IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2019, accepted June 30, 2019, date of publication July 4, 2019, date of current version July 25, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2926782

Matrix Structure Driven Interior Point Method for Quadrotor Real-Time Trajectory Planning

GUANGTONG XU, TENG LONG, ZHU WANG, AND YAN CAO
School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China
Key Laboratory of Dynamics and Control of Flight Vehicle, Ministry of Education, Beijing 100081, China

Corresponding author: Zhu Wang (wangzhubit@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 51675047, and in part by the China Postdoctoral Science Foundation under Grant 2018M631361.

ABSTRACT Sequential convex programming (SCP) has been recently employed in various trajectory planning problems, including entry flight, planetary landing, and aircraft formation. In SCP, convex programming subproblems are sequentially solved to obtain the optimum of original nonconvex problems. For SCP-based quadrotor trajectory planning, this paper proposes a matrix-structure-driven interior point method (MSD-IPM) to improve the efficiency of solving search directions in convex programming. In MSD-IPM, primal-dual systems for solving search directions are derived from the Karush–Kuhn–Tucher (KKT) conditions of quadrotor trajectory planning subproblems. Then, the successive elimination technique is used to solve the inverse of large-scale coefficient matrices of primal-dual systems by more efficient operations on small-scale matrices. In successive elimination, the positive definiteness of several small-scale matrices is used to enhance the numerical stability of computing search directions, and the specific diagonal structures of small-scale matrices are exploited to efficiently compute the search directions. The complexity analysis shows that the efficiency of the proposed method is about one order of magnitude higher than that of the standard IPM. The comparative studies on simulation experiments demonstrate that the MSD-IPM generally outperforms several well-known optimizers (e.g., MOSEK, SDPT3, and SeDuMi) in terms of efficiency and robustness. Finally, the indoor trajectory tracking experiments indicate that the proposed method can generate smooth trajectories for real-world applications.

INDEX TERMS Quadrotor trajectory planning, sequential convex programming, interior point method, matrix structure exploitation.

NOMENCLATURE

The following symbols are used in this paper:

$\boldsymbol{A}$	equality constraint matrix of convex programming;
$\boldsymbol{a}$	acceleration vector;
$\boldsymbol{a}_{min}, \boldsymbol{a}_{max}$	acceleration bound;
B	bandwidth of banded matrices;
$\boldsymbol{b}$	equality constraint vector of convex programming;
$\boldsymbol{C}$	inequality constraint matrix of convex programming;
$\boldsymbol{d}$	inequality constraint vector of convex programming;
$\boldsymbol{e}$	column vector with all elements set to one;
$\boldsymbol{G}$	objective function matrix of convex programming;
$\boldsymbol{g}$	objective function vector of convex programming;
h	search step size of MSD-IPM;
I	current iteration number of MSD-IPM;
$I_{\max}$	maximum iteration number of MSD-IPM;
K	number of discretized intervals;
$\boldsymbol{L}$	coordinate matrix of primal-dual systems;
l	number of equality constraints;
$[\boldsymbol{lb}, \boldsymbol{ub}]$	bound constraint of convex programming;
M	number of obstacles;
N	number of design variables;
N_c	number of control variables;
N_s	number of state variables;
n_c	number of control variables at each discrete point;

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Luo.

n_s	number of state variables at each discrete point;
$\boldsymbol{p}$	position vector;
$\boldsymbol{p}_m^C$	center position of circular obstacles;
$\bar{\boldsymbol{p}}$	position vector of nominal trajectories;
$\boldsymbol{p}_{min}, \boldsymbol{p}_{max}$	position bound;
q	number of inequality constraints;
$\boldsymbol{r}$	right-hand side of primal-dual systems;
r_m^C	radius of circular obstacles;
$\Re_m$	obstacle region;
$\mathbb{R}$	real;
$\mathbb{R}^n$	n dimension column vector;
$\mathbb{R}^{n\times n}$	$n \times n$ matrix;
S_m	scale of coordinate matrix $\boldsymbol{L}$;
$\boldsymbol{s}_{sta}$	state variable;
$\boldsymbol{s}; \boldsymbol{u}; \boldsymbol{w}$	slack variable of MSD-IPM;
t	time of trajectories;
t_0	initial time of trajectories;
t_f	final time of trajectories;
Δt	discretized step size of trajectories;
$\boldsymbol{v}$	velocity vector;
$\boldsymbol{v}_{min}, \boldsymbol{v}_{max}$	velocity bound;
$\boldsymbol{x}$	design variable of convex programming;
$\boldsymbol{X}$	primal-dual solution of convex programming;
$\Delta \boldsymbol{X}$	search direction of MSD-IPM;
$\Delta \boldsymbol{X}^{\text{aff}}$	affine search direction of MSD-IPM;
ε	convergence tolerance vector of SCP;
ε_e	convergence tolerance of MSD-IPM;
σ	centering corrective parameter of MSD-IPM;
μ	complementary measure of MSD-IPM;
$\boldsymbol{\lambda}; \boldsymbol{\xi}; \boldsymbol{\vartheta}; \boldsymbol{\gamma}$	dual variable of MSD-IPM.

I. INTRODUCTION

Due to the attractive features in low-cost, maneuverability, flexibility, and hovering-ability, quadrotors are appropriate for various applications including surveillance and reconnaissance, disaster and crisis management, infrastructure inspection, agriculture and forestry, and express delivery [1]. Most of these applications require quadrotors to have high-level autonomous capabilities to execute complex missions in hazardous environments [2]. Currently, trajectory-based autonomous control is a common approach to fulfill such requirements [3]. Real-time trajectory planning is the foundation for trajectory-based control of quadrotors in dynamic environments [4]–[6].

The developed real-time trajectory planning methods can be classified into two categories [6]. First, real-time trajectory planning is solved from a hierarchical planning perspective [7]–[9]. A geometric path is constructed preliminarily, and it is then parameterized in time domain subject to dynamic constraints to generate smooth trajectories. Hierarchical planning methods are efficient, but sacrifice the optimality of trajectories [10]. Second, the trajectory planning is formulated as an optimal control problem (OCP), which can be solved by indirect and direct methods [11], [12]. OCPs have received great attentions in a number of research groups [13]–[17] due to its appealing tradeoffs between computational efficiency and optimality. Since it is hard to derive the necessary condition of problems with complex constraints for indirect methods, the direct method is mostly used to solve OCPs for real-world problems. The direct method transcribes the infinite dimensional OCP into a finite dimensional parameter optimization problem [15], which in general is a high-dimensional nonconvex problem and is computationally intractable for real-time implementation. Fortunately, the efficient convex optimization method [18] have been successfully used to tackle the trajectory planning problem for saving computational cost. By using convexification techniques [19] including equivalent transformation [20], successive linearization [21], and relaxation [22], the nonconvex trajectory planning problem is converted into convex optimization problems, which can be solved efficiently and robustly by the state-of-the-art interior point method (IPM) [23]. Recently, IPM has been successfully applied in various trajectory planning problems, such as spacecraft rendezvous [24], entry flight [25], guidance of aircraft [26], and planetary landing [27].

In the field of convex-optimization-based quadrotor real-time trajectory planning, Augugliro *et al.* [28] first proposed a sequential convex programming (SCP) method to enhance the computational efficiency. Dueri *et al.* [29] combined SCP with inter-sample obstacle avoidance methods to ensure that the generated trajectories do not cross obstacles between discrete states. Chen *et al.* [30] developed an incremental SCP (iSCP) to incrementally add the collision-avoidance constraints during the iterative process of SCP, which leads to significant improvement in computational tractability. Szmuk *et al.* [31] implemented real-time on-board trajectory generation using SCP for high-performance quadrotor flight in indoor flight platforms. Besides the leverage of SCP, the computational efficiency can also be improved by using customized convex optimizers [32], [33] which can exploit the problem structure to automatically generating custom codes and have been applied to landing guidance fields. Dueri *et al.* [34] developed the customized IPM using the sparsity of the planetary powered-descent guidance (PDG) problem to reduce the runtime by two to three orders of magnitude. However, Dueri's IPM [34] was specifically tailored for PDG problems, which cannot be directly used for other problems (e.g., quadrotor trajectory planning problems). To the best of the authors' knowledge, quite few studies were reported to customize efficient IPM for quadrotor trajectory planning problems. In this work, we develop a matrix structure driven interior-point method (MSD-IPM) according to the unique characteristics of quadrotor trajectory planning to improve the computational efficiency.

The main contributions of this paper are listed as follows. I) MSD-IPM is proposed to exploit specific structures of the equality and inequality constraint matrixes in the quadrotor trajectory planning problem. II) The algorithm complexity of MSD-IPM is analyzed, and the results illustrate that MSD-IPM costs only about one-tenth CPU time of the standard IPM (i.e., Mehrotra's IPM) [35]. III) The appealing

performance of MSD-IPM is demonstrated on comparative simulation experiments and trajectory tracking experiments in an indoor flight platform.

The paper is organized as follows. The formulation of quadrotor trajectory planning problem is given in section II. In section III, convex quadratic programming subproblems of quadrotor trajectory planning are established and SCP is introduced. The proposed MSD-IPM method is presented in section IV. In section V, comparative studies between MSD-IPM and several convex optimization toolboxes are performed on simulation experiments. In section VI, indoor flight experiments are implemented. Finally, conclusions are given in section VII.

II. PROBLEM FORMULATION

Quadrotor trajectory planning can generate flight trajectories for optimizing specific performance index (e.g., minimizing the flight time or minimizing energy consumptions) subject to initial and final state, state/control bound, quadrotor dynamic, and obstacle-avoidance constraints. Figure 1 illustrates a scenario of quadrotor trajectory planning where the generated trajectory guides the quadrotor to reach its destination and avoid multiple obstacles.

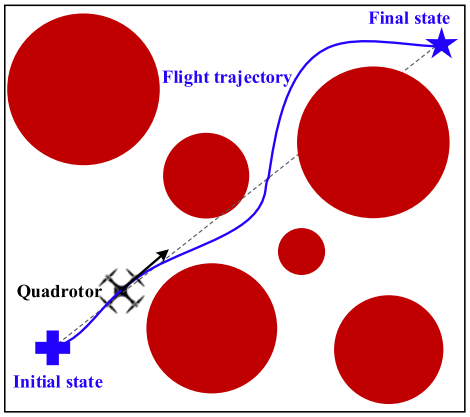


FIGURE 1. Illustration of quadrotor trajectory planning problem.

In this paper, the fixed-time obstacle-avoidance trajectory planning problem for quadrotors is formulated as a nonconvex optimal control problem (NOCP) shown in Eq. (1). The quadrotor trajectory planning aims at minimizing the sum of the total thrust [28].

$$
\begin{aligned}
&\textit{Problem}-I\ (\textbf{\textit{NOCP}}):\\
&\min_{\boldsymbol{p}(t),\ \boldsymbol{v}(t),\boldsymbol{a}(t)} J=\int_{t_0}^{t_f}\|\boldsymbol{a}(t)\|_2^2\\
&\quad \textbf{\textit{subject to}}\ \dot{\boldsymbol{p}}(t)=\boldsymbol{v}(t)\\
&\qquad\qquad \dot{\boldsymbol{v}}(t)=\boldsymbol{a}(t)\\
&\qquad\qquad \boldsymbol{p}(t_0)=\boldsymbol{p}_0,\boldsymbol{v}(t_0)=\boldsymbol{v}_0\\
&\qquad\qquad \boldsymbol{p}(t_f)=\boldsymbol{p}_f,\boldsymbol{v}(t_f)=\boldsymbol{v}_f\\
&\qquad\qquad \boldsymbol{p}_{\min}\le\boldsymbol{p}(t)\le\boldsymbol{p}_{\max},\quad t\in[t_0,t_f]\\
&\qquad\qquad \boldsymbol{v}_{\min}\le\boldsymbol{v}(t)\le\boldsymbol{v}_{\max},\quad t\in[t_0,t_f]\\
&\qquad\qquad \boldsymbol{a}_{\min}\le\boldsymbol{a}(t)\le\boldsymbol{a}_{\max},\quad t\in[t_0,t_f]\\
&\qquad\qquad \boldsymbol{p}(t)\notin\Re_m,\quad m=1,2,\ldots,M
\end{aligned}
\tag{1}
$$

where $\boldsymbol{p}=(p_x,p_y)^{\mathrm{T}}$, $\boldsymbol{v}=(v_x,v_y)^{\mathrm{T}}$, and $\boldsymbol{a}=(a_x,a_y)^{\mathrm{T}}$. $(\boldsymbol{p}_0;\boldsymbol{v}_0)$ and $(\boldsymbol{p}_f;\boldsymbol{v}_f)$ denote the initial and final state respectively. $\boldsymbol{p}_{\min}$, $\boldsymbol{p}_{\max}$, $\boldsymbol{v}_{\min}$, $\boldsymbol{v}_{\max}$, $\boldsymbol{a}_{\min}$, and $\boldsymbol{a}_{\max}$ denote the bound constraints on states and controls. Only circular obstacles are considered in this paper, and domain of obstacles $\Re_m$ can be expressed as

$$
\Re_m=\left\{\boldsymbol{p}\middle|\left\|\boldsymbol{p}-\boldsymbol{p}_m^C\right\|_2<r_m^C\right\},\quad m=1,2,\ldots,M \tag{2}
$$

where $\|\cdot\|_2$ represents the 2-norm.

III. SEQUENTIAL CONVEX PROGRAMMING FOR TRAJECTORY PLANNING

In this section, the NOCP of quadrotor trajectory planning is transcribed into a series of convex quadratic programming subproblems via discretization and successive convexification. And SCP is introduced to solve the trajectory planning problem.

A. CONVEX QUADRATIC PROGRAMMING FORMULATION

The direct collocation method [11] is applied to reformulate the *Problem-I* as a parameter optimization problem. The flight duration is uniformly divided into predefined K intervals with a constant step size $\Delta t=(t_f-t_0)/K$. Then the trajectory is approximated with $K+1$ points at t_k, where $t_k=t_0+k\Delta t$, $k=0,1,\ldots,K$. The state and control variables are discretized as $\boldsymbol{p}[k]=\boldsymbol{p}(t_k)$, $\boldsymbol{v}[k]=\boldsymbol{v}(t_k)$, and $\boldsymbol{a}[k]=\boldsymbol{a}(t_k)$. The kinematics of quadrotors are discretized using the trapezoidal method [11]. The obstacle-avoidance constraints $\boldsymbol{p}(t)\notin\Re_m$ are discretized as $\boldsymbol{p}[k]\notin\Re_m$, which are still concave constraints [29].

To reformulate the *Problem-I* as convex optimization problems, concave constraints $\boldsymbol{p}[k]\notin\Re_m$ is tackled via successive convexification method [17]. The obstacle-avoidance constraints $\boldsymbol{p}[k]\notin\Re_m$ can be approximated with affine constraints at discrete time in Eq. (3) [17]. In addition, the inter-sample obstacle-avoidance constraint in Eq. (4) [17] is introduced to enhance the safety of quadrotors between discrete time.

$$
\begin{aligned}
&||\bar{\boldsymbol{p}}[k]-\boldsymbol{p}_m^C||+\frac{(\bar{\boldsymbol{p}}[k]-\boldsymbol{p}_m^C)^T}{||\bar{\boldsymbol{p}}[k]-\boldsymbol{p}_m^C||}\cdot(\boldsymbol{p}[k]-\bar{\boldsymbol{p}}[k])\ \ge r_m^C\\
&m=1,2,\ldots,M
\end{aligned}
\tag{3}
$$

where $\bar{\boldsymbol{p}}[k]$ denotes the nominal position of quadrotors at time t_k.

$$
\begin{aligned}
&||\bar{\boldsymbol{p}}[k-1]-\boldsymbol{p}_m^C||+\frac{(\bar{\boldsymbol{p}}[k-1]-\boldsymbol{p}_m^C)^T}{||\bar{\boldsymbol{p}}[k-1]-\boldsymbol{p}_m^C||}\\
&\cdot(\boldsymbol{p}[k]-\bar{\boldsymbol{p}}[k-1])\ \ge r_m^C\ m=1,2,\ldots,M,\ k=1,\ldots,K
\end{aligned}
\tag{4}
$$

Through discretization and convexification, the convex optimization subproblem of the quadrotor trajectory planning is formulated as *Problem-II* in Eq. (5). The obstacle-avoidance trajectories are acquired by iteratively solving the

Problem-II. The detailed iterative method is described in section III-B.

Problem-II

(***Convex quadratic optimization sub–problem***) :

$$\min_{\boldsymbol{p}[k],\boldsymbol{v}[k],\boldsymbol{a}[k],k=0,1,\ldots,K} J' = \sum_{k=0}^{K} \|\boldsymbol{a}[k]\|_2^2$$

$$\begin{aligned}
&\textit{subject to } \boldsymbol{p}[k+1] = \boldsymbol{p}[k] + \frac{\Delta t}{2} \cdot (\boldsymbol{v}[k] + \boldsymbol{v}[k+1]) \\
&\boldsymbol{v}[k+1] = \boldsymbol{v}[k] + \frac{\Delta t}{2} \\
&\cdot (\boldsymbol{a}[k] + \boldsymbol{a}[k+1]) \\
&\boldsymbol{p}[0] = \boldsymbol{p}_0, \boldsymbol{v}[0] = \boldsymbol{v}_0 \\
&\boldsymbol{p}[K] = \boldsymbol{p}_f, \boldsymbol{v}[K] = \boldsymbol{v}_f \\
&\boldsymbol{p}_{\min} \le \boldsymbol{p}[k] \le \boldsymbol{p}_{\max}, \ k = 0, 1, \ldots, K \\
&\boldsymbol{v}_{\min} \le \boldsymbol{v}[k] \le \boldsymbol{v}_{\max}, \ k = 0, 1, \ldots, K \\
&\boldsymbol{a}_{\min} \le \boldsymbol{a}[k] \le \boldsymbol{a}_{\max}, \ k = 0, 1, \ldots, K \\
&\textit{Eqs.}(3) \textit{ and } (4) \qquad (5)
\end{aligned}$$

B. SEQUENTIAL CONVEX PROGRAMMING

The conservative convexification of obstacle-avoidance constraints may lead to an infeasible problem [36]. Thus, SCP is developed to alleviate the over-conservative approximations and successively improve trajectory results in an iterative framework [17]. The procedures of SCP for quadrotor trajectory planning are shown in Algorithm 1 [17].

Algorithm 1 Sequential Convex Programming for Trajectory Planning

Input : initial/final states and control $(\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{p}_f, \boldsymbol{v}_f, \boldsymbol{a}_0, \boldsymbol{a}_f)$
obstacle regions $\Re_m (m = 1, \ldots, M)$
bounds ($\boldsymbol{p}_{\min}$, $\boldsymbol{p}_{\max}$, $\boldsymbol{v}_{\min}$, $\boldsymbol{v}_{\max}$, $\boldsymbol{a}_{\min}$, and $\boldsymbol{a}_{\max}$)
number of intervals $K, j \leftarrow 0$
time step size Δt, tolerance vector ε

Output: optimized trajectory of quadrotors

1: $(\boldsymbol{p}^0,\boldsymbol{v}^0,\boldsymbol{a}^0) \leftarrow$ initializeStraightLine($\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{p}_f, \boldsymbol{v}_f, \Delta t, K$)
2: $(\bar{\boldsymbol{p}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{a}}) \leftarrow (\boldsymbol{p}^0,\boldsymbol{v}^0,\boldsymbol{a}^0)$
3: **while** constraints in Eq. (5) are not satisfied or convergence criteria in Eq. (6) are not satisfied **do**
4: $j \leftarrow j+1$
5: *Problem-II* $\leftarrow$ formConvexproblem($\bar{\boldsymbol{p}}, \bar{\boldsymbol{v}}$, bound, and $\Re_m$)
6: $(\boldsymbol{p}^j,\boldsymbol{v}^j,\boldsymbol{a}^j) \leftarrow$ solve(*Problem-II*)
7: $(\bar{\boldsymbol{p}}, \bar{\boldsymbol{v}}, \bar{\boldsymbol{a}}) \leftarrow (\boldsymbol{p}^j,\boldsymbol{v}^j,\boldsymbol{a}^j)$
8: **end while**
9: **return**$(\boldsymbol{p}^j,\boldsymbol{v}^j,\boldsymbol{a}^j)$

The initial trajectories ($\boldsymbol{p}^0$, $\boldsymbol{v}^0$, $\boldsymbol{a}^0$), i.e., straight lines connecting the initial position and final position of quadrotors, are generated without considering obstacle/collision avoidance constraints (line 1). And ($\boldsymbol{p}^0$, $\boldsymbol{v}^0$, $\boldsymbol{a}^0$) are utilized as nominal trajectories ($\bar{\boldsymbol{p}}$, $\bar{\boldsymbol{v}}$, $\bar{\boldsymbol{a}}$) for the next iteration (line 2). In the iterative process, a series of convex programming subproblems are constructed and solved considering all the constraints (lines 4-7). The iterative process terminates when all of the trajectory constraints and the convergence criterion in Eq. (6) [31] are satisfied (lines 3 and 8).

$$\left| \boldsymbol{s}_{sta}^{j} - \boldsymbol{s}_{sta}^{j-1} \right|_{\infty} \le \boldsymbol{\varepsilon} \qquad (6)$$

where $\boldsymbol{s}_{sta} = (\boldsymbol{p}[0]; \boldsymbol{v}[0]; \boldsymbol{p}\ [1]; \boldsymbol{v}\ [1]; \ldots, \boldsymbol{p}[k]^{\mathrm{T}}; \boldsymbol{v}[k]; \ldots, \boldsymbol{p}[K]; \boldsymbol{v}[K])$; j denotes the index of iterations; ε represents the tolerance vector.

IV. MATRIX STRUCTURE DRIVEN INTERIOR POINT METHOD

This section presents MSD-IPM to solve *Problem-II*. The procedure of MSD-IPM is provided, and then the details including KKT condition derivation, primal-dual system construction, and search direction solver are described. The algorithm complexity analysis is conducted to demonstrate the effectiveness of MSD-IPM.

A. PROCEDURE OF MSD-IPM

MSD-IPM utilizes predictor-corrector algorithm [37] to iteratively refine the solutions of *Problem-II*. The flowchart of MSD-IPM is illustrated in Fig. 2. And the procedure is described as follows.

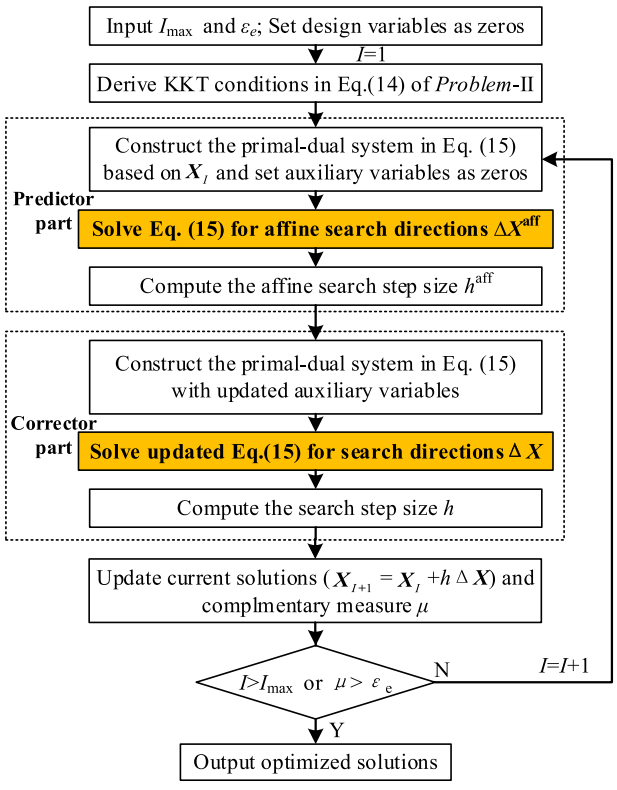


FIGURE 2. Flowchart of MSD-IPM. The differences between MSD-IPM and Mehrotra's IPM are highlighted.

Step (1): Initialization. The initial parameters ($I_{\max}$ and ε_e) of MSD-IPM are configured. The design variables are initialized to be zeros.

Step (2): Derivation of KKT conditions. The KKT conditions of *Problem-II* in Eq. (14) is derived from the primal problem in Eq. (7) and the corresponding dual problem in Eq. (13).

Step (3): Construction of primal-dual systems. Based on current solution $\boldsymbol{X}_I$, nonlinear KKT conditions are converted into linear primal-dual systems in Eq. (15) to solve the search direction.

Step (4): Predictor step. The primal-dual systems are solved to obtain the affine search direction $\Delta \boldsymbol{X}^{\text{aff}}$ by our customized search direction solver, which is the unique technical contribution of this paper and will be detailed in section IV-C. And the affine search step size h^{aff} is acquired by line search.

Step (5): Corrector step. A corrected primal-dual system is constructed by updating τ_1, τ_2, and τ_3 with Eq. (18) and solved to generate the search direction $\Delta \boldsymbol{X}$ using the customized solver. And search step size h is obtained by line search.

Step (6): Solution update. The current solution $\boldsymbol{X}_I$ is updated as $\boldsymbol{X}_{I+1} = \boldsymbol{X}_I + h\Delta \boldsymbol{X}$.

Step (7): Convergence check. If the number of iterations I exceeds $I_{\max}$ or complementary measure μ satisfies the convergence error ε_e, the iterative process terminates and the current best solution is output as the optimized trajectory. Otherwise, the process returns to Step 3.

B. KKT CONDITIONS OF PROBLEM-II

Problem-II in Eq. (5) can be rewritten as the following form for simplification.

$$\begin{aligned} &\min \frac{1}{2}\boldsymbol{x}^T\boldsymbol{G}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} \\ &s.t.\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\quad \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d} \\ &\quad \boldsymbol{lb} \le \boldsymbol{x} \le \boldsymbol{ub} \end{aligned} \tag{7}$$

where $\boldsymbol{x}$=($\boldsymbol{p}$; $\boldsymbol{v}$; $\boldsymbol{a}$)$\in \mathbb{R}^N$ and $N = (n_s + n_c)\cdot(K+1)$. The objective function is a quadratic polynomial, where $\boldsymbol{G} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{g} \in \mathbb{R}^N$ are expressed in Eq. (8).

$$\boldsymbol{G} = \begin{bmatrix} 0_{N_s\times N_s} & 0 \\ 0 & \boldsymbol{I}_{N_c\times N_c} \end{bmatrix}, \quad \boldsymbol{g} = 0_{n\times 1} \tag{8}$$

where $N_s = n_s(K+1)$ and $N_c = n_c(K+1)$.

Constraint conditions include $l = n_s(K+3)+2n_c$ equality constraints ($\boldsymbol{A} \in \mathbb{R}^{l\times N}$ and $\boldsymbol{b} \in \mathbb{R}^l$), $q = 2KM$ inequality constraints ($\boldsymbol{C} \in \mathbb{R}^{q\times n}$ and $\boldsymbol{d} \in \mathbb{R}^q$), and $2N$ bound constraints ($\boldsymbol{lb} \in \mathbb{R}^N$ and $\boldsymbol{ub} \in \mathbb{R}^N$).

In equality constraints, the forms of coefficient matrix $\boldsymbol{A}$ and right-hand side $\boldsymbol{b}$ are given in Eq. (9). The superscripts "*ini*", "*fin*", and "*kin*" denote the matrices and vectors of initial state, final state, and kinematic constraints, respectively. The subscripts of matrices inside and outside parentheses denote the matrix index and matrix dimension respectively. Note that $\boldsymbol{A}$ is a block banded matrix, as shown in Eq. (10).

$$\boldsymbol{A} = \begin{bmatrix} (\boldsymbol{A}^{ini})_{n\times N} \\ (\boldsymbol{A}^{kin})_{(N_s+n_s)\times N} \\ (\boldsymbol{A}^{fin})_{n\times N} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} (\boldsymbol{b}^{ini})_{n\times 1} \\ (\boldsymbol{b}^{kin})_{(N_s+n_s)\times 1} \\ (\boldsymbol{b}^{fin})_{n\times 1} \end{bmatrix} \tag{9}$$

where $n = n_s + n_c$ denotes the number of design variables at each discrete point.

$$\boldsymbol{A} = \begin{bmatrix} \left(\boldsymbol{A}^{ini}\right)_{n_s\times n} & & & & & & & \\ \left(\boldsymbol{A}^{kin}_{0,0}\right)_{n_s\times n} & \ddots & & & & & & \\ \left(\boldsymbol{A}^{kin}_{1,0}\right)_{n_s\times n} & \left(\boldsymbol{A}^{kin}_{1,1}\right)_{n_s\times n} & \ddots & & & & & \\ 0 & \left(\boldsymbol{A}^{kin}_{2,1}\right)_{n_s\times n} & \left(\boldsymbol{A}^{kin}_{2,2}\right)_{n_s\times n} & \ddots & & & & \\ & \ddots & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & 0 & & \\ & & & \ddots & \ddots & \left(\boldsymbol{A}^{kin}_{K,K}\right)_{n_s\times n} & & \\ & & & & \ddots & \left(\boldsymbol{A}^{kin}_{K+1,K}\right)_{n_s\times n} & & \\ & & & & & \left(\boldsymbol{A}^{fin}\right)_{n_s\times n} & & \end{bmatrix} \tag{10}$$

In inequality constraints, structures of coefficient matrix $\boldsymbol{C}$ and right-hand side $\boldsymbol{d}$ are shown in Eq. (11). The subscript "*obs*" denotes obstacle avoidance constraints. Note that matrix $\boldsymbol{C}_{obs,i}$ has the block diagonal properties as shown in Eq. (12).

$$\boldsymbol{C} = \begin{bmatrix} (\boldsymbol{C}_1^{obs})_{K\times N} \\ \vdots \\ (\boldsymbol{C}_{2M}^{obs})_{K\times N} \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} (\boldsymbol{d}_1^{obs})_{K\times 1} \\ \vdots \\ (\boldsymbol{d}_{2M}^{obs})_{K\times 1} \end{bmatrix} \tag{11}$$

where,

$$(\boldsymbol{C}_i^{obs})_{K\times N} = \begin{bmatrix} (\boldsymbol{C}_{i,0}^{obs})_{1\times n} & & & \\ & (\boldsymbol{C}_{i,1}^{obs})_{1\times n} & & \\ & & \ddots & \\ & & & (\boldsymbol{C}_{i,K}^{obs})_{1\times n} \end{bmatrix}, \quad i = 1, \ldots, m \tag{12}$$

where $\boldsymbol{C}_{i,k}^{obs} \in \mathbb{R}^{1\times n}$, $k = 0, \ldots, K$ denotes the inequality constraints for i^{th} obstacle at discrete time t_k.

The dual form of *Problem-II* is generally used to derive the KKT conditions in IPM [37]. By introducing dual variables ($\boldsymbol{\lambda} \in \mathbb{R}^l$, $\boldsymbol{\xi} \in \mathbb{R}^q$, $\boldsymbol{\vartheta} \in \mathbb{R}^N$, and $\boldsymbol{\gamma} \in \mathbb{R}^N$) and slack variables ($\boldsymbol{s} \in \mathbb{R}^q$, $\boldsymbol{u} \in \mathbb{R}^N$, and $\boldsymbol{w} \in \mathbb{R}^N$), the dual form of primal problem is formulated in Eq. (13).

$$\begin{aligned} &\max \boldsymbol{\lambda}^T\boldsymbol{b} + \boldsymbol{\xi}^T\boldsymbol{d} - \boldsymbol{\vartheta}^T\boldsymbol{lb} + \boldsymbol{\gamma}^T\boldsymbol{ub} \\ &s.t.\ \boldsymbol{A}^T\boldsymbol{\lambda} - \boldsymbol{G}\boldsymbol{x} - \boldsymbol{g} - \boldsymbol{C}^T\boldsymbol{\xi} + \boldsymbol{\vartheta} - \boldsymbol{\gamma} = 0 \\ &\quad \boldsymbol{\xi}^T\boldsymbol{s} = 0 \\ &\quad \boldsymbol{\vartheta}^T\boldsymbol{u} = 0 \\ &\quad \boldsymbol{\gamma}^T\boldsymbol{w} = 0 \\ &\quad (\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\vartheta}, \boldsymbol{\gamma}) \ge 0 \end{aligned} \tag{13}$$

where $\boldsymbol{\xi}^T\boldsymbol{s}$, $\boldsymbol{\vartheta}^T\boldsymbol{u}$, and $\boldsymbol{\gamma}^T\boldsymbol{w}$ represent the complementary conditions.

According to the primal problem in Eq. (7) and dual problem in Eq. (13), the KKT conditions of *Problem-II* are derived in Eq. (14) [18].

$$\begin{aligned} \boldsymbol{A}^T\boldsymbol{\lambda} - \boldsymbol{G}x - \boldsymbol{g} - \boldsymbol{C}^T\boldsymbol{\xi} + \boldsymbol{\vartheta} - \boldsymbol{\gamma} &= 0 \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} &= 0 \\ \boldsymbol{C}\boldsymbol{x} + \boldsymbol{s} - \boldsymbol{d} &= 0 \\ \boldsymbol{x} - \boldsymbol{u} - \boldsymbol{lb} &= 0 \\ \boldsymbol{x} + \boldsymbol{w} - \boldsymbol{ub} &= 0 \\ \boldsymbol{S}\boldsymbol{E}\boldsymbol{e_1} &= 0 \\ \boldsymbol{U}\boldsymbol{Q}\boldsymbol{e_2} &= 0 \\ \boldsymbol{W}\boldsymbol{Y}\boldsymbol{e_3} &= 0 \\ (\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\vartheta}, \boldsymbol{\gamma}) &\geq 0 \end{aligned} \tag{14}$$

where $\boldsymbol{S} \in \mathbb{R}^{q\times q}$, $\boldsymbol{E} \in \mathbb{R}^{q\times q}$, $\boldsymbol{U} \in \mathbb{R}^{N\times N}$, $\boldsymbol{Q} \in \mathbb{R}^{N\times N}$, $\boldsymbol{W} \in \mathbb{R}^{N\times N}$, and $\boldsymbol{Y} \in \mathbb{R}^{N\times N}$ are the diagonal matrices, whose diagonal elements are the corresponding elements of $\boldsymbol{s}$, $\boldsymbol{\xi}$, $\boldsymbol{u}$, $\boldsymbol{\vartheta}$, $\boldsymbol{w}$, and $\boldsymbol{\gamma}$ respectively; $e_1 \in \mathbb{R}^q$, $e_2 \in \mathbb{R}^N$, and $e_3 \in \mathbb{R}^N$ are column vectors with all elements set to one.

C. PRIMAL-DUAL SYSTEM

The first order Taylor approximation of the KKT conditions in Eq. (14) near current solutions ($\boldsymbol{x}$; $\boldsymbol{\lambda}$; $\boldsymbol{\xi}$; $\boldsymbol{\vartheta}$; $\boldsymbol{\gamma}$; $\boldsymbol{s}$; $\boldsymbol{u}$; $\boldsymbol{w}$) is performed to construct the primal-dual systems in Eq. (15) [35], which is used to solve search direction $\Delta\boldsymbol{X} = (\Delta\boldsymbol{x}; \Delta\boldsymbol{\lambda}; \Delta\boldsymbol{\xi}; \Delta\boldsymbol{\vartheta}; \Delta\boldsymbol{\gamma}; \Delta\boldsymbol{s}; \Delta\boldsymbol{u}; \Delta\boldsymbol{w})$.

$$\boldsymbol{L}\cdot\Delta\boldsymbol{X} = \boldsymbol{r} \tag{15}$$

where matrix $\boldsymbol{L} \in \mathbb{R}^{S_m\times S_m}$ and vector $\boldsymbol{r} \in \mathbb{R}^{S_m}$ are the coefficient matrix and the right-hand side of the primal-dual system respectively. The expressions of $\boldsymbol{L}$ and $\boldsymbol{r}$ are given in Eqs. (16) and (17), where $S_m = 5N + 2q + l$.

$$\boldsymbol{L}_{S_m\times S_m} = \begin{bmatrix} -\boldsymbol{G} & \boldsymbol{A}^T & -\boldsymbol{C}^T & \boldsymbol{I} & -\boldsymbol{I} & 0 & 0 & 0 \\ \boldsymbol{A} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \boldsymbol{C} & 0 & 0 & 0 & 0 & \boldsymbol{I} & 0 & 0 \\ \boldsymbol{I} & 0 & 0 & 0 & 0 & 0 & -\boldsymbol{I} & 0 \\ \boldsymbol{I} & 0 & 0 & 0 & 0 & 0 & 0 & \boldsymbol{I} \\ 0 & 0 & \boldsymbol{S} & 0 & 0 & \boldsymbol{E} & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{U} & 0 & 0 & \boldsymbol{Q} & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{W} & 0 & 0 & \boldsymbol{Y} \end{bmatrix} \tag{16}$$

$$\boldsymbol{r}_{S_m\times 1} = \begin{bmatrix} \boldsymbol{G}x + \boldsymbol{g} - \boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{C}^T\boldsymbol{\xi} - \boldsymbol{\vartheta} + \boldsymbol{\gamma} \\ \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \\ \boldsymbol{d} - \boldsymbol{C}\boldsymbol{x} - \boldsymbol{s} \\ \boldsymbol{lb} - \boldsymbol{x} + \boldsymbol{u} \\ \boldsymbol{ub} - \boldsymbol{x} - \boldsymbol{w} \\ \tau_1 - \boldsymbol{S}\boldsymbol{E}\boldsymbol{e_1} \\ \tau_2 - \boldsymbol{U}\boldsymbol{Q}\boldsymbol{e_2} \\ \tau_3 - \boldsymbol{W}\boldsymbol{Y}\boldsymbol{e_3} \end{bmatrix} \tag{17}$$

In predictor step, auxiliary variable τ_1, τ_2, and τ_3 are set to be zero and affine search direction $\Delta\boldsymbol{X}^{\text{aff}} = (\Delta\boldsymbol{x}^{\text{aff}}; \Delta\boldsymbol{\lambda}^{\text{aff}}; \Delta\boldsymbol{\xi}^{\text{aff}}; \Delta\boldsymbol{\vartheta}^{\text{aff}}; \Delta\boldsymbol{\gamma}^{\text{aff}}; \Delta\boldsymbol{s}^{\text{aff}}; \Delta\boldsymbol{u}^{\text{aff}}; \Delta\boldsymbol{w}^{\text{aff}})$ is obtained by solving Eq. (15). In the corrector step, τ_1, τ_2, and τ_3 are updated according to Eq. (18) [35]. And the updated primal-dual system is solved again with renewed right-hand side $\boldsymbol{r}$ to determine the correctional search direction $\Delta\boldsymbol{X}$.

$$\begin{cases} \tau_1 = (\sigma\mu - \Delta\boldsymbol{S}^{aff}\Delta\boldsymbol{E}^{aff})\boldsymbol{e}_1 \\ \tau_2 = (\sigma\mu - \Delta\boldsymbol{U}^{aff}\Delta\boldsymbol{Q}^{aff})\boldsymbol{e}_2 \\ \tau_3 = (\sigma\mu - \Delta\boldsymbol{W}^{aff}\Delta\boldsymbol{Y}^{aff})\boldsymbol{e}_3 \end{cases} \tag{18}$$

where $\sigma \in (0, 1)$ and μ denote the centering corrective parameter and the complementary measure, respectively [35]. The expressions of σ and μ are shown in Eqs. (19) and (20).

$$\sigma = (\mu^{aff}/\mu)^3 \tag{19}$$

$$\mu^{aff} = \frac{(\boldsymbol{\xi}^{aff})^T\boldsymbol{s}^{aff} + (\boldsymbol{\vartheta}^{aff})^T\boldsymbol{u}^{aff} + (\boldsymbol{\gamma}^{aff})^T w^{aff}}{2n + l} \tag{20}$$

$$\mu = \frac{\boldsymbol{\xi}^T\boldsymbol{s} + \boldsymbol{\vartheta}^T u + \boldsymbol{\gamma}^T w}{2n + l}$$

D. CUSTOMIZED SEARCH DIRECTION SOLVER

To improve the efficiency of solving the primal-dual systems in Eq. (15), this subsection presents the customized search direction solver incorporated with successive elimination and matrix structure exploitation. The algorithm complexity of the customized search direction solver is analyzed via flop counts. A flop is defined as one addition, subtraction, multiplication, or division operation of two floating-point numbers. The total number of flops is counted as a function (i.e., a polynomial) of the dimensions of matrices and vectors. Ref. [18] summarized the costs of basic matrix and vector operations, as shown in Table 1.

TABLE 1. Flop count of matrix and vector operations [18].

Operation	Scale	Flop Count
$\boldsymbol{x}$+$\boldsymbol{x}$, $\boldsymbol{x}$-$\boldsymbol{x}$	$\boldsymbol{x} \in \mathbb{R}^N$	N
$\boldsymbol{A}$+$\boldsymbol{A}$, $\boldsymbol{A}$-$\boldsymbol{A}$	$\boldsymbol{A} \in \mathbb{R}^{l\times N}$	lN
$\boldsymbol{Ax}$	$\boldsymbol{A} \in \mathbb{R}^{l\times N}$, $\boldsymbol{x} \in \mathbb{R}^N$	$2lN$
$\boldsymbol{AA}^{\text{T}}$	$\boldsymbol{A} \in \mathbb{R}^{l\times N}$	$2lNl$
$\boldsymbol{G}^{-1}$	$\boldsymbol{G} \in \mathbb{R}^{N\times N}$	$2.67N^3$

1) SUCCESSIVE ELIMINATION FOR SOLVING SEARCH DIRECTION

Generally, the primal-dual system in Eq. (15) is solved by directly using matrix inversion, i.e., $\Delta\boldsymbol{X} = \boldsymbol{L}^{-1}\cdot\boldsymbol{r}$. In this paper, successive elimination in Yan's IPM [38] is extended to solve search directions of general convex programming problems in Eq. (5) subject to bound and equality/inequality constraints. Using the successive elimination, the primal-dual system in Eq. (15) is solved via substituting step by step, as shown in Eq. (21). For instance, $\Delta\boldsymbol{\lambda}$ can be calculated in terms of the first line of Eq. (20), which is then substituted into the second line of Eq. (20) to solve $\Delta\boldsymbol{x}$. According to the conclusion in Ref [38], the successive elimination is

numerically more stable because of the positive definiteness of matrix $\boldsymbol{D}$ and $\boldsymbol{ADA}^{-1}$.

$$\begin{aligned}
\Delta\boldsymbol{\lambda} &= (\boldsymbol{ADA}^T)^{-1}(\boldsymbol{b} - \boldsymbol{Ax} + \boldsymbol{AD\eta}) \\
\Delta\boldsymbol{x} &= \boldsymbol{D}(-\boldsymbol{A}^T\Delta\boldsymbol{\lambda} + \boldsymbol{\eta}) \\
\Delta\boldsymbol{s} &= \boldsymbol{d} - \boldsymbol{C}(\Delta\boldsymbol{x} + \boldsymbol{x}) - \boldsymbol{s}, \\
\Delta\boldsymbol{u} &= \Delta\boldsymbol{x} + \boldsymbol{r_{lb}} \\
\Delta\boldsymbol{w} &= -\boldsymbol{r_{ub}} - \Delta\boldsymbol{x} \\
\Delta\boldsymbol{\xi} &= \boldsymbol{S}^{-1}\tau_1 - \boldsymbol{S}^{-1}\boldsymbol{E}\Delta\boldsymbol{s} - \boldsymbol{\xi} \\
\Delta\boldsymbol{\vartheta} &= \boldsymbol{U}^{-1}\tau_2 - \boldsymbol{U}^{-1}\boldsymbol{Q}\Delta\boldsymbol{u} - \boldsymbol{\vartheta} \\
\Delta\boldsymbol{\gamma} &= \boldsymbol{W}^{-1}\tau_3 - \boldsymbol{W}^{-1}\boldsymbol{Y}\Delta\boldsymbol{w} - \boldsymbol{\gamma}
\end{aligned} \tag{21}$$

where,

$$\begin{aligned}
\boldsymbol{D} &= (\boldsymbol{G} + \boldsymbol{C}^T\boldsymbol{S}^{-1}\boldsymbol{EC} + \boldsymbol{U}^{-1}\boldsymbol{Q} + \boldsymbol{W}^{-1}\boldsymbol{Y})^{-1} \\
\boldsymbol{\eta} &= \rho + \boldsymbol{C}^T\boldsymbol{S}^{-1}\tau_1 - \boldsymbol{U}^{-1}\tau_2 + \boldsymbol{W}^{-1}\tau_3 \\
\boldsymbol{\rho} &= \boldsymbol{Gx} + \boldsymbol{g} - \boldsymbol{A}^T\boldsymbol{\lambda} \\
&\quad + \boldsymbol{C}^T\boldsymbol{S}^{-1}\boldsymbol{E}(\boldsymbol{s} - \boldsymbol{d} + \boldsymbol{Cx}) + \boldsymbol{U}^{-1}\boldsymbol{Qr_{lb}} + \boldsymbol{W}^{-1}\boldsymbol{Yr_{ub}} \\
\boldsymbol{r_{lb}} &= -\boldsymbol{lb} + \boldsymbol{x} - \boldsymbol{u} \\
\boldsymbol{r_{ub}} &= -\boldsymbol{ub} + \boldsymbol{x} + \boldsymbol{w}
\end{aligned} \tag{22}$$

Flop counts of generating search direction via successive elimination are shown in Table 2. The expression of flop counts for obtaining $\Delta\boldsymbol{X}$ shown in the last row of Table 2 only involves the leading (i.e., highest order or dominant) terms of polynomials for simplification. The flop counts of solving search direction by successive elimination are smaller than that of directly solving $\Delta\boldsymbol{X} = \boldsymbol{L}^{-1} \cdot \boldsymbol{r}$, i.e., $2.67S_m^3 = 2.67(5N + 2q + l)^3$.

Remark 1: Successive elimination for solving primal-dual systems considering equality/inequality constraints and bounds is an extension of that in Yan's IPM, which cannot address inequality constraints.

Remark 2: Successive elimination has higher computational efficiency compared with direct inversion operation, i.e., $\Delta\boldsymbol{X} = \boldsymbol{L}^{-1} \cdot \boldsymbol{r}$ in terms of the flop counts.

Remark 3: Through successive elimination, coefficient matrix $\boldsymbol{L}$ of the primal-dual system is split into several small-scale matrices including $\boldsymbol{G}$, $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{S}$, $\boldsymbol{E}$, $\boldsymbol{U}$, $\boldsymbol{Q}$, $\boldsymbol{W}$, and $\boldsymbol{Y}$. Hence, the unique diagonal structures, i.e., diagonal, block diagonal, and block banded diagonal of these small-scale matrices shown in Table 3 can be exploited to further reduce the computational consumptions. See next subsection for details.

2) MATRIX STRUCTURE EXPLOITATION

The specific structures of small-scale matrices shown in Table 3 are utilized to improve the computational efficiency of solving search directions.

(I) Computational cost reduction in generating $\boldsymbol{S}^{-1}$, $\boldsymbol{U}^{-1}$, and $\boldsymbol{W}^{-1}$. Since matrices $\boldsymbol{S}$, $\boldsymbol{U}$, and $\boldsymbol{W}$ are diagonal, the inversion of these matrices only costs q, N, and N flop counts respectively. Compared with ordinary matrix inversion ($2.67q^3$, $2.67N^3$, and $2.67N^3$) in Table 2, the inversion of diagonal matrix is more efficient.

TABLE 2. Flop count of solving the search direction via successive elimination and ordinary matrix and vector operations [18].

Term	Scale	Flop count
$\boldsymbol{S}^{-1}$	$\mathbb{R}^{q\times q}$	$2.67q^3$
$\boldsymbol{U}^{-1}$, $\boldsymbol{W}^{-1}$	$\mathbb{R}^{N\times N}$	$2.67N^3$
$\boldsymbol{C}^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{EC}$	$\mathbb{R}^{N\times N}$	$2N^2q+4Nq^2$
$\boldsymbol{U}^{-1}\boldsymbol{Q}$, $\boldsymbol{W}^{-1}\boldsymbol{Y}$	$\mathbb{R}^{N\times N}$	$2N^3$
$\boldsymbol{D}$	$\mathbb{R}^{N\times N}$	$2.67N^3+3N^2$
$\boldsymbol{r_{lb}}$, $\boldsymbol{r}_{\mathrm{ub}}$	$\mathbb{R}^{N}$	$2N$
$\boldsymbol{\rho}$	$\mathbb{R}^{N}$	$6N^2+2Nq+2Nl+5N+2q$
$\boldsymbol{U}^{-1}\tau_2$	$\mathbb{R}^{N}$	$2N^2$
$\boldsymbol{W}^{-1}\tau_3$	$\mathbb{R}^{N}$	$2N^2$
$\boldsymbol{\eta}$	$\mathbb{R}^{N}$	$2Nq+3N$
$(\boldsymbol{ADA}^{\mathrm{T}})^{-1}$	$\mathbb{R}^{l\times l}$	$2.67l^3+2Nl^2+2N^2l$
$\Delta\boldsymbol{\lambda}$	$\mathbb{R}^{l}$	$2N^2l+4Nl+2l^2+2l$
$\Delta\boldsymbol{x}$	$\mathbb{R}^{N}$	$2N^2+2Nl+N$
$\Delta\boldsymbol{s}$	$\mathbb{R}^{q}$	$2Nq+N+2q$
$\Delta\boldsymbol{u}$, $\Delta\boldsymbol{w}$	$\mathbb{R}^{N}$	N
$\Delta\boldsymbol{\xi}$	$\mathbb{R}^{q}$	$2q^2+2q$
$\Delta\boldsymbol{\vartheta}$, $\Delta\boldsymbol{\gamma}$	$\mathbb{R}^{N}$	$2N^2+2N$
$\Delta\boldsymbol{X}$	$\mathbb{R}^{s_m}$	$7.34N^3+2.67q^3+2.67l^3+2N^2q+Nq^2+4N^2l+2Nl^2$

TABLE 3. Unique structure of small-scale matrices.

Matrix	Scale	Structure
$\boldsymbol{G}$	$\mathbb{R}^{N\times N}$	diagonal
$\boldsymbol{S}$, $\boldsymbol{E}$	$\mathbb{R}^{q\times q}$	diagonal
$\boldsymbol{U}$, $\boldsymbol{Q}$, $\boldsymbol{W}$, $\boldsymbol{Y}$	$\mathbb{R}^{N\times N}$	diagonal
$\boldsymbol{A}$	$\mathbb{R}^{l\times N}$	block banded diagonal
$\boldsymbol{C}$	$\mathbb{R}^{q\times N}$	block diagonal

(II) Computational cost reduction in generating $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{EC}$. According to the block diagonal feature of matrix $\boldsymbol{C}$ in Eqs. (11) and (12), matrix $\boldsymbol{CTS}^{-1}\boldsymbol{EC}$ can be rewritten as the summation form in Eq. (23), where $S_{obs,i}^{-1}$ and $E_{obs,i}$ are diagonal matrices. Thus, $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{S}^{-1}\boldsymbol{EC}$ is factorized as the product of a serial of low-dimensional block diagonal matrices.

$$\boldsymbol{C}^T\boldsymbol{S}^{-1}\boldsymbol{EC} = \sum_{i=1}^{M}\left(\boldsymbol{C}_{obs,i}^T\boldsymbol{S}_{obs,i}^{-1}\boldsymbol{E}_{obs,i}\boldsymbol{C}_{obs,i}\right) \tag{23}$$

The flop counts of computing $\boldsymbol{C}^{\mathrm{T}}\ \boldsymbol{S}^{-1}\boldsymbol{EC}$ by Eq. (23), i.e., $N^2q/(K+1)^2+2q$, are much fewer compared with general matrix-matrix product ($2N^2q+4Nq^2$) shown in Table 2.

(III) Computational cost reduction in generating $\boldsymbol{U}^{-1}\boldsymbol{Q}$ and $\boldsymbol{W}^{-1}\boldsymbol{Y}$. The diagonal feature of matrices $\boldsymbol{U}^{-1}$, $\boldsymbol{Q}$, $\boldsymbol{W}^{-1}$, and $\boldsymbol{Y}$ is exploited to reduce the computational burden of generating $\boldsymbol{U}^{-1}\ \boldsymbol{Q}$ and $\boldsymbol{W}^{-1}\ \boldsymbol{Y}$. By using diagonal matrix multiplication, the flop counts are decreased from $2N^3$ to N.

(IV) Computational cost reduction in generating $\boldsymbol{D}$. The inversion of nonsingular symmetric matrix can be computed by $\mathrm{LDL}^{\mathrm{T}}$ factorization [18] to reduce the computational cost.

Thus, the computational cost of generating $\boldsymbol{D}$ is reduced from $2.67N^3+3N^2$ to $0.33N^3+3N^2$ using LDL$^{\mathrm{T}}$ factorization.

(V) Computational cost reduction in generating $(\boldsymbol{ADA}^{\mathrm{T}})^{-1}$. Since matrix $\boldsymbol{A}$ in Eq. (10) is a block banded matrix, $\boldsymbol{ADA}^{\mathrm{T}}$ is also a block banded matrix. Thus, inversion of matrix $\boldsymbol{ADA}^{\mathrm{T}}$ can be obtained via LDL$^{\mathrm{T}}$ factorization to reduce the computational cost from $2.67l^3$ to lB^2, where $B = 4n$ denotes the bandwidth of matrix $\boldsymbol{ADA}^{\mathrm{T}}$.

By utilizing matrix structure exploitation, the flop counts for solving search directions in Eq. (21) are reduced to $(0.33N^3 + N^2q/(K+1)^2+4N^2l+16N^2l/(K+1)^2+2Nl^2)$. Since $N/(K+1)<< N$, the trivial terms $N^2q/(K+1)^2$ and $16N^2l/(K+1)^2$ can be ignored compared with the leading terms. For the quadrotor trajectory problem in this paper, the number of state variables n_s and the number of control variables n_c are set to be 4 and 2 respectively. The relationship of N and l is expressed as follows.

$$\frac{l}{N} = \frac{n_s\,(K+3)+2n_c}{(n_s+n_c)\cdot(K+1)} = \frac{4\,(K+3)+4}{6\,(K+1)} \approx 0.67, \quad K \gg 1 \tag{24}$$

Besides, the relationship of N and q is expressed as follows.

$$\frac{q}{N} = \frac{2KM}{(n_s+n_c)\cdot(K+1)} = \frac{2KM}{6\,(K+1)} \approx \frac{M}{3}, \; K \gg 1 \tag{25}$$

We assume that there are three obstacles in the environment, i.e., the number of obstacles $M = 3$. Then, we can replace q and l with N and $0.67N$, respectively. Thus, the flop count of solving search directions by customized search direction solver is rewritten as $3.91N^3$. Note that the Mehrotra's IPM solves Eq. (15) using LDL$^{\mathrm{T}}$ factorization to obtain the search direction with the flops $0.33(5N+2q+l)^3 \approx 74.54N^3$.

Remark 4: The complexity analyses indicate that the computational efficiency of MSD-IPM is about one order of magnitude higher than that of the Mehrotra's IPM. Although the current MSD-IPM method is developed for quadrotor trajectory planning, the idea of matrix structure exploitation can be used to quadrotor swarm, fixed-wing UAV, and spacecraft trajectory planning.

E. SEARCH STEP SIZE

Once search direction $\Delta\boldsymbol{X} = (\Delta\boldsymbol{x};\, \Delta\,\boldsymbol{\lambda};\, \Delta\,\boldsymbol{\xi};\, \Delta\vartheta;\, \Delta\,\boldsymbol{\gamma};\, \Delta\boldsymbol{s};\, \Delta\boldsymbol{u};\, \Delta\boldsymbol{w})$ is determined, search step size h can be obtained by line search as shown in Eq. (26) [35].

$$\begin{aligned} h &= \max\{h \in [0,\,1] : (\boldsymbol{s}',\,\boldsymbol{u}',\,\boldsymbol{w}',\,\boldsymbol{\xi}',\,\boldsymbol{\vartheta}',\,\boldsymbol{\gamma}') \geq 0\} \\ \boldsymbol{s}' &= \boldsymbol{s} + h\Delta s,\, \boldsymbol{u}' = \boldsymbol{u} + h\Delta\boldsymbol{u} \\ \boldsymbol{w}' &= \boldsymbol{w} + h\Delta\boldsymbol{w},\, \boldsymbol{\xi}' = \boldsymbol{\xi} + h\Delta\boldsymbol{\xi} \\ \boldsymbol{\vartheta}' &= \boldsymbol{\vartheta} + h\Delta\boldsymbol{\vartheta},\, \boldsymbol{\gamma}' = \boldsymbol{\gamma} + h\Delta\boldsymbol{\gamma} \end{aligned} \tag{26}$$

V. NUMERICAL SIMULATIONS

In this section, numerical simulations are performed on several obstacle-avoidance scenarios to demonstrate the effectiveness of MSD-IPM. And the robustness tests of MSD-IPM on stochastic obstacle-avoidance cases are also conducted. The runtime of MSD-IPM is compared with four representative methods to show its appealing computational efficiency for real-time trajectory planning. Numerical simulations are implemented in MATLAB R2017a on a desktop computer equipped with Intel Core i7-6700 3.40 GHz and 8GB RAM.

Parameters of four scenarios are listed in Table 4. The information of obstacles is listed in Table 5. The initial states and final states are set to be [0m, 0m, 0m/s, 0m/s] and [3m, 1m, 0m/s, 0m/s] respectively. The bounds of flight zones, velocities and accelerations are provided in Eq. (26).

$$\begin{aligned} \boldsymbol{p}_{\min} &= [-1m, -1m],\, \boldsymbol{p}_{\max} = [4m,\, 2m] \\ \boldsymbol{v}_{\min} &= [-1m/s, -1m/s],\, \boldsymbol{v}_{\max} = [1m/s,\, 1m/s] \\ \boldsymbol{a}_{\min} &= [-2.5m/s^2, -2.5m/s^2],\, \boldsymbol{a}_{\max} = [2.5m/s^2,\, 2.5m/s^2] \end{aligned} \tag{27}$$

TABLE 4. Parameter setting of simulation scenarios.

Scenario	Discretization number (K)	Discretized time step (Δt)	Obstacle
I	30	0.80	Obs_1, Obs_2
II	40	0.60	Obs_1 - Obs_3
III	50	0.48	Obs_1 - Obs_5
IV	60	0.40	Obs_1 - Obs_7

TABLE 5. Information of obstacles.

Obstacle #	Position	Radius
Obs_1	[0.80m, 0.10m]	0.35m
Obs_2	[1.50m, 0.60m]	0.35m
Obs_3	[2.40m, 0.50m]	0.35m
Obs_4	[1.70m, -0.15m]	0.35m
Obs_5	[2.50m, 1.25m]	0.35m
Obs_6	[0.60m, 0.83m]	0.35m
Obs_7	[1.60m, 1.40m]	0.35m

The tolerance vector ε in Algorithm 1 is set to be $(0.01,\, 0.01,\, 0.01,\, 0.01)^{\mathrm{T}}$. The convergence error ε_e and maximum number of iterations $I_{\max}$ of MSD-IPM are set to be 10^{-6} and 100 respectively.

A. TRAJECTORY PLANNING RESULTS

The trajectory results for different scenarios are shown in Fig. 3. MSD-IPM succeeds in generating feasible and optimized trajectories for all simulations. The initial trajectory is generated as a straight line connecting the initial and end points, illustrated as the dashed line. Generated feasible trajectories shown as dash-dotted lines satisfy initial/final state, bound, kinematic, and obstacle-avoidance constraints, but have an inferior performance index. In contrast, optimal trajectories shown as the lines with circle marks are solved for the best performance index at the expense of more computational time.

B. ROBUSTNESS TEST

Robustness tests are conducted to verify the numerical stability of MSD-IPM based on the four scenarios in Section V-A.

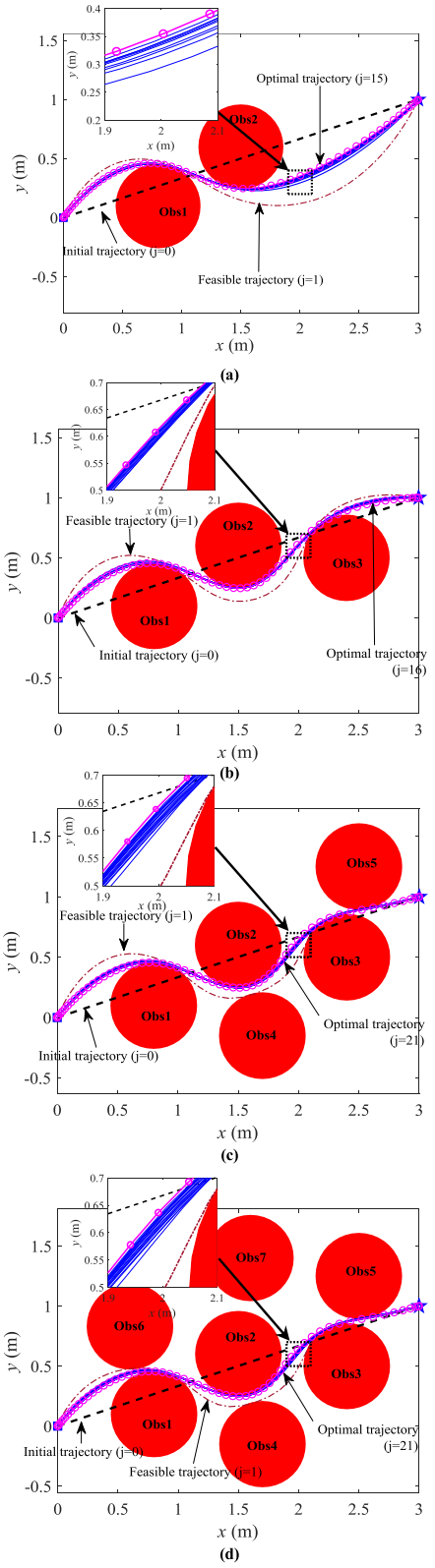


FIGURE 3. Simulation result of the obstacle-avoidance trajectory planning for (a) Scenario-I, (b) Scenario-II, (c) Scenario-III, and (d) Scenario-IV. The 'square' and 'star' denote the start point and end point respectively. The feasible trajectory is planned in the first iteration. It is needed 15, 16, 21, 21 iterations for obtaining optimal trajectory for these four scenarios, respectively. The local zoom-in figures show trajectories (as denoted by blue lines) generated in iterative process.

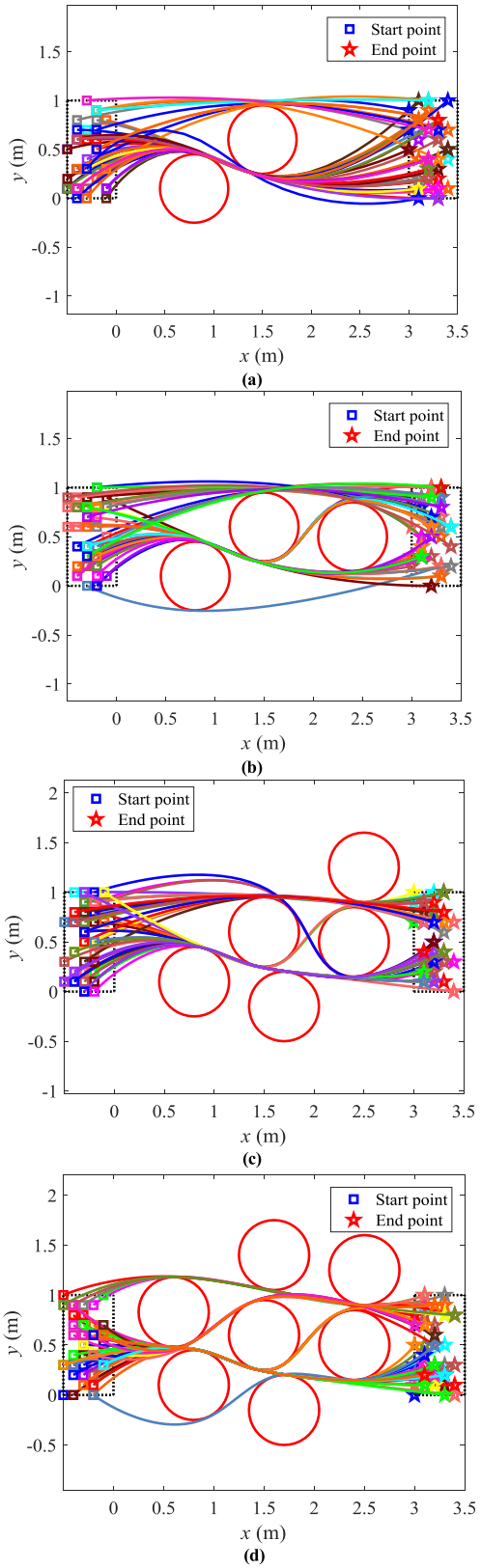


FIGURE 4. Optimized trajectories of the robustness test for (a) Scenario-I, (b) Scenario-II, (c) Scenario-III, and (d) Scenario-IV. The "red circles" denote obstacles. The "solid lines" represent the optimized trajectories. The dashed rectangles in the left and right sides restrict the areas of initial and end positions respectively.

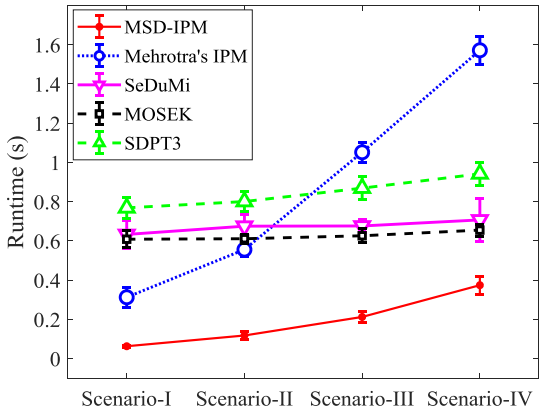


FIGURE 5. Average runtime of generating feasible trajectories by MSD-IPM, Mehrotra's IPM [35], SeDuMi [41], MOSEK [39], and SDPT3 [40]. Each algorithm is run on 50 randomly generated test cases for these four scenarios. Error bars indicate the standard deviation.

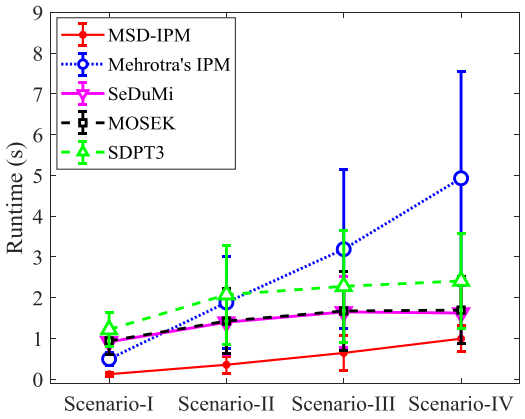


FIGURE 6. Average runtime of generating optimized trajectories. Error bars indicate the standard deviation.

For each scenario, 50 different test cases are generated with randomly initial and final positions selected in predefined areas. Therefore, MSD-IPM is run on 200 randomly generated test cases and the simulation results are illustrated in Fig. 4. According to the planning results, MSD-IPM can generate optimized trajectories without violating any constraints in all the simulations.

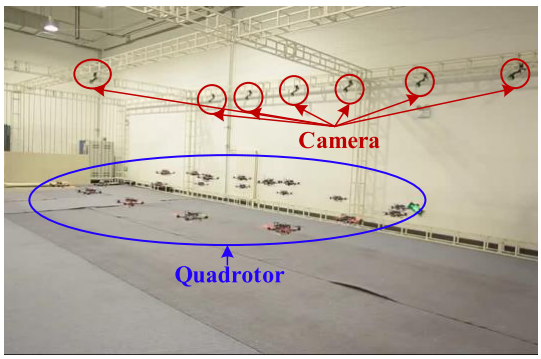
C. ALGORITHM RUNTIME COMPARISON

In this subsection, we analyze the efficiency of generating feasible and optimized trajectories by MSD-IPM based on the randomly generated cases shown in the robustness tests (section V-B). Comparative numeric simulations are conducted between MSD-IPM and a number of convex optimizers, i.e., MOSEK [39], SDPT3 [40], SeDuMi [41], and Mehrotra's IPM to verify the computational efficiency of the proposed method.

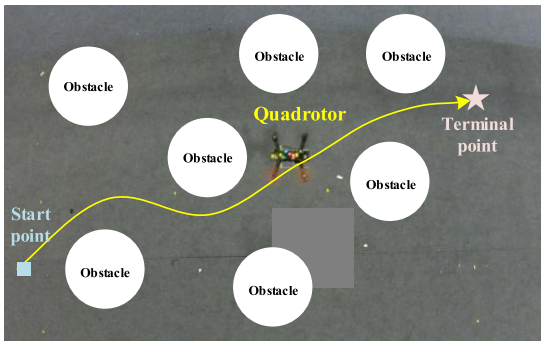
The runtime statistic results of finding feasible and optimized trajectories are illustrated in Figs. 5 and 6 respectively. For these four scenarios, the average runtime of generating feasible and optimized trajectories by MSD-IPM are always fewer than those by MOSEK, SDPT3, SeDuMi, and Mehrotra's IPM. For instance, simulations of 50 random cases on the scenario-I show that the runtime of generating feasible trajectories by MSD-IPM (0.06s) are generally 90.0%, 91.8%, 89.7%, and 79.9% lower than that of MOSEK (0.63s), SDPT3 (0.77s), SeDuMi (0.61s), and Mehrotra's IPM (0.313s) respectively. For generating optimized trajectories, MSD-IPM cost average 0.12s, 0.35s, 0.64s, and 0.99s for these four scenarios, which is approximately one-fifth of the runtime by Mehrotra's IPM (0.50s, 1.90s, 3.20s, 4.90s). More importantly, the comparison results of MSD-IPM and Mehrotra's IPM in simulations are consistent with the algorithm complexity analysis results, i.e., the algorithm complexity of

TABLE 6. Standard deviation of runtime for solving feasible/optimized trajectories.

Scenario	I	II	III	IV
MSD-IPM	**0.0055** (**0.0509**)	**0.0181** (**0.2097**)	**0.0269** (**0.4340**)	0.0455 (**0.3238**)
Mehrotra's IPM	0.0502 (0.1633)	0.0361 (1.1344)	0.0507 (1.9438)	0.0717 (2.6233)
SeDuMi	0.0703 (0.2990)	0.0584 (0.8729)	0.0348 (0.8018)	0.1079 (0.7480)
MOSEK	0.0425 (0.3232)	0.0229 (0.9771)	0.0358 (0.8070)	**0.0306** (0.8302)
SDPT3	0.0529 (0.4068)	0.0532 (1.3707)	0.0585 (1.2173)	0.0571 (1.1668)



(a)



(b)

FIGURE 7. Illustration of the indoor flight experiment testbed. (a) Illustration of the experiment testbed. (b) Illustration of the trajectory tracking experiment.

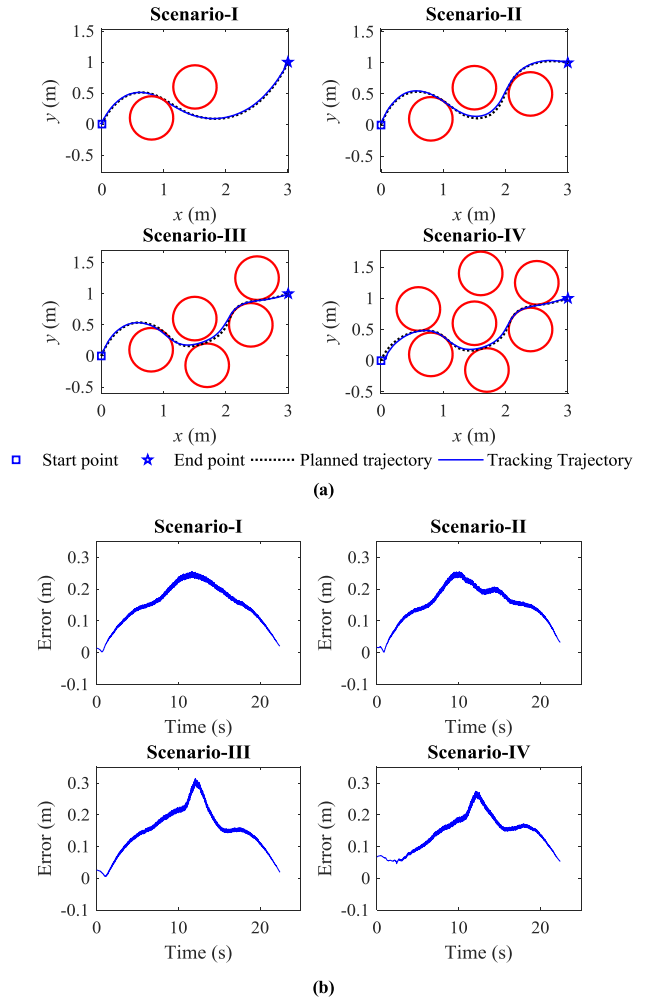


FIGURE 8. Feasible trajectory tracking result of different scenarios. (a) Tracking trajectory. (b) Tracking error.

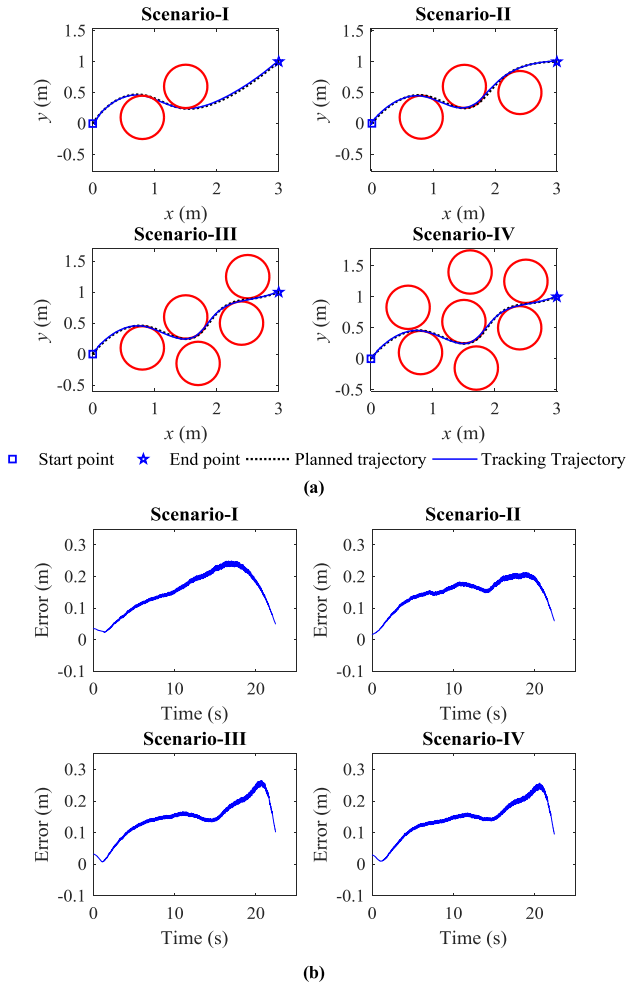


FIGURE 9. Optimized trajectory tracking result of different scenarios. (a) Tracking trajectory. (b) Tracking error.

MSD-IPM is about one order of magnitude lower than that of IPM.

Moreover, the generation of feasible trajectories (average runtime: 0.06s, 0.12s, 0.21s, and 0.37s) costs one-third runtime of generating optimized trajectories (average runtime: 0.12s, 0.36s, 0.65s, and 1.00s) in the four scenarios. Therefore, for some emergency situations, the feasible trajectories can be generated immediately and used to adapt the dynamic environments.

In addition, the standard deviations of the runtime for solving feasible and optimized trajectory are shown in Table 6, where the bold text represents the smallest standard deviation. The data in parentheses represent the standard deviations of the runtime for solving optimized trajectory. The standard deviation of runtime by MSD-IPM is always the best except for generating feasible trajectories in scenario-IV. The statistic results indicate that MSD-IPM is more robust than Mehrotra's IPM, SeDuMi, SDPT3, and MOSEK. Thus, MSD-IPM can provide appealing numerical stability for solving quadrotor trajectory planning problems.

VI. HARDWARE EXPERIMENTS

In this section, hardware experiments are conducted with physical quadrotors. We establish the indoor flight testbed in a rectangular area of approximately 6m×3m and 2.5m height shown in Fig. 7(a). The quadrotor utilized in the experiments is based on QAV260 frame equipped with PX4. The position and velocity information of quadrotors is measured by an OptiTrack motion capture system.

In the hardware experiments, the planned trajectories are uploaded to QAV260 before the quadrotor launching. In the trajectory tracking process, ground control center broadcasts the location messages to quadrotors via WIFI communications. Figure 7(b) shows a snapshot of the execution when the quadrotor heading for the end point. The supplemental video shows the full trajectory execution of scenario-IV.

The generated feasible and optimized trajectories in section V-A are tracked to guide the quadrotor to reach its destination safely. The tracking results for different scenarios are illustrated in Figs. 8(a) and 9(a) respectively. The planned trajectories almost coincide with the actual flight

trajectories. The trajectory tracking error profiles for feasible and optimized trajectories are shown in Fig. 8(b) and Fig. 9(b) respectively. The maximum trajectory tracking error is less than 30 cm, which is acceptable for practical applications.

VII. CONCLUSION

In this paper, MSD-IPM is developed for SCP-based quadrotor real-time trajectory planning. The optimal control problem of trajectory planning is translated into a series of convex quadratic programming subproblems via discretization and successive convexification. The efficient search direction solver is customized to improve the efficiency of convex quadratic programming. According to the algorithm complexity analysis results, the runtime of MSD-IPM is reduced by approximately one order of magnitude with respect to Mehrotra's IPM. The numerical simulation results demonstrate the effectiveness of MSD-IPM via the comparison with Mehrotra's IPM, SDPT3, SeDuMi, and MOSEK in several scenarios. And a number of random tests illustrate the numerical stability of the proposed method. Finally, the trajectory tracking experiments verify the feasibility of the trajectories planned by MSD-IPM for practical engineering applications. In future work, the proposed MSD-IPM will be extended to solve cooperative trajectory planning problems for quadrotor swarms.

REFERENCES

[1] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *J. Field Robot.*, vol. 29, no. 2, pp. 315–378, 2012.
[2] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robot. Autom. Syst.*, vol. 100, no. 2018, pp. 171–185, Feb. 2018.
[3] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *J. Intell. Robot. Syst.*, vol. 57, pp. 65–100, Jan. 2010.
[4] W. Dong, Y. Ding, J. Huang, X. Zhu, and H. Ding, "An efficient approach of time-optimal trajectory generation for the fully autonomous navigation of the quadrotor," *J. Dyn. Syst., Meas., Control*, vol. 139, no. 6, Apr. 2017, Art. no. 061012.
[5] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2013, pp. 3480–3486.
[6] M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 877–892, Aug. 2015.
[7] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme, "Downwash-aware trajectory planning for large quadrotor teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 250–257.
[8] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 856–869, Aug. 2018.
[9] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Feb. 2017.
[10] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 2872–2879.
[11] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Philadelphia, PA, USA: SIAM, 2010.
[12] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 1485–1491, Jan. 2011.
[13] X. Liu and P. Lu, "Solving nonconvex optimal control problems by convex optimization," *J. Guid. Control Dyn.*, vol. 37, no. 3, pp. 750–765, Feb. 2014.
[14] M. Turpin, N. Michael, and V. Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," *Auto. Robots*, vol. 33, nos. 1–2, pp. 143–156, 2012.
[15] W. Van Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2013, pp. 1788–1792.
[16] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2012, pp. 477–483.
[17] Z. Wang, G. Xu, L. Liu, and T. Long, "Obstacle-avoidance trajectory planning for attitude-constrained quadrotors using second-order cone programming," in *Proc. AIAA Aviat. Technol., Integr., Oper. Conf.*, Jun. 2018, pp. 1–10.
[18] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge Univ. Press, 2004.
[19] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, Sep. 2017.
[20] B. Açikmeşe and L. Blackmore, "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, vol. 47, no. 2, pp. 341–347, Feb. 2011.
[21] Y. Mao, M. Szmuk, and B. Açikmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2016, pp. 3636–3641.
[22] X. Liu, "Fuel-optimal rocket landing with aerodynamic controls," *J. Guid. Control Dyn.*, vol. 42, no. 1, pp. 65–77, Nov. 2018.
[23] E. D. Andersen, C. Roos, and T. Terlaky, "On implementing a primal-dual interior-point method for conic quadratic optimization," *Math. Program.*, vol. 95, no. 2, pp. 249–277, 2003.
[24] P. Lu and X. Liu, "Autonomous trajectory planning for rendezvous and proximity operations by conic optimization," *J. Guid., Control, Dyn.*, vol. 36, no. 2, pp. 375–389, 2013.
[25] X. Liu, Z. Shen, and P. Lu, "Entry trajectory optimization by second-order cone programming," *J. Guid. Control Dyn.*, vol. 39, no. 2, pp. 227–241, Aug. 2015.
[26] X. Liu, Z. Shen, and P. Lu, "Exact convex relaxation for optimal flight of aerodynamically controlled missiles," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 52, no. 4, pp. 1881–1892, Aug. 2016.
[27] U. Eren, D. Dueri, and B. Açikmeşe, "Constrained reachability and controllability sets for planetary precision landing via convex optimization," *J. Guid. Control Dyn.*, vol. 38, no. 11, pp. 2067–2083, Mar. 2015.
[28] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 1917–1922.
[29] D. Dueri, Y. Mao, Z. Mian, J. Ding, and B. Açikmeşe, "Trajectory optimization with inter-sample obstacle avoidance via successive convexification," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2017, pp. 1150–1156.
[30] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 5954–5961.
[31] M. Szmuk, C. A. Pascucci, D. Dueri, and B. Açikmeşe, "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4862–4868.
[32] D. Dueri, J. Zhang, and B. Açikmeşe, "Automated custom code generation for embedded, real-time second order cone programming," *IFAC Proc. Volumes*, vol. 47, no. 3, pp. 1605–1612, Aug. 2014.
[33] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, no. 1, pp. 1–27, 2012.
[34] D. Dueri, B. Açikmeşe, D. P. Scharf, and M. W. Harris, "Customized real-time interior-point methods for onboard powered-descent guidance," *J. Guid. Control Dyn.*, vol. 40, no. 2, pp. 197–212, Oct. 2016.
[35] T. R. Kruth, "Interior-point algorithms for quadratic programming," M.S. thesis, Inform. Math. Model., Tech. Univ. Denmark, Lyngby, Denmark, 2008.

[36] D. Morgan, S.-J. Chung, and F. Hadaegh, "Spacecraft swarm guidance using a sequence of decentralized convex optimizations," in *Proc. AIAA/AAS Astrodyn. Spec. Conf.*, Aug. 2012, pp. 1–16.

[37] J. Nocedal and S. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2006.

[38] X. Yan and V. H. Quintana, "An efficient predictor-corrector interior point algorithm for security-constrained economic dispatch," *IEEE Trans. Power Syst.*, vol. 12, no. 2, pp. 803–810, May 1997.

[39] *The MOSEK Optimization Toolbox for MATLAB Manual, Version 7.1 (Revision 35)*, MOSEK ApS, København, Copenhagen, Denmark, 2015.

[40] K. C. Toh, M. J. Todd, and R. H. Tütüncü, "SDPT3—A MATLAB software package for semidefinite programming, version 1.3," *Optim. Methods Softw.*, vol. 11, nos. 1–4, pp. 545–581, Jan. 2008.

[41] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, nos. 1–4, pp. 625–653, Jan. 2008.

GUANGTONG XU received the B.Sc. degree in flight vehicle design and engineering from the Beijing Institute of Technology, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree in aeronautical and astronautical science and technology. His research interests include path and trajectory planning, optimal control, and numerical optimization.

TENG LONG received the B.Sc. degree in flight vehicle engine engineering and the Ph.D. degree in flight vehicle design from the Beijing Institute of Technology, Beijing, China, in 2004 and 2009, respectively.

He is currently a Professor with the School of Aerospace Engineering, Beijing Institute of Technology. His research interests include multidisciplinary design optimization theories and applications for flight vehicles, cooperative control, and decision-making.

ZHU WANG received the B.Sc. degree in flight vehicle design and engineering and the Ph.D. degree in aeronautical and astronautical science and technology from the Beijing Institute of Technology, Beijing, China, in 2011 and 2017, respectively, where he is currently a Postdoctoral Researcher. His research interests include intelligent mission planning, numerical optimization, and multi-agent cooperation.

YAN CAO received the B.Sc. degree in flight vehicle design and engineering from the Beijing Institute of Technology, Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree in aeronautical and astronautical science and technology. His research interests include swarm intelligent and cooperative mission planning.