*, 2011, pp. 4569–4574.

[8] X. Zhou, X. Wen, J. Zhu, H. Zhou, C. Xu, and F. Gao, "Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," *arXiv:2011.04183*, 2020.

[9] R. Ni, T. Schneider, D. Panozzo, Z. Pan, and X. Gao, "Robust & asymptotically locally optimal uav-trajectory generation based on spline subdivision," *arXiv:2010.09904*, 2020.

[10] L. F. Shampine and C. W. Gear, "A user's view of solving stiff ordinary differential equations," *SIAM review*, vol. 21, no. 1, pp. 1–17, 1979.

[11] W. Gao, D. Goldfarb, and F. E. Curtis, "Admm for multiaffine constrained optimization," *Optimization Methods and Software*, vol. 35, no. 2, pp. 257–303, 2020.

[12] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 2513–2518.

[13] Y. Choi, M. Chen, Y. Choi, S. Briceno, and D. Mavris, "Multi-uav trajectory optimization utilizing a nurbs-based terrain model for an aerial imaging mission," *Journal of Intelligent & Robotic Systems*, vol. 97, no. 1, pp. 141–154, 2020.

[14] M. Ragaglia, A. M. Zanchettin, and P. Rocco, "Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements," *Mechatronics*, vol. 55, pp. 267–281, 2018.

[15] M. M. G. Ardakani, J. H. Cho, R. Johansson, and A. Robertsson, "Trajectory generation for assembly tasks via bilateral teleoperation," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10 230–10 235, 2014.

[16] 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.

[17] S. Shimoda, Y. Kuroda, and K. Iagnemma, "Potential field navigation of high speed unmanned ground vehicles on uneven terrain," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 2828–2833.

[18] Z. Qu, J. Wang, and C. E. Plaisted, "A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 978–993, 2004.

[19] M. Shomin and R. Hollis, "Differentially flat trajectory generation for a dynamically stable mobile robot," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 4467–4472.

[20] K. Belghith, F. Kabanza, L. Hartman, and R. Nkambou, "Anytime dynamic path-planning with flexible probabilistic roadmaps," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 2372–2377.

[21] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 1478–1483.

[22] O. Von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization," *Annals of operations research*, vol. 37, no. 1, pp. 357–373, 1992.

[23] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 5026–5033.

[24] Z. Cheng, J. Ma, X. Zhang, and T. H. Lee, "Semi-proximal admm for model predictive control problem with application to a uav system," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 82–87.

[25] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.

[26] 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.

[27] 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.

[28] S. Liu *et al.*, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[29] J. Tordesillas and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Transactions on Robotics*, 2021.

[30] G. Vasudevan, L. T. Watson, and F. Lutze, "A homotopy approach for solving constrained optimization problems," in *1989 American Control Conference*, IEEE, 1989, pp. 780–785.

[31] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[32] Z. Pan and D. Manocha, "Efficient solver for spacetime control of smoke," *ACM Trans. Graph.*, vol. 36, no. 5, Jul. 2017.

[33] Z. Zhou and Y. Zhao, "Accelerated admm based trajectory optimization for legged locomotion with coupled rigid body dynamics," in *2020 American Control Conference (ACC)*, IEEE, 2020, pp. 5082–5089.

[34] B. Jiang, T. Lin, S. Ma, and S. Zhang, "Structured nonconvex and nonsmooth optimization: Algorithms and iteration complexity analysis," *Computational Optimization and Applications*, vol. 72, no. 1, pp. 115–157, 2019.

[35] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[36] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[37] P. Cheng, J. Keller, and V. Kumar, "Time-optimal uav trajectory planning for 3d urban structure coverage," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 2750–2757.

[38] Z. Wang, X. Zhou, C. Xu, J. Chu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," *IEEE RA-L*, vol. 5, no. 3, pp. 4836–4843, 2020.

[39] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[40] R. Ni, Z. Pan, and X. Gao, "Provably robust & efficient multi-robot trajectory generation using alternating direction method of multiplier," *arXiv:2111.07016*, 2021.

[41] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for se (3) planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, 2021.

[42] C. J. Taylor and D. J. Kriegman, "Minimization on the lie group so (3) and related manifolds," 1994.

[43] 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.

[44] 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.

[45] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019.

[46] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *arXiv:2103.00190*, 2021.

[47] H. Ha, J. Xu, and S. Song, "Learning a decentralized multi-arm motion planner," in *Conference on Robotic Learning (CoRL)*, 2020.