Real-Time Optimal Collision Avoidance for Multiple UAVs: An Embedded CCP-ADMM Approach

Jae-Jin Lee[a,b,1], Gyubin Park[a,b,1], Seungyeop Lee[a,b] and Jong-Han Kim[a,b,c,*]

[a]Department of Aerospace Engineering, Inha University, 100 Inha-ro, Michuhol-gu, Incheon 22212, Republic of Korea
[b]Program in Aerospace Systems Convergence, Inha University, 100 Inha-ro, Michuhol-gu, Incheon 22212, Republic of Korea
[c]Aerospace Systems Research Institute, Inha University, 100 Inha-ro, Michuhol-gu, Incheon 22212, Republic of Korea

ARTICLE INFO

Keywords:
Collision Avoidance
Optimal Trajectory Design
Convex-Concave Procedure
Alternating Direction Method of Multipliers

ABSTRACT

This paper presents a real-time trajectory optimization framework for multi-UAV collision avoidance designed for embedded implementation. Optimization-based planners can explicitly handle dynamics and safety constraints, but repeated onboard solves remain computationally demanding. To reduce this burden, we combine the convex-concave procedure (CCP) for non-convex collision avoidance constraints with a projection-oriented ADMM formulation based on reusable linear-system updates and closed-form radial and pairwise half-space projections. The proposed CCP-ADMM planner is benchmarked against ORCA and a MOSEK-based sequential convexification baseline in head-on and intersection scenarios, and further evaluated through K-scaling Monte Carlo tests using runtime, normalized control effort, constraint satisfaction, and success-rate metrics. Processor-in-the-loop simulations on NVIDIA Jetson Orin Nano boards validate embedded execution and replanning, while indoor quadrotor flight experiments demonstrate practical safety-distance maintenance on a physical testbed. The results indicate that the proposed framework provides a practical balance between trajectory-level constraint handling and embedded computational efficiency for multi-UAV collision avoidance.

1. Introduction

With the increasing deployment of unmanned aerial vehicles (UAVs) in various fields such as military, disaster response, logistics, and agriculture, multiple vehicles are often required to operate in shared airspace. Since loss of separation between these vehicles can undermine system safety and mission reliability, collision avoidance for multi-agent systems has become an important problem.

In the field of autonomous and unmanned vehicles, collision avoidance problems have traditionally been addressed using local reactive approaches.

A representative class of such methods is based on the velocity obstacle (VO) framework, which selects collision-free velocities by excluding relative velocities that would result in future collisions (Fiorini and Shiller, 1998; Bareiss and Van den Berg, 2013). Extensions of this framework, such as Optimal Reciprocal Collision Avoidance (ORCA) (Van den Berg et al., 2008; Van Den Berg et al., 2011), incorporate reciprocal collision avoidance by distributing responsibility among interacting agents. Other local reactive approaches include the dynamic window approach, which selects feasible control inputs within the admissible velocity space under dynamic constraints (Fox et al., 2002; Missura and Bennewitz, 2019).

These techniques are computationally efficient and relatively simple to implement, making them attractive for real-time applications. However, they primarily rely on local interaction geometry to determine collision-avoiding actions, with limited consideration of system evolution over time. As a result, it is difficult to enforce system dynamics and trajectory-level constraints within such frameworks.

More recently, optimal trajectory design for collision avoidance has gained significant attention. Approaches such as model predictive control (MPC) compute current control inputs by solving an optimization problem over a prediction horizon while explicitly incorporating system dynamics and constraints (Baca et al., 2018; Ji et al., 2016; Castillo-Lopez et al., 2018).

MPC–VO/ORCA frameworks, such as DCAD, incorporate vehicle dynamics into reciprocal velocity-space avoidance for decentralized multi-UAV replanning, and Swarm-CCO further extends this direction with uncertainty-aware probabilistic constraints (Arul and Manocha, 2020, 2021). In contrast to these reactive velocity-space approaches, this work enforces pairwise position-separation constraints directly over optimized finite-horizon state trajectories and exploits CCP–ADMM splitting with closed-form projections for predictable embedded computation.

Similarly, sequential convex programming (SCP) solves nonlinear trajectory optimization problems through repeated convex approximation processes (Augugliaro et al., 2012; Matiussi Ramalho et al., 2018; Niu et al., 2022). These methods provide a systematic framework for generating dynamically feasible and constraint-satisfying trajectories.

However, many optimization-based planners still require repeated numerical solves whose cost increases with the number of agents and time discretization nodes, making onboard implementation challenging in multi-agent settings. To reduce this burden, structured first-order methods such as alternating direction method of multipliers (ADMM)

*Corresponding author
jonghank@inha.ac.kr (J. Kim)
https://jonghank.github.io (J. Kim)
ORCID(s): 0009-0004-9732-1085 (J. Lee); 0009-0009-4243-4679 (G. Park); 0009-0001-4704-7805 (S. Lee); 0000-0002-9030-0490 (J. Kim)
[1]These authors contributed equally to this work.

have been used to split trajectory updates from constraint enforcement and to exploit reusable linear algebra and projection operations (Boyd and Vandenberghe, 2004; Boyd et al., 2011; Parikh et al., 2014). Recent ADMM-based trajectory planners also show that the choice of ADMM variant is closely tied to the problem formulation: Zheng et al. (Zheng et al., 2024) used over-relaxed ADMM for barrier-enhanced homotopic trajectory optimization, whereas Huang et al. (Huang et al., 2023) used dual consensus ADMM for decentralized cooperative planning over a communication graph. In contrast, the present work formulates multi-UAV collision avoidance as a projection-friendly two-block trajectory optimization problem designed for embedded implementation. Accordingly, we use standard ADMM for formulation simplicity and embedded predictability, rather than as a claim of superiority over over-relaxed or consensus ADMM.

Therefore, the remaining gap addressed in this paper is a finite-horizon multi-UAV planner that handles non-convex collision avoidance constraints at the trajectory level while retaining an implementation-oriented solver structure for resource-constrained embedded platforms.

In this study, we address this gap by developing a real-time CCP-ADMM trajectory planning framework for embedded multi-UAV collision avoidance. The non-convex collision avoidance constraints are handled by CCP, which yields affine half-space approximations over the planned trajectory. The resulting subproblems are solved by a projection-oriented ADMM splitting: the trajectory update is computed through reusable linear algebra, while the speed bounds and CCP-linearized collision avoidance constraints are handled through radial and half-space projection steps. This formulation reduces dependence on general-purpose constrained solvers and supports predictable embedded implementation.

The main contributions of this paper are summarized as follows.

- We propose a CCP-ADMM-based trajectory optimization method for multi-UAV collision avoidance, designed for embedded implementation.
- We reduce online computation using reusable linear-system updates, radial speed projections, and pairwise half-space corrections.
- We benchmark the proposed method against ORCA and CCP-MOSEK in head-on and intersection scenarios.
- We evaluate scalability through K-scaling Monte Carlo simulations using runtime, cost, feasibility, and success-rate metrics.
- We validate embedded receding-horizon replanning through processor-in-the-loop simulations (PILS) and indoor flight tests, confirming safety-distance maintenance and commanded speed-bound compliance.

2. Optimal Collision Avoidance Trajectory Planning

2.1. Problem Formulation for Multi-Agent Trajectory Planning

The problem of ensuring that multiple agents reach their target destinations within a designated time without colliding with one another can be formulated as shown in (1). Here, K denotes the total number of agents, T represents the number of discretization nodes, and Δt is the sampling interval for discrete time. For all variables in (1), the superscript indicates the agent index, and the subscript denotes the discrete time index. The agents are assumed to be multicopter-type vehicles operating in a 2D plane capable of generating control forces in arbitrary directions, which is commonly adopted in multi-UAV operations where altitude separation is used to reduce inter-agent collision risk. While the experiments in this work are conducted in a planar setting ($n_d = 2$), the formulation can be extended to an n_d-dimensional space ($n_d \in \{2, 3\}$) by replacing the position, velocity, and control vectors with their n_d-dimensional counterparts. The CCP linearization and ADMM update structure remain the same in form, with only dimension-dependent changes in the vector and matrix sizes. The computational analysis for extending the formulation to the 3D case is discussed in 2.4.

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{k=1}^{K} \sum_{t=0}^{T-1} \|u_t^{(k)}\|^2 \\
\text{subject to} \quad & p_{t+1}^{(k)} = p_t^{(k)} + v_t^{(k)} \Delta t, \\
& v_{t+1}^{(k)} = v_t^{(k)} + u_t^{(k)} \Delta t, \\
& p_T^{(k)} = p_{\text{des}}^{(k)}, \\
& v_T^{(k)} = v_{\text{des}}^{(k)}, \\
& v_{\text{lb}} \le \|v_t^{(k)}\| \le v_{\text{ub}}, \\
& \|p_t^{(k)} - p_t^{(l)}\| \ge d_{\text{safety}}, \quad \text{for all } l \ne k.
\end{aligned}
\tag{1}
$$

The objective function of problem (1) represents the minimization of the control effort. The constraints sequentially represent the discrete-time dynamics for the position $p_t^{(k)} \in \mathbb{R}^{n_d}$ and velocity $v_t^{(k)} \in \mathbb{R}^{n_d}$, the terminal conditions for the target position and velocity at the end of the mission, and the operational limits on the magnitude of the agent's velocity, $[v_{\text{lb}}, v_{\text{ub}}]$. Here, the speed limits define a direction-independent admissible velocity set, represented by an annulus in two-dimensional velocity space or a spherical shell in the three-dimensional case.

The final constraint corresponds to the inter-agent collision avoidance condition, requiring that the relative distance between any pair of agents (k, l) at each time step remains larger than the minimum safety distance d_{safety}, which is enforced for all pairwise agent combinations at each time step. It implies that the relative position vector of two agents must lie outside a circle (or a sphere in the three-dimensional case) of radius d_{safety}.

Since the feasible set defined by this condition is non-convex, the formulation becomes a non-convex optimization problem. Directly solving such a problem typically incurs a

high computational burden due to the complexity of finding a solution in a non-convex domain, making it challenging to implement on resource-constrained onboard processors for real-time applications. To address this implementation challenge, this paper proposes a real-time optimal trajectory design technique that applies the convex-concave procedure (CCP), leveraging the property that the collision avoidance constraints in (1) can be decomposed into convex and concave components.

2.2. Sequential Convexification using CCP

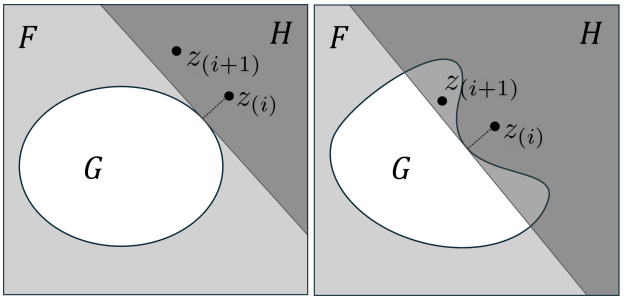
To handle the non-convexity of the collision avoidance constraints efficiently, we employ the Convex-Concave Procedure (CCP). CCP is an iterative optimization technique that searches for a solution by approximating a non-convex problem as a convex one at each step. While CCP shares structural similarities with Sequential Convex Programming (SCP), it distinguishes itself through its approximation methodology. SCP typically convexifies the entire problem using techniques such as Taylor series expansion or trust regions. In this process, even inherently convex components may be approximated, potentially leading to a loss of the original problem's structural information and requiring complex parameter tuning for convergence.

In contrast, CCP linearizes *only* the concave components of the non-convex constraints around the current solution, while preserving the convex components in their original form. This selective linearization minimizes approximation errors and ensures that the subproblems constructed at each iteration serve as a global lower approximation of the original problem. Crucially for real-time implementation, CCP does not require auxiliary stabilization techniques, such as trust regions or line search methods, due to the structure of the convex-concave decomposition.

Furthermore, when the initial trajectory is feasible and the convexified subproblems are constructed via inner approximations of the feasible set, CCP guarantees that subsequent iterates remain feasible with respect to the convexified constraints (Yuille and Rangarajan, 2003; Lanckriet and Sriperumbudur, 2009; Lipp and Boyd, 2016).

The geometric interpretation of this process is illustrated in Fig. 1. Let the feasible set be defined as $F = G^C$, where G denotes the infeasible region. Let $z_{(i)}$ denote the optimal solution at the i-th CCP iteration, and H represent the sublevel set obtained by linearly approximating F around $z_{(i)}$. Since the optimal solution at the $(i+1)$-th CCP iteration, $z_{(i+1)}$, is searched within H, if the constraint defining F is concave (as in Fig. 1(a)), then $H \subset F$. This implies that $z_{(i+1)}$ is obtained from a valid feasible region of the convexified problem. However, if F is a general non-convex set (Fig. 1(b)), it is possible that $H \not\subset F$, in which case $z_{(i+1)}$ may lie outside the feasible set F of the original problem.

In the considered problem, the collision avoidance constraint requires that the squared Euclidean distance between two vehicles remains greater than or equal to the squared safety distance d^2_{safety}. This constraint can be decomposed into



(a) Concave constraint ($z \in F$ with convex F^C)

(b) General non-convex constraint ($z \in F$ with non-convex F^C)

Figure 1: Affine approximation of non-convex constraint

a convex component and a concave component as follows:

$$\underbrace{-\|p_t^{(k)} - p_t^{(l)}\|^2}_{\text{concave}} + \underbrace{d^2_{\text{safety}}}_{\text{convex}} \leq 0.$$

To linearize the concave component, we define a reference trajectory $\bar{p}_t$ from the previous iteration. By modeling the system behavior around this reference trajectory with a small displacement δp_t, we have:

$$\|p_t^{(k)} - p_t^{(l)}\|^2 = \|\left(\bar{p}_t^{(k)} + \delta p_t^{(k)}\right) - \left(\bar{p}_t^{(l)} + \delta p_t^{(l)}\right)\|^2.$$

Expanding the above and neglecting higher-order terms (first-order Taylor approximation) yields the linearized collision avoidance constraint:

$$2\left(\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\right)^\top \left(p_t^{(k)} - p_t^{(l)}\right) \geq d^2_{\text{safety}} + \|\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\|^2. \quad (2)$$

Equation (2) represents affine half-space constraints for the pairwise collision avoidance condition at each time step. If the lower speed bound were also linearized, each CCP step could be formulated as a convex Quadratically Constrained Quadratic Program (QCQP), or an equivalent conic problem. However, repeatedly solving such subproblems with general-purpose interior-point solvers or software stacks such as `CVXPY` with MOSEK or Gurobi can be computationally expensive and less suitable for embedded deployment due to solver overhead and library dependencies.

In this work, the collision avoidance constraint is convexified through CCP, while the lower speed bound is retained as a norm constraint and handled in ADMM through a non-convex projection, as described in Section 2.3. Therefore, the proposed formulation is designed to be projection friendly. Since the transformed problem still involves the non-convex lower speed bound condition, a global feasibility guarantee or global optimality certificate with respect to the original problem is not claimed. Instead, convergence is monitored through the ADMM residuals and CCP cost history, and the final trajectory is explicitly checked against the original speed bounds and collision avoidance constraints. The reported numerical, PILS, and flight-test results empirically confirm constraint satisfaction in the tested cases.

2.3. Fast Optimization Solver via ADMM

ADMM is an optimization algorithm that combines the decomposability of dual ascent methods with the convergence properties of the method of multipliers. Ideally suited for problems with complex constraints, ADMM decomposes them into sequential updates of primal and dual variables. In the primal update step, optimization variables are updated by minimizing the augmented Lagrangian, while in the dual update step, Lagrange multipliers are updated based on constraint violations (Boyd et al., 2011). This iterative process is computationally attractive when the problem structure allows each update to be reduced to simple linear algebra or projection operations. Therefore, ADMM is well suited for embedded optimization problems in which repeatedly invoking a general-purpose constrained solver is computationally expensive.

In the present trajectory planning problem, ADMM is employed as a fast optimization solver because the CCP-based formulation admits a projection-friendly structure. For a given CCP iteration, the non-convex collision avoidance constraints are approximated by affine half-space constraints, while the system dynamics remain linear and the control-effort objective remains quadratic. The upper speed bound is handled by a radial Euclidean projection onto the ℓ_2-norm ball, whereas the lower speed bound is handled by an expansive radial projection onto the inner boundary of the admissible speed annulus. Together, these operations define a closed-form radial update for the admissible speed set. Rather than replacing the speed bound constraint with a component-wise box approximation to obtain a QP subproblem or introducing an additional CCP linearization for the lower speed bound, the proposed update retains the original admissible speed set and reduces the online velocity update to simple norm-scaling operations. The affine half-space constraints obtained from the CCP approximation are also enforced through closed-form half-space projections. Thus, the computational advantage of the proposed solver comes from exploiting closed-form projections and reusable linear algebra, rather than solving a generic constrained optimization problem at every iteration.

Although the expansive projection used for the lower speed bound is a practical and empirically useful operation, as also demonstrated in related first-order projection methods (Choi and Kim, 2025b,a; Choi et al., 2026; Kim et al., 2026; Park et al., 2024; Choi and Kim, 2024), it does not provide a global convergence certificate for the original non-convex trajectory planning problem. Accordingly, global optimality of the complete CCP-ADMM algorithm is not claimed; instead, its practical feasibility, constraint satisfaction, and real-time performance are evaluated through the numerical, PILS, and flight-test validations.

The ADMM formulation used at the $(i+1)$-th CCP iteration is presented below. For notational simplicity, the CCP iteration index is omitted.

$$\begin{aligned} \text{minimize} \quad & \|S^u\tilde{x}\|^2 + I_C(z) \\ \text{subject to} \quad & G\tilde{x} = h, \\ & S^z\tilde{x} = z. \end{aligned} \tag{3}$$

Throughout this subsection, the notation $(a_1, \dots, a_r) := [a_1^\top, \dots, a_r^\top]^\top$ denotes vertical stacking of vectors. For K agents in n_d spatial dimensions, the stacked state and control vectors are defined as

$$\begin{aligned} x_t &= (x_t^{(1)}, \dots, x_t^{(K)}) \in \mathbb{R}^{2n_dK}, \quad t = 0, \dots, T, \\ u_t &= (u_t^{(1)}, \dots, u_t^{(K)}) \in \mathbb{R}^{n_dK}, \quad t = 0, \dots, T-1, \\ x_t^{(k)} &= (p_t^{(k)}, v_t^{(k)}) \in \mathbb{R}^{2n_d}, \quad k = 1, \dots, K, \end{aligned}$$

where $p_t^{(k)}, v_t^{(k)} \in \mathbb{R}^{n_d}$ and $u_t^{(k)} \in \mathbb{R}^{n_d}$. Here, n_d denotes the spatial dimension.

Dynamics Constraint The first constraint, $G\tilde{x} = h$, encodes the double-integrator model discretized by the forward Euler method, together with the initial and terminal boundary conditions. For each agent $k = 1, \dots, K$, the resulting discrete-time model is $x_{t+1}^{(k)} = A_0 x_t^{(k)} + B_0 u_t^{(k)}$ where

$$A_0 = \begin{bmatrix} I_{n_d} & \Delta t I_{n_d} \\ 0_{n_d \times n_d} & I_{n_d} \end{bmatrix}, \quad B_0 = \begin{bmatrix} 0_{n_d \times n_d} \\ \Delta t I_{n_d} \end{bmatrix}.$$

For K agents, the stacked dynamics matrices are

$$A = I_K \otimes A_0 \in \mathbb{R}^{2n_dK \times 2n_dK}, \quad B = I_K \otimes B_0 \in \mathbb{R}^{2n_dK \times n_dK},$$

where $\otimes$ denotes the Kronecker product. The multi-agent dynamics are then expressed as $x_{t+1} = Ax_t + Bu_t$.

To simplify the notation, let $q = 2n_dK$ and $m = n_dK$, so that $x_t \in \mathbb{R}^q$ and $u_t \in \mathbb{R}^m$. For the block representation of $G\tilde{x} = h$, define the stage block

$$\xi_t = \begin{bmatrix} x_{t+1} \\ u_t \end{bmatrix} \in \mathbb{R}^{q+m}, \quad t = 0, \dots, T-1.$$

Since the initial state x_0 is given, it is excluded from the optimization variable, which is constructed as $\tilde{x} = (\xi_0, \dots, \xi_{T-1})$.

The stacked equality constraint has $G \in \mathbb{R}^{q(T+1)\times(q+m)T}$, $\tilde{x} \in \mathbb{R}^{(q+m)T}$, and $h \in \mathbb{R}^{q(T+1)}$, and is written in the following block-banded form:

$$\underbrace{\begin{bmatrix} \Phi & & & \\ \Psi & \Phi & & \\ & \ddots & \ddots & \\ & & \Psi & \Phi \\ & & & \Theta \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} \xi_0 \\ \xi_1 \\ \vdots \\ \xi_{T-1} \end{bmatrix}}_{\tilde{x}} = \underbrace{\begin{bmatrix} Ax_0 \\ 0_q \\ \vdots \\ 0_q \\ x_T^{\text{des}} \end{bmatrix}}_{h},$$

$$\Phi = \begin{bmatrix} I_q & -B \end{bmatrix}, \quad \Psi = \begin{bmatrix} -A & 0_{q\times m} \end{bmatrix}, \quad \Theta = \begin{bmatrix} I_q & 0_{q\times m} \end{bmatrix}.$$

The block-banded sparsity pattern of G is preserved for any spatial dimension n_d; extending the formulation from 2D to 3D only increases the size of each block, while the same block structure is maintained.

Selection Matrices The matrix S^u in the objective selects the control input components from $\tilde{x}$. The second constraint, $S^z\tilde{x} = z$, introduces split variables for the velocity and position components involved in the inequality constraints. Thus, S^u is used for the objective term, whereas S^v and S^p are used to construct the auxiliary variable z.

The position and velocity components are extracted using agent-wise block selection matrices. For a single agent, define $C_p = \begin{bmatrix} I_{n_d} & 0_{n_d\times n_d}\end{bmatrix}$, $C_v = \begin{bmatrix} 0_{n_d\times n_d} & I_{n_d}\end{bmatrix}$. The multi-agent extraction matrices are then constructed as $E_p = I_K \otimes C_p$, $E_v = I_K \otimes C_v$. Recall that the decision variable is ordered by the stage block $\xi_t = (x_{t+1}, u_t) \in \mathbb{R}^{q+m}$ for $t = 0, \dots, T-1$. Therefore, the stage-wise selection matrices are defined as $\bar{S}^p = \begin{bmatrix} E_p & 0_{m\times m}\end{bmatrix}$, $\bar{S}^v = \begin{bmatrix} E_v & 0_{m\times m}\end{bmatrix}$, $\bar{S}^u = \begin{bmatrix} 0_{m\times q} & I_m\end{bmatrix}$. Here, $\bar{S}^p\xi_t = p_{t+1}$, $\bar{S}^v\xi_t = v_{t+1}$, and $\bar{S}^u\xi_t = u_t$.

The full-horizon selection matrices are obtained by repeating these stage-wise selectors along the horizon: $S^p = I_T \otimes \bar{S}^p$, $S^v = I_T \otimes \bar{S}^v$, $S^u = I_T \otimes \bar{S}^u$. Thus, $S^p, S^v, S^u \in \mathbb{R}^{mT\times(q+m)T}$, and $S^p\tilde{x} = (p_1, \dots, p_T)$, $S^v\tilde{x} = (v_1, \dots, v_T)$, $S^u\tilde{x} = (u_0, \dots, u_{T-1})$. The auxiliary-variable selection matrix is then written as

$$S^z = \begin{bmatrix} S^v \\ S^p \end{bmatrix}, \quad z = \begin{bmatrix} z_v \\ z_p \end{bmatrix}.$$

where $S^z \in \mathbb{R}^{2mT\times(q+m)T}$ and $z_v, z_p \in \mathbb{R}^{mT}$. The auxiliary variable z represents copies of the velocity and position trajectories selected by S^v and S^p, respectively. The admissible sets imposed on these copies are defined below.

Constraint Sets for auxiliary Variables The indicator function $I_C(z)$ in (3) imposes the inequality constraints on the auxiliary variable z. Specifically, $I_C(z) = 0$ if $z \in \mathcal{C}$ and $I_C(z) = +\infty$ otherwise. Using the partition $z = (z_v, z_p)$ defined above, the feasible set is written as the product set $\mathcal{C} = \mathcal{C}^v \times \mathcal{C}^p$.

The velocity-related feasible set is defined as

$$\mathcal{C}^v = \prod_{t=1}^{T}\prod_{k=1}^{K}\left\{ z_{v,t}^{(k)} \;\middle|\; v_{\text{lb}} \le \|z_{v,t}^{(k)}\| \le v_{\text{ub}} \right\}.$$

Here, $z_{v,t}^{(k)} \in \mathbb{R}^{n_d}$ denotes the velocity copy of agent k at time step t. Let $\mathcal{P} = \{(k,l) \mid 1 \le k < l \le K\}$ denote the set of unordered agent pairs. The position-related feasible set is defined using the CCP-linearized collision avoidance constraints:

$$\mathcal{C}^p = \prod_{t=1}^{T}\left\{ z_{p,t} \;\middle|\; a_t^{(k,l)\top}\left(z_{p,t}^{(k)} - z_{p,t}^{(l)}\right) \ge b_t^{(k,l)},\ \forall (k,l)\in\mathcal{P} \right\}.$$

where $z_{p,t}^{(k)} \in \mathbb{R}^{n_d}$ denotes the position copy of agent k at time step t. The coefficients of the affine half-space constraints are computed from the reference trajectory of the previous CCP iteration as

$$a_t^{(k,l)} = 2\left(\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\right), \quad b_t^{(k,l)} = d_{\text{safety}}^2 + \|\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\|^2.$$

Thus, the original collision avoidance constraints are replaced by affine constraints for all agent pairs at each time step. The corresponding projection operations onto $\mathcal{C}^v$ and $\mathcal{C}^p$ are described in the z-update step.

The augmented Lagrangian is defined as follows, with scaled dual variables w_1, w_2 and penalty parameter $\rho > 0$:

$$\mathcal{L}_\rho(\tilde{x}, z, w) = \|S^u\tilde{x}\|^2 + I_C(z) + \frac{\rho}{2}\left(\|G\tilde{x} - h + w_1\|^2 + \|S^z\tilde{x} - z + w_2\|^2\right).$$

The ADMM algorithm iterates through the following three steps. The superscript $[j]$ denotes the ADMM iteration index, which is distinguished from the agent and CCP iteration index omitted in this subsection.

2.3.1. $\tilde{x}$-update step (Global Planning):

The $\tilde{x}$-update can be interpreted as the global planning step:

$$\tilde{x}^{[j+1]} = \underset{\tilde{x}}{\text{argmin}}\|S^u\tilde{x}\|^2 + \frac{\rho}{2}\big(\|G\tilde{x} - h + w_1^{[j]}\|^2 + \|S^z\tilde{x} - z^{[j]} + w_2^{[j]}\|^2\big).$$

This is an unconstrained quadratic minimization problem, whose solution is obtained by solving a linear system. Since the coefficient matrix is fixed for a given horizon length and penalty parameter ρ, its sparse factorization is precomputed and reused across ADMM iterations. With the block-banded temporal structure and agent-wise sparsity exploited, the online $\tilde{x}$-update is reduced to forward/backward substitutions or sparse matrix-vector operations, scaling as $\mathcal{O}(KT)$ for fixed spatial dimension.

2.3.2. z-update step (Local Projection):

The z-update as the local projection step is given by:

$$\begin{aligned} z^{[j+1]} &= \underset{z}{\text{argmin}}\big(I_C(z) + \frac{\rho}{2}\|S^z\tilde{x}^{[j+1]} - z + w_2^{[j]}\|^2\big) \\ &= \Pi_C\big(S^z\tilde{x}^{[j+1]} + w_2^{[j]}\big). \end{aligned}$$

This step updates the auxiliary variable by enforcing the velocity and collision-avoidance constraints through projection operations. Let

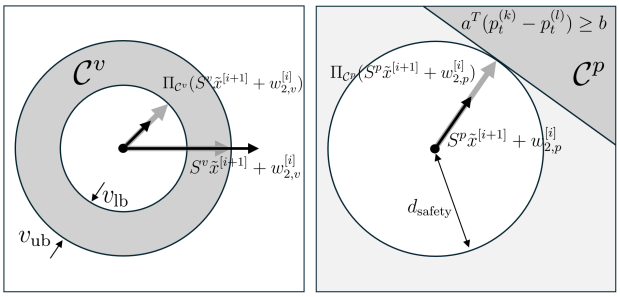
$$y^{[j+1]} = S^z\tilde{x}^{[j+1]} + w_2^{[j]} = \begin{bmatrix} y_v^{[j+1]} \\ y_p^{[j+1]} \end{bmatrix}.$$

Since $\mathcal{C} = \mathcal{C}^v \times \mathcal{C}^p$, the z-update is implemented as

$$z_v^{[j+1]} = \Pi_{\mathcal{C}^v}\left(y_v^{[j+1]}\right), \quad z_p^{[j+1]} = \widehat{\Pi}_{\mathcal{C}^p}\left(y_p^{[j+1]}\right).$$

where $\Pi_{\mathcal{C}^v}$ denotes the closed-form radial projection onto the admissible speed set, and $\widehat{\Pi}_{\mathcal{C}^p}$ denotes the pairwise half-space projection-sweep used for collision avoidance. The hat notation emphasizes that the implementation avoids solving the full coupled projection problem over the intersection of all pairwise half-spaces.

For velocity constraints ($\mathcal{C}^v$), the projection is computed analytically by radial scaling while preserving the direction



(a) Radial projection onto $\mathcal{C}^v$ (b) Pairwise half-space correction

Figure 2: Geometric interpretation of the z-update.

(Fig. 2(a)).

$$z_{v,t}^{(k)} = \Pi_{\mathcal{C}^v}\left(y_{v,t}^{(k)}\right) = \begin{cases} v_{\mathrm{ub}} \dfrac{y_{v,t}^{(k)}}{\|y_{v,t}^{(k)}\|}, & \text{if } \|y_{v,t}^{(k)}\| > v_{\mathrm{ub}}, \\ v_{\mathrm{lb}} \dfrac{y_{v,t}^{(k)}}{\|y_{v,t}^{(k)}\|}, & \text{if } 0 < \|y_{v,t}^{(k)}\| < v_{\mathrm{lb}}, \\ y_{v,t}^{(k)}, & \text{otherwise.} \end{cases}$$

Both the upper- and lower-bound cases have the same radial-scaling form: they preserve the direction of $y_{v,t}^{(k)}$ and modify only its magnitude. When $\|y_{v,t}^{(k)}\| > v_{\mathrm{ub}}$, the vector is projected onto the outer boundary of the admissible speed annulus by scaling its magnitude down to v_{ub}. When $0 < \|y_{v,t}^{(k)}\| < v_{\mathrm{lb}}$, the same radial operation is applied in the outward direction, scaling the magnitude up to v_{lb}. In this sense, the lower-bound update is an expansive radial projection, i.e., the outward counterpart of the usual upper-bound norm-clipping operation.

For collision avoidance ($\mathcal{C}^p$), the position part of the z-update uses pairwise half-space corrections (Fig. 2(b)).

For more than two agents, multiple pairwise collision avoidance constraints may become active simultaneously and may share common agents. The proposed projection-sweep computes a pairwise correction independently for each active pair using the same current auxiliary positions. Each correction is first obtained in the relative-position space and then split between the two agents involved in that pair. If an agent participates in multiple active pairs, the corresponding correction terms are summed agent-wise, and the accumulated correction is applied once when updating z_p. This handles multi-agent coupling through agent-wise accumulated pair corrections, avoiding a coupled projection problem over the full intersection of pairwise half-spaces at every ADMM iteration.

For a pair $(k, l) \in \mathcal{P}$, the pairwise half-space projection problem is formulated as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|r - \eta\|^2 \\ \text{subject to} \quad & a^\top r \geq b, \end{aligned}$$

where $\eta = y_{p,t}^{(k)} - y_{p,t}^{(l)}$ is the relative position before projection, and r is the projected relative position. For compact notation, $a = a_t^{(k,l)}$ and $b = b_t^{(k,l)}$ are used here. When the reference relative position is nearly zero, the half-space normal becomes ill-defined. In implementation, if $\|\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\| < \epsilon$, a small deterministic regularization is applied to define a valid separating direction before constructing the CCP half-space.

The optimal solution is obtained in closed form from the KKT conditions:

$$r^* = \begin{cases} \eta + \dfrac{b - a^\top \eta}{\|a\|^2} a, & \text{if } a^\top \eta < b, \\ \eta, & \text{otherwise.} \end{cases}$$

Define the relative-position correction associated with pair (k, l) as

$$\delta_t^{(k,l)} = \begin{cases} \dfrac{b - a^\top \eta}{\|a\|^2} a, & \text{if } a^\top \eta < b, \\ 0, & \text{otherwise.} \end{cases}$$

For each time step t, the accumulated correction for each agent is initialized as $c_t^{(k)} = 0$. For every active collision avoidance constraint, the relative correction is distributed symmetrically as

$$c_t^{(k)} \leftarrow c_t^{(k)} + \frac{1}{2}\delta_t^{(k,l)}, \quad c_t^{(l)} \leftarrow c_t^{(l)} - \frac{1}{2}\delta_t^{(k,l)}.$$

After all active collision avoidance constraints at time step t are evaluated, the position auxiliary variables are updated by $z_{p,t}^{(k)} = y_{p,t}^{(k)} + c_t^{(k)}, \quad k = 1, \ldots, K$. For a single collision avoidance constraint, this symmetric distribution exactly realizes the relative-position projection, because $z_{p,t}^{(k)} - z_{p,t}^{(l)} = \eta + \delta_t^{(k,l)} = r^*$. The factor $1/2$ gives the minimum-norm correction when the two agents are weighted equally. Specifically, if d_k and d_l denote the position corrections applied to agents k and l, respectively, then

$$\text{minimize}_{d_k, d_l} \ \frac{1}{2}\|d_k\|^2 + \frac{1}{2}\|d_l\|^2 \quad \text{subject to } d_k - d_l = \delta_t^{(k,l)}$$

yields $d_k = \frac{1}{2}\delta_t^{(k,l)}$ and $d_l = -\frac{1}{2}\delta_t^{(k,l)}$. When multiple active constraints share common agents, these pairwise corrections are accumulated agent-wise, which corresponds to the projection-sweep approximation $\widehat{\Pi}_{\mathcal{C}^p}$ used in the implementation.

The velocity projection and pairwise collision-avoidance corrections are computed in closed form. Therefore, the z-update avoids inner numerical optimization loops and is implemented using only norm computations, dot products, and vector additions. For each ADMM iteration, the velocity part requires KT radial scaling operations, while the collision-avoidance part evaluates at most $|\mathcal{P}|T = K(K-1)T/2$ pairwise half-space corrections. Thus, for a fixed spatial dimension n_d, the worst-case operation count of the z-update is $\mathcal{O}(K^2T)$. The projection-sweep is not used as a one-shot

feasibility certificate for the full intersection of all pairwise half-spaces; rather, it is repeatedly applied within the ADMM iterations as an efficient correction operator for the CCP-linearized collision-avoidance constraints. Together with the reusable linear algebra in the $\tilde{x}$-update, this closed-form z-update contributes to the online computational efficiency of the proposed CCP-ADMM solver.

2.3.3. *w*-update step (Dual Update):

The last step is the dual update given by:

$$w_1^{[j+1]} = w_1^{[j]} + G\tilde{x}^{[j+1]} - h,$$
$$w_2^{[j+1]} = w_2^{[j]} + S^z\tilde{x}^{[j+1]} - z^{[j+1]}$$

As shown above, this step consists solely of simple element-wise vector additions and subtractions. Its operation count scales as $\mathcal{O}(KT)$, making it a lower-order term compared with the pairwise collision-avoidance correction in the z-update.

2.3.4. Stopping Criteria and Feasibility Check:

For a fixed CCP-linearized subproblem, the inner ADMM iterations are terminated when the primal and dual residuals fall below the prescribed tolerances ϵ_{pri} and ϵ_{dual}, or when the maximum number of ADMM iterations is reached. The residuals are defined as

$$r^{[j+1]} = \begin{bmatrix} G\tilde{x}^{[j+1]} - h \\ S^z\tilde{x}^{[j+1]} - z^{[j+1]} \end{bmatrix},$$

$$s^{[j+1]} = \rho(S^z)^\top \left(z^{[j+1]} - z^{[j]}\right).$$

The primal residual measures the violation of the dynamics constraint and the consistency condition $S^z\tilde{x} = z$, whereas the dual residual measures the change in the auxiliary-variable update. Therefore, small ADMM residuals indicate convergence of the split formulation for the current CCP-linearized subproblem.

However, the ADMM residuals alone do not certify feasibility with respect to the original non-convex velocity and collision-avoidance constraints. After each outer CCP iteration, the obtained trajectory is evaluated against the original constraints. In particular, the maximum pairwise collision violation is computed as

$$e_{\text{ca}}^{(i)} = \max_{t,(k,l)\in\mathcal{P}} \left[d_{\text{safety}} - \|p_t^{(k,i)} - p_t^{(l,i)}\|\right]_+,$$

and the speed bound violation is computed as

$$e_{\text{v}}^{(i)} = \max_{t,k} \left\{ \left[v_{\text{lb}} - \|v_t^{(k,i)}\|\right]_+, \left[\|v_t^{(k,i)}\| - v_{\text{ub}}\right]_+ \right\},$$

where $[\alpha]_+ = \max(\alpha, 0)$. The stabilization of the outer CCP loop is also monitored using the relative objective change, $\Delta_J^{(i)} = |J^{(i)} - J^{(i-1)}| / \max\{1, |J^{(i-1)}|\}$. Thus, the termination condition is evaluated using the inner ADMM primal and dual residuals, the original-constraint violations $e_{\text{ca}}^{(i)}$ and $e_{\text{v}}^{(i)}$, and the relative objective change $\Delta_J^{(i)}$. The outer CCP loop is terminated when these quantities fall below their prescribed tolerances, or when the maximum number of CCP iterations is reached. When a fixed iteration schedule is used, these quantities are still monitored, and the final trajectory is explicitly checked against the original velocity and collision avoidance constraints.

2.4. Computational Complexity Analysis

The following analysis focuses on the online operation count per ADMM iteration for a fixed CCP-linearized subproblem. The total online solve time additionally depends on the number of outer CCP iterations and the number of ADMM iterations used for each CCP subproblem.

One of the primary contributions of this work is the reduction of online computational burden by exploiting the structure of the trajectory optimization problem compared to conventional optimization methods. Standard trajectory planning approaches often rely on Interior-Point Methods (IPM) to solve convexified trajectory optimization subproblems. For a problem with N_{var} decision variables, the complexity of a generic dense IPM is typically dominated by the solution of a coupled KKT linear system at each iteration, scaling as $\mathcal{O}(N_{\text{var}}^3)$. In our formulation, the number of variables is proportional to the spatial dimension, the number of agents, and the number of time steps, i.e., $N_{\text{var}} \propto n_d KT$. Consequently, if the problem structure is not exploited, the dense reference complexity scales as $\mathcal{O}((n_d KT)^3)$, which reduces to $\mathcal{O}(K^3T^3)$ for a fixed spatial dimension. This severely limits scalability and real-time performance on embedded processors.

In contrast, the proposed CCP-ADMM framework leverages the temporal block-banded structure, agent-wise sparsity, and closed-form projection updates. For a fixed spatial dimension n_d, the online operation count of each ADMM iteration is summarized as follows:

1. *Offline Precomputation:* For a fixed horizon length and penalty parameter ρ, the coefficient matrix in the $\tilde{x}$-update step is constant across ADMM iterations. Its sparse factorization can therefore be precomputed and reused in the online loop. If several values of ρ are used in different planning modes, the corresponding factorizations can be precomputed separately.
2. *$\tilde{x}$-update:* In real-time, this step involves only forward and backward substitutions using the precomputed factors. When the block-banded temporal structure and agent-wise sparsity are exploited, the online $\tilde{x}$-update scales linearly with the horizon length and the number of agents, i.e., $\mathcal{O}(KT)$. This complexity refers to the implemented sparse factor-reuse structure; using generic dense linear algebra would lead to a higher cost.
3. *z-update:* The velocity projection is separable across agents and time steps, requiring KT radial scaling operations. The collision-avoidance update is performed through pairwise half-space corrections over all unordered agent pairs. Since there are $|\mathcal{P}| = K(K-1)/2$ pairs at each time step, the pairwise correction sweep scales as $\mathcal{O}(|\mathcal{P}|T) = \mathcal{O}(K^2T)$. Each

pairwise correction requires only dot products, norm computations, and vector additions.

4. *Dual update:* This step involves simple vector additions and subtractions, scaling linearly as $\mathcal{O}(KT)$.

The effect of extending the formulation from the planar case to the three-dimensional case can be made explicit by retaining n_d in the operation count. Each radial velocity projection and each pairwise half-space correction is performed in $\mathbb{R}^{n_d}$. Hence, the velocity projection scales as $\mathcal{O}(n_d KT)$, the pairwise collision-avoidance correction sweep scales as $\mathcal{O}(n_d K^2 T)$, and the remaining vector operations scale as $\mathcal{O}(n_d KT)$. Therefore, increasing the planning space from $n_d = 2$ to $n_d = 3$ introduces only a dimension-dependent constant factor and does not change the asymptotic dependence on K or T. In particular, real-time feasibility is affected more strongly by the number of agents, the horizon length, and the prescribed CCP/ADMM iteration schedule than by the change from 2D to 3D alone.

Therefore, under the implemented sparse factor reuse and pairwise projection-sweep structure, the per-ADMM-iteration complexity is dominated by the collision-avoidance part of the z-update,

$$\mathcal{O}(n_d KT) + \mathcal{O}(n_d K^2 T) + \mathcal{O}(n_d KT) = \mathcal{O}(n_d K^2 T).$$

For a fixed spatial dimension, this reduces to $\mathcal{O}(K^2 T)$. Thus, the proposed method scales linearly with the time horizon and quadratically with the number of agents, while the extension from 2D to 3D appears only as a constant-factor increase in the operation count. This is substantially lower than the dense reference complexity $\mathcal{O}((n_d KT)^3)$ of a generic centralized IPM that does not exploit the problem structure.

If N_{CCP} denotes the number of CCP iterations and $N_{\mathrm{ADMM}}^{(i)}$ denotes the number of ADMM iterations used at the i-th CCP iteration, the total online operation count scales as

$$\mathcal{O}\left(n_d K^2 T \sum_{i=1}^{N_{\mathrm{CCP}}} N_{\mathrm{ADMM}}^{(i)}\right),$$

up to lower-order terms such as warm-starting, residual evaluation, and bookkeeping. When fixed iteration schedules are used, the summation term acts as a constant multiplier for a given solver configuration.

3. Numerical Validation and Experimental Results

3.1. Numerical Benchmark and Convergence Verification

The numerical simulations were executed on a standard desktop workstation (Intel Core i5-13500, 32 GB RAM) using Python 3.9. This subsection considers two canonical collision avoidance scenarios: a head-on encounter with two agents and an intersection scenario with three agents. For each scenario, trajectory, velocity, and relative distance profiles are reported for the proposed CCP-ADMM method and representative baselines. The convergence behavior of CCP-ADMM is then examined using the primal and dual residuals and the CCP cost history. Finally, a K-scaling Monte Carlo analysis is conducted to evaluate runtime scalability and constraint satisfaction as the number of agents increases.

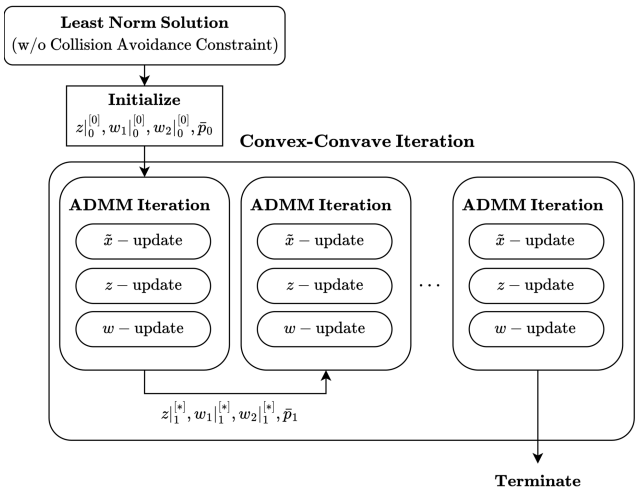


Figure 3: Schematic of the proposed CCP-ADMM framework.

Across all numerical tests, CCP-ADMM is executed with a fixed number of outer CCP iterations and a linearly decreasing inner ADMM iteration schedule. More ADMM iterations are assigned to early CCP steps because the reference trajectory is initialized from the least-norm solution and the linearization error can be relatively large. As the reference trajectory stabilizes, the inner ADMM iteration count is reduced to avoid unnecessary computation. In the K-scaling Monte Carlo study, the same iteration schedule is used for all values of K and all random seeds, so that the runtime statistics reflect the scaling effect of the number of agents rather than changes in the solver configuration.

The workflow of the proposed CCP-ADMM framework is summarized in Figure 3. In the figure notation, $q|_m^{[n]}$, where $q \in \{z, w_1, w_2\}$, denotes the value of q at ADMM iteration n within CCP iteration m. The symbol [0] denotes the initial value of the inner ADMM loop. The symbol $[*]$ denotes the value returned when the inner ADMM loop terminates, either after satisfying the stopping criteria or after reaching the prescribed iteration limit. The reference trajectory $\bar{p}_m$ defines the CCP linearization at CCP iteration m. The initial reference trajectory $\bar{p}_0$ is obtained from the least-norm solution computed without collision avoidance constraints, and for $m \geq 1$, $\bar{p}_m$ is obtained from the position trajectory returned by the previous CCP iteration.

3.1.1. Benchmark Methods and Implementation Settings

The numerical benchmark compares the proposed CCP-ADMM method with two baseline methods: Optimal Reciprocal Collision Avoidance (ORCA) and CCP-MOSEK. ORCA is selected as a fast local reactive collision avoidance baseline, while CCP-MOSEK is selected as a solver-based sequential convexification baseline. These allow the proposed method

to be compared with both a reactive velocity method and a trajectory optimization method.

The ORCA baseline is implemented using the `pyrvo` simulator. At each ORCA update instant, agent k computes a preferred velocity toward its goal,

$$v_{\mathrm{pref},t}^{(k)} = v_{\mathrm{cr}}^{(k)} \frac{p_{\mathrm{des}}^{(k)} - p_t^{(k)}}{\|p_{\mathrm{des}}^{(k)} - p_t^{(k)}\|},$$

where $v_{\mathrm{cr}}^{(k)}$ is the prescribed cruise speed used to define the preferred ORCA velocity. The `pyrvo` solver then returns a collision avoiding velocity command by enforcing reciprocal collision avoidance constraints. The agent radius is set to $d_{\mathrm{safety}}/2$, so that the combined radius of two agents corresponds to the prescribed safety distance.

Since ORCA directly generates a velocity command, a command shaping step is applied before propagating the acceleration input model. The ORCA velocity command is passed through a first order command filter and a command update limit to suppress abrupt changes caused by the discrete reactive update. The shaped velocity command is then converted into an acceleration input through proportional velocity tracking, and the state is propagated using this acceleration input.

The CCP-MOSEK baseline is implemented in `CVXPY` and solved using `MOSEK`. Rather than solving the original non-convex problem in (1) directly, CCP-MOSEK solves a sequence of convexified problems around a reference trajectory. The dynamics, initial and terminal conditions, and upper speed bound are kept in the same form as in (1). The objective is augmented with slack and trust region penalties as

$$\begin{aligned} J_{\mathrm{MOSEK}} &= J_u + w_s J_s + w_\eta J_\eta, \\ J_u &= \textstyle\sum_{k=1}^{K} \sum_{t=0}^{T-1} \|u_t^{(k)}\|^2, \\ J_s &= \textstyle\sum_{k,t} (s_{k,t}^{v})^2 + \sum_{k<l,t} (s_{k,l,t}^{\mathrm{col}})^2, \\ J_\eta &= \textstyle\sum_{k,t} \eta_{k,t}^2. \end{aligned}$$

The non-convex lower speed bound is handled by linearizing $\|v_t^{(k)}\|^2$ around the reference velocity $\bar{v}_t^{(k)}$:

$$2(\bar{v}_t^{(k)})^\top v_t^{(k)} \geq v_{\mathrm{lb}}^2 + \|\bar{v}_t^{(k)}\|^2 - s_{k,t}^{v}.$$

The non-convex collision avoidance constraint is handled using the same CCP linearization as in (2), with an additional slack variable:

$$\begin{aligned} &2\big(\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\big)^\top \big(p_t^{(k)} - p_t^{(l)}\big) \\ &\quad \geq d_{\mathrm{safety}}^2 + \|\bar{p}_t^{(k)} - \bar{p}_t^{(l)}\|^2 - s_{k,l,t}^{\mathrm{col}}. \end{aligned}$$

A soft trust region is imposed as

$$\begin{aligned} &\|x_{t+1}^{(k)} - \bar{x}_{t+1}^{(k)}\|^2 + \|u_t^{(k)} - \bar{u}_t^{(k)}\|^2 \leq \eta_{k,t}, \\ &s_{k,t}^{v} \geq 0, \quad s_{k,l,t}^{\mathrm{col}} \geq 0, \quad \eta_{k,t} \geq 0. \end{aligned}$$

The slack variables $s_{k,t}^{v}$ and $s_{k,l,t}^{\mathrm{col}}$ are introduced to prevent the convexified subproblem from becoming infeasible when the current linearization is too restrictive. They are penalized by w_s, so violations are discouraged while numerical feasibility is preserved during the CCP iterations. The trust region variable $\eta_{k,t}$ limits large deviations from the reference trajectory so that the first order approximation remains reliable, and its relaxation is penalized by w_η. The CCP-MOSEK loop is terminated when the relative objective change and the maximum slack values fall below prescribed tolerances for consecutive iterations.

Table 1
Simulation parameters (Scenario S-1).

Parameter	Agent #1	Agent #2	Unit
Initial & Terminal Conditions			
p_0	(0, 0)	(100, 0)	m
p_{des}	(100, 0)	(0, 0)	m
v_0	(12, 0)	(-12, 0)	m/s
v_{des}	(12, 0)	(-12, 0)	m/s
Constraints & Solver Settings			
d_{safety}	25		m
$(v_{\mathrm{lb}}, v_{\mathrm{ub}})$	(10, 15)		m/s
T	40		-
Δt	0.25		s

Table 2
Benchmark method settings for Scenario S-1.

Method	Settings
ORCA	$v_{\mathrm{cr}} = 12\mathrm{m/s}$, $r_{\mathrm{agent}} = d_{\mathrm{safety}}/2$, $d_{\mathrm{neighbor}} = 150\mathrm{m}$, $N_{\mathrm{neighbor}} = 2$, $K_v = 4$.
CCP-MOSEK	$w_s = 10$, $w_\eta = 1.0$.
CCP-ADMM	$\rho = 8$.

For the proposed CCP-ADMM method, the ADMM penalty parameter ρ is fixed within each reported scenario. For the head-on and intersection scenarios, ρ is selected through preliminary calibration to obtain stable primal and dual residual reduction within the prescribed iteration schedule while satisfying the original velocity and collision avoidance constraints. For the K-scaling Monte Carlo analysis, ρ is calibrated separately for each number of agents. Specifically, several calibration seeds are first tested by sweeping candidate ρ values, the value with the highest constraint satisfaction rate is selected for each calibration seed, and the average of the selected values is then used as a fixed ρ for new Monte Carlo test seeds.

3.1.2. Scenario S-1: Head on Encounter

The first scenario simulates a head-on encounter between two vehicles. This configuration represents a worst-case geometry where the relative velocity is maximized, and the optimal avoidance direction is singular. Detailed parameters are listed in Table 1 and Table 2. The algorithm was configured with 10 CCP iterations, and the ADMM iterations were linearly reduced from 300 to 100.

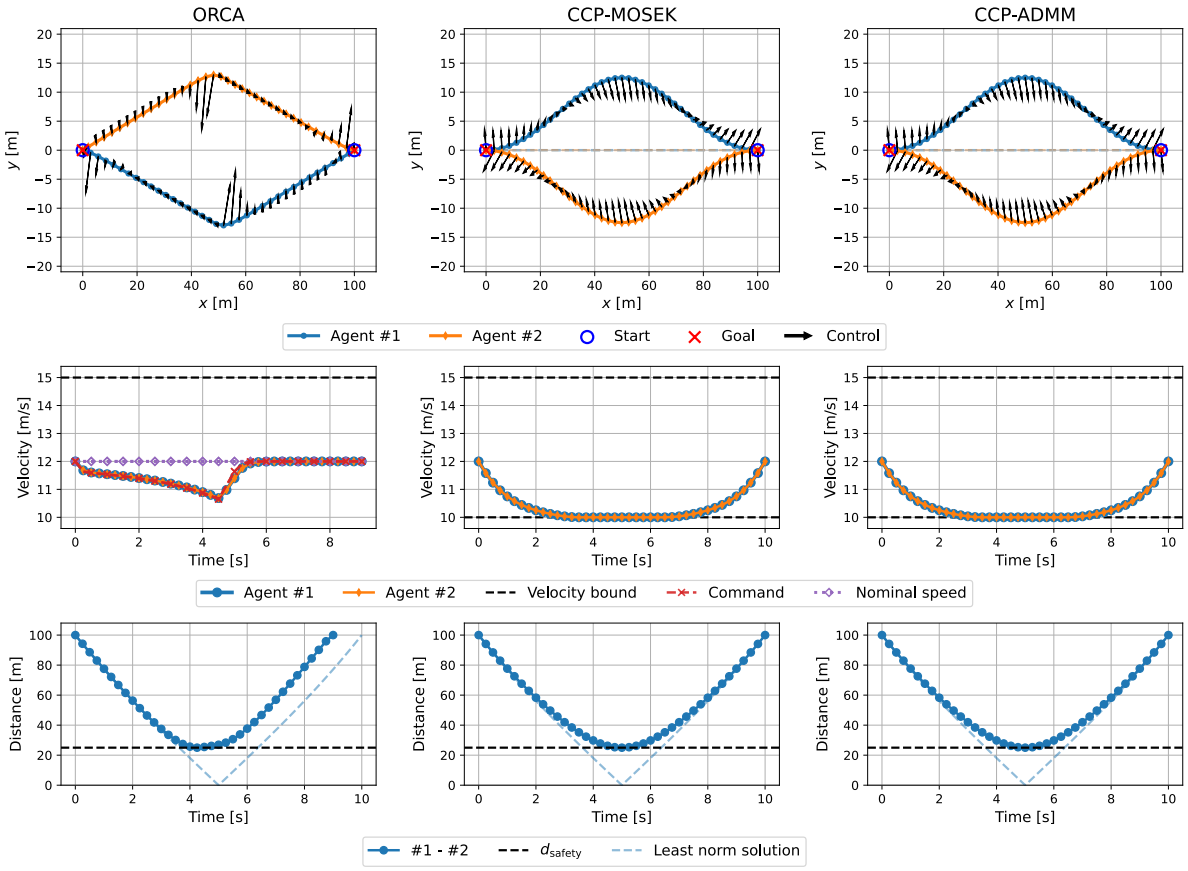


Figure 4: Trajectories, velocity profiles, and relative distances for the benchmarks and the proposed method (Scenario S-1).

Figure 4 compares ORCA, CCP-MOSEK, and CCP-ADMM in terms of trajectory, speed, and relative distance. All three methods maintain the required safety distance throughout the maneuver.

ORCA computes a collision avoiding velocity command at each update instant from the current reciprocal velocity constraints and the preferred velocity, whereas CCP-MOSEK and CCP-ADMM optimize the state and input trajectories over the prescribed planning horizon. Although ORCA maintains the required safety distance in this scenario, it reaches the goal earlier than the fixed 10 s horizon because the terminal state is not explicitly imposed as a horizon constraint. Therefore, the ORCA result depends on the prescribed cruise speed and timing settings, while the CCP-based methods satisfy the terminal position and velocity constraints within the specified horizon.

In contrast, CCP-MOSEK and CCP-ADMM generate smooth and nearly symmetric avoidance trajectories by optimizing the state and input trajectories over the horizon. The control vectors are distributed over a broader portion of the maneuver, and the velocity profiles gradually decrease toward the lower speed bound before returning to the desired terminal speed. This behavior indicates that the trajectory optimization methods coordinate lateral deviation, speed variation, and terminal constraints over the horizon while maintaining the required separation.

Table 3
Simulation parameters (Scenario S-2).

Parameter	Agent #1	Agent #2	Agent #3	Unit
Initial & Terminal Conditions				
p_0	(0, 150)	(0, 0)	(175, 75)	m
p_{des}	(150, 0)	(150, 150)	(-25, 75)	m
v_0	(10, 0)	(10, 0)	(-10, 0)	m/s
v_{des}	(10, 0)	(10, 0)	(-10, 0)	m/s
Constraints & Solver Settings				
d_{safety}		25		m
$(v_{\text{lb}}, v_{\text{ub}})$		(5, 25)		m/s
T		40		-
Δt		0.25		s

3.1.3. Scenario S-2: Intersection

The second scenario considers an intersection encounter in which three agents cross a shared airspace. This scenario evaluates the behavior of the methods when multiple pairwise collision avoidance constraints become relevant within the same time interval. Detailed parameters are listed in Table 3 and Table 4. As in Scenario S-1, CCP-ADMM was configured with 10 CCP iterations, and the ADMM iterations were linearly reduced from 300 to 100.

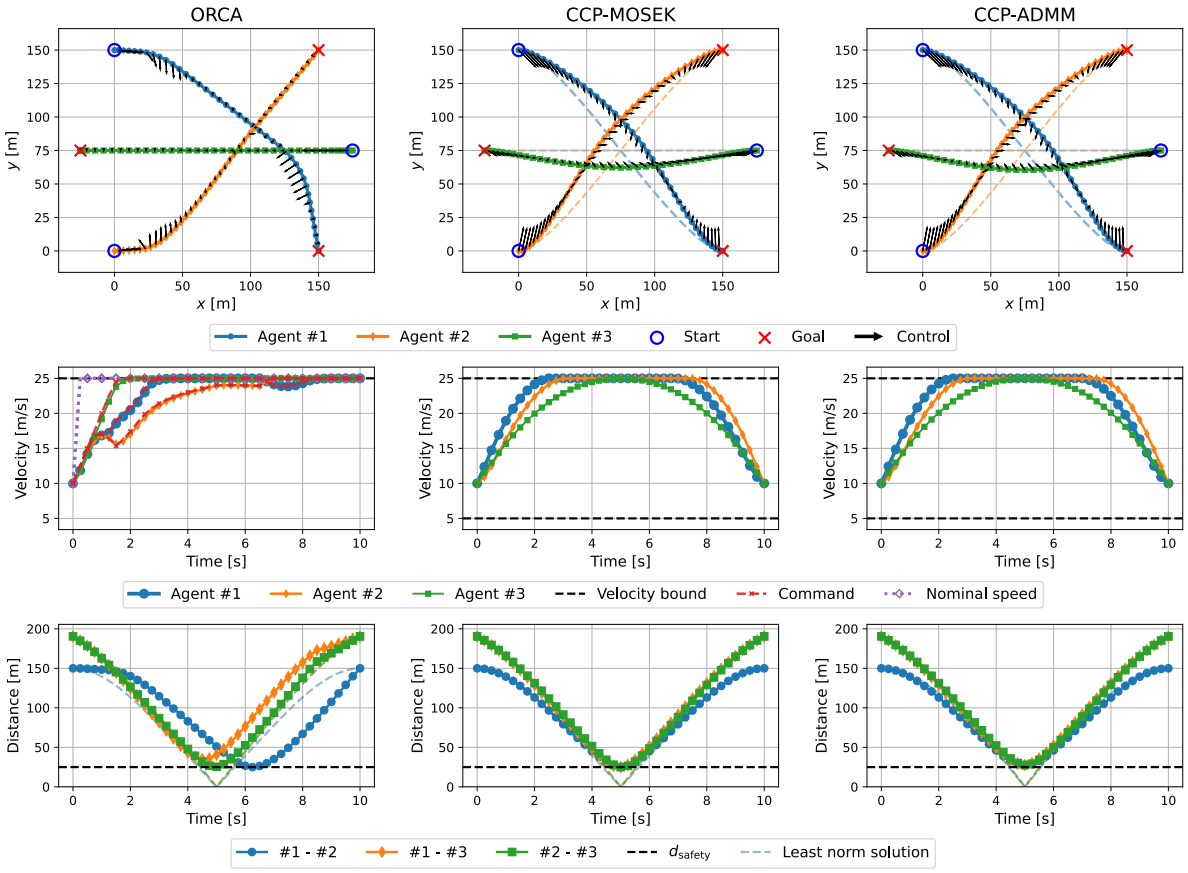


Figure 5: Trajectories, velocity profiles, and relative distances for the benchmarks and the proposed method (Scenario S-2).

Table 4
Benchmark method settings for Scenario S-2.

Method	Settings
ORCA	$v_{\text{cr}} = 25\text{m/s}$, $r_{\text{agent}} = d_{\text{safety}}/2$, $d_{\text{neighbor}} = 300\text{m}$, $N_{\text{neighbor}} = 3$, $K_v = 4$.
CCP-MOSEK	$w_s = 10$, $w_\eta = 1.0$.
CCP-ADMM	$\rho = 7$.

Figure 5 compares ORCA, CCP-MOSEK, and CCP-ADMM in terms of trajectory, speed, and pairwise relative distance. All three methods maintain the required safety distance for all agent pairs. The least-norm solution passes through the shared crossing region without considering inter-agent separation, which confirms that the nominal paths must be modified for safe intersection crossing.

In the ORCA result, the agents avoid collision through successive velocity commands, and the velocity profiles quickly approach the prescribed command range. The resulting trajectories satisfy the safety constraint, but the maneuver is shaped mainly by the velocity commands generated at each update instant. In contrast, CCP-MOSEK and CCP-ADMM optimize the state and input trajectories over the planning horizon, producing smoother and more coordinated deviations around the shared crossing region.

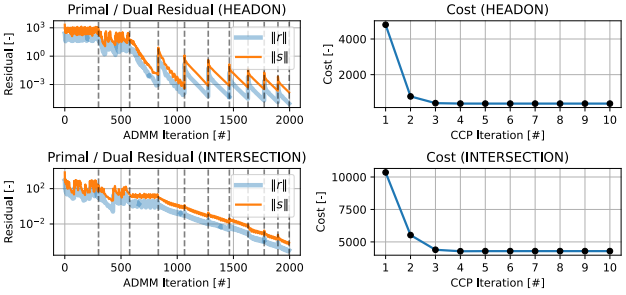


Figure 6: Residual and cost history of CCP-ADMM.

3.1.4. Convergence Behavior of CCP-ADMM

The convergence behavior of CCP-ADMM is evaluated using the primal and dual residuals and the CCP cost history in the two deterministic benchmark scenarios. Figures 6 show the residual and cost histories for Scenario S-1 and S-2. In the residual plots, the gray vertical dashed lines indicate the boundaries between consecutive outer CCP iterations.

Within each CCP iteration, the decrease of the primal residual indicates that the dynamics equality constraint and the consistency relation $S^z\tilde{x} = z$ are being better satisfied for the current CCP-linearized subproblem. The decrease of the dual residual indicates that the projected auxiliary variables change less between successive ADMM iterations,

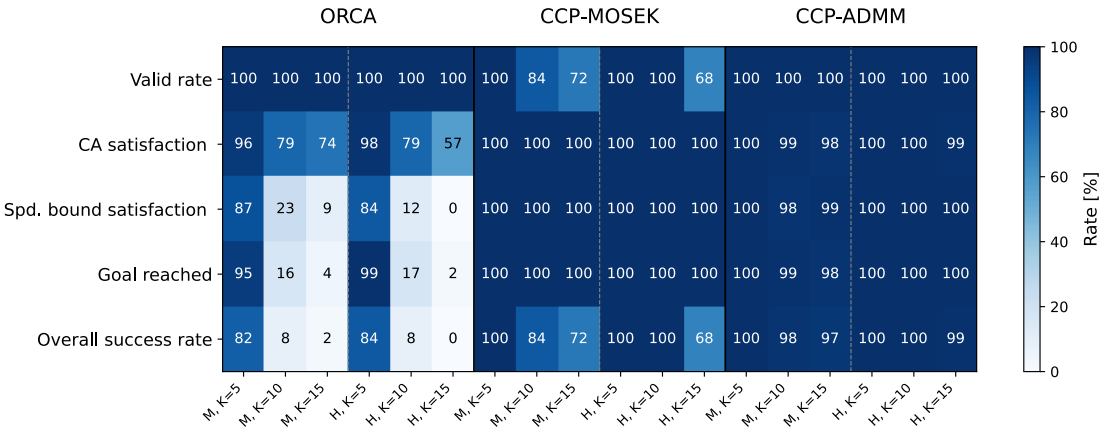


Figure 7: Tolerance-based outcome heatmap (Monte Carlo simulation).

meaning that the projection step and the multiplier update are approaching a steady state. At the beginning of a new CCP iteration, the reference trajectory and the corresponding affine collision avoidance constraints are updated, which can cause transient residual increases or oscillations. As the outer CCP iterations proceed, the residuals decrease within the prescribed ADMM iteration schedule.

The CCP cost histories show that the objective value decreases during the early outer iterations and then stabilizes. This behavior indicates that the successive CCP updates no longer produce large changes in the control effort objective, and that the trajectory has reached a consistent solution for the deterministic benchmark case.

These convergence trends support the solution profiles observed in Sections 3.1.2 and 3.1.3. In both the head-on and intersection scenarios, CCP-ADMM produces trajectory, speed, and relative distance profiles that are almost identical to those obtained by CCP-MOSEK. Here, CCP-MOSEK solves a convex trajectory-planning subproblem obtained from CCP-based convexification at each CCP iteration using MOSEK, whereas CCP-ADMM solves a projection-friendly CCP-ADMM formulation through projection-based ADMM updates. The close agreement between the two methods indicates that the proposed CCP-ADMM implementation obtains solutions of nearly the same practical quality as the MOSEK-based convexification baseline in these deterministic scenarios. The runtime and scalability advantages are evaluated separately through the K-scaling Monte Carlo analysis in Section 3.1.5.

3.1.5. K-Scaling Monte Carlo Simulation

Monte Carlo setup To evaluate scalability under randomized multi-agent configurations, a K-scaling Monte Carlo analysis was conducted with $K \in \{5, 10, 15\}$. For each trial, start nodes were sampled without duplication among agents, and goal nodes were also sampled without duplication among agents. A node used as the start node of one agent could be selected as the goal node of another agent, but each agent was assigned a goal node different from its own start node. Once the start and goal nodes were assigned, the initial velocity of each agent was set to have a speed of 10 m/s in the direction from its start node to its goal node.

Table 5
Monte Carlo parameters and benchmark settings.

Case	K	n_g	S	ρ	$t_f(=T\Delta t)$ (s)
Medium	5	4	0.245	22	10.0
	10	6	0.218	45	12.5
	15	7	0.240	50	15.0
High	5	3	0.436	22	10.0
	10	4	0.491	45	12.5
	15	5	0.471	50	15.0

Method	Parameters
ORCA	$v_{\mathrm{cr}} = 25\,\mathrm{m/s}$, $r_{\mathrm{agent}} = d_{\mathrm{safety}}/2$, $d_{\mathrm{neighbor}} = 300\,\mathrm{m}$, $N_{\mathrm{neighbor}} = K$, $K_v = 5$
CCP-MOSEK	$w_s = 10$, $w_\eta = 1.0$

The congestion level of each randomized scenario was quantified using a spatial saturation ratio. The workspace was constructed as an $n_g \times n_g$ square grid, where each grid cell has side length d_{safety}. Thus, the workspace area is $(n_g d_{\mathrm{safety}})^2$. Using an effective safety disk of radius $d_{\mathrm{safety}}/2$ for each agent, the saturation ratio is defined as $S = K\pi/4n_g^2$. Based on the number of agents and grid cells, the randomized scenarios were divided into medium- and high-saturation cases. For each case, 100 Monte Carlo trials were conducted. The grid size, saturation ratio, and scenario parameters for each case are summarized in Table 5.

Across all Monte Carlo trials, CCP-ADMM was executed with a fixed solver configuration: 6 outer CCP iterations and 300 inner ADMM iterations for each CCP subproblem. The same iteration setting was used for all values of K and all random seeds, so that the runtime statistics reflect the scaling effect of the number of agents rather than changes in the iteration policy. As described in Section 3.1.1, the ADMM penalty parameter ρ was calibrated separately for each value of K.

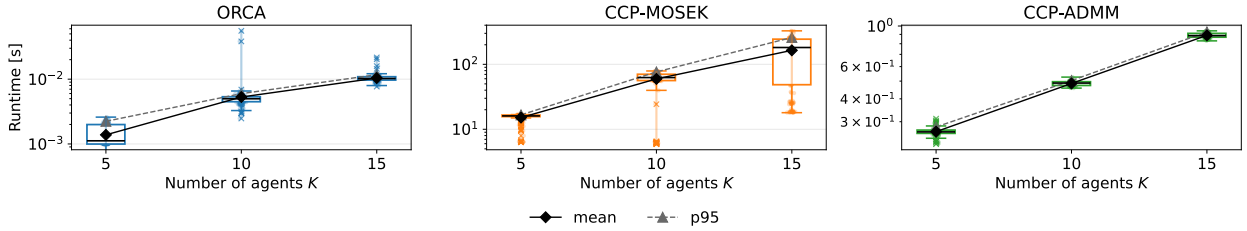


Figure 8: Runtime statistics vs. Number of agents.

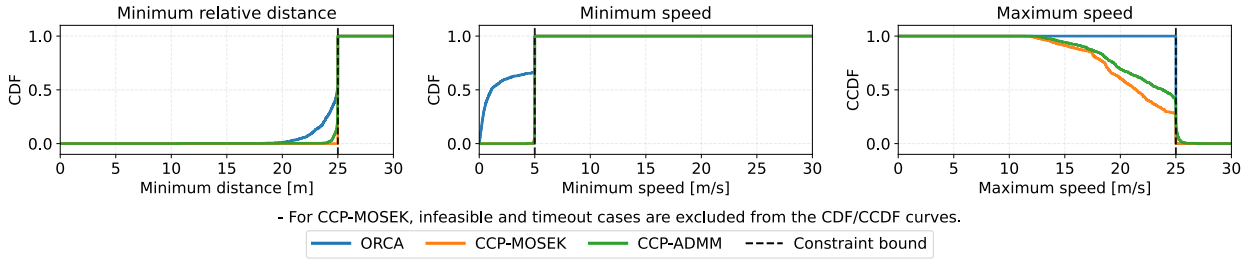


Figure 9: CDF/CCDF of constraint metrics.

Evaluation metrics The Monte Carlo results are evaluated using five tolerance-based outcome metrics: valid rate, collision avoidance satisfaction rate, speed bound satisfaction rate, goal reached rate and overall success rate. For CCP-MOSEK and CCP-ADMM, infeasible outcomes or trials exceeding the 300 s computation limit were counted as invalid, whereas ORCA was always treated as valid because it returns a velocity command at each update step.

A trial was classified as collision safe if $d_{\min} \geq d_{\text{safety}} - \epsilon_d$, speed feasible if the speed bound violation was within ϵ_v, and goal reaching if all agents reached their assigned goals within ϵ_g at the final time. In this study, $\epsilon_d = 1$ m, $\epsilon_v = 1$ m/s, and $\epsilon_g = 1$ m were used. The overall success rate was computed as the fraction of trials satisfying the rest four criteria.

Results Analysis Figure 7. summarizes the Monte Carlo outcome statistics. ORCA and CCP-ADMM return valid outputs in all tested cases, whereas the validity of CCP-MOSEK decreases as K increases in some medium and high saturation cases. This is because CCP-MOSEK can produce infeasible outcomes or exceed the 300 s timeout when the randomized scenario becomes more difficult. Although ORCA always returns a velocity command, its overall success rate decreases significantly as the number of agents and the saturation level increase. This behavior is expected because ORCA is a reactive velocity method and does not explicitly optimize a full horizon trajectory satisfying terminal, speed, and pairwise collision avoidance constraints. In contrast, CCP-ADMM maintains high overall success rates in most tested cases. The few failed cases are mainly associated with randomized configurations where reaching the assigned goals within the fixed final time while satisfying the speed and collision avoidance constraints becomes physically restrictive.

Figure 8 shows the runtime statistics with respect to the tested values of K. ORCA gives the shortest runtime because

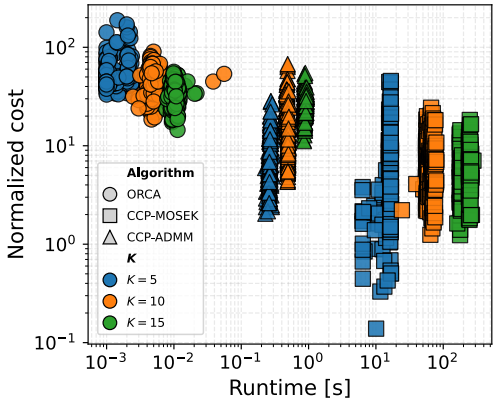


Figure 10: Runtime vs. Normalized cost.

it generates reactive velocity commands without solving a finite-horizon trajectory optimization problem. CCP-MOSEK requires the largest and most variable runtime, with both the min–max range and p95 runtime increasing as K grows. In contrast, CCP-ADMM exhibits a more regular timing trend, with the mean and p95 runtimes remaining close to each other over the tested cases. This result indicates that the proposed implementation provides a more predictable runtime profile than CCP-MOSEK in the Monte Carlo tests.

Figure 9 presents the cumulative distribution function (CDF) and complementary cumulative distribution function (CCDF) of the constraint metrics over the Monte Carlo trials. The minimum relative distance and minimum speed are shown using CDFs, whereas the maximum speed is shown using a CCDF to highlight upper speed bound violations. Samples on the admissible side of each constraint bound indicate satisfaction, while samples beyond the bound indicate violation. For valid trajectories, CCP-MOSEK shows the tightest concentration near the admissible region, followed by

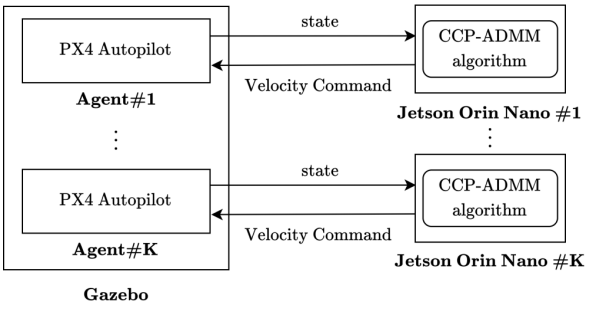


Figure 11: Schematic of the Processor-in-the-Loop Simulation (PILS) setup.

CCP-ADMM and ORCA. However, infeasible and timeout cases of CCP-MOSEK are excluded from these curves, so the CDF/CCDF results should be interpreted together with the solver outcome statistics in Figure 7. Considering validity, collision safety, speed feasibility, and goal reaching together, CCP-ADMM provides the most favorable overall success rate in the tested randomized scenarios.

Figure 10 compares runtime and normalized cost. The control-effort cost is normalized by KT to allow comparison across different numbers of agents and horizon lengths. ORCA lies in the low-runtime and high-cost region, reflecting its computational efficiency but lack of finite-horizon control-effort minimization. CCP-MOSEK achieves low normalized cost when it returns a valid solution, but its runtime is much larger and more variable. CCP-ADMM lies between these two methods, achieving substantially lower runtime than CCP-MOSEK while keeping the normalized cost closer to the solver-based optimization baseline than to ORCA. These results show that CCP-ADMM provides a practical balance among computation time, trajectory cost, and constraint satisfaction in randomized multi-agent scenarios.

3.2. Real-Time Feasibility Verification via Processor-in-the-Loop Simulation

While numerical simulations evaluate the solution behavior and constraint satisfaction of the proposed method under controlled conditions, they do not capture the execution characteristics that arise when the planner is deployed on onboard hardware and connected to a flight control stack. Therefore, we conducted Processor-in-the-Loop Simulations (PILS) to validate the *real-time feasibility* and *replanning performance* of the algorithm on an embedded platform. Here, real-time feasibility refers to the ability of the planner to operate within the embedded PILS execution loop and provide trackable reference trajectories to the flight controller.

Hardware & Software Setup: The setup, illustrated in Figure 11, employs NVIDIA Jetson Orin Nano Developer Kits as the high-level mission computers. The flight dynamics and low-level control are simulated using PX4-based firmware within the Gazebo physics engine. To mimic a distributed system, each agent is assigned a dedicated Jetson board. The CCP-ADMM algorithm runs on each Jetson board, computes

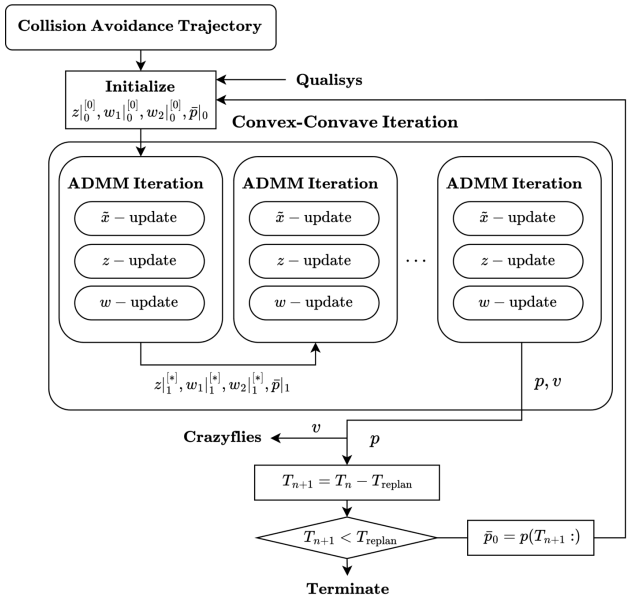


Figure 12: Flowchart of the CCP-ADMM trajectory replanning strategy.

Table 6
PILS parameters (PILS, Scenario P-1).

Parameter	Agent #1	Agent #2	Unit
Initial & Terminal Conditions			
p_0	(0, 7.5)	(0, -7.5)	m
p_{des}	(0, -7.5)	(0, 7.5)	m
v_0	(0, 0)	(0, 0)	m/s
v_{des}	(0, 0)	(0, 0)	m/s
Constraints & Solver Settings			
d_{safety}	4.5		m
$(v_{\text{lb}}, v_{\text{ub}})$	(0.3, 2.4)		m/s
T_0	32		-
Δt	0.25		s
w_{replan}	3		s
$(\rho_{\text{init}}, \rho_{\text{replan}})$	(30, 50)		-

*The lower speed bound was enforced only during the cruise portion.

the reference velocity command for its assigned agent, and transmits it to the PX4 controller via a MAVLink bridge. Inter-agent state information is exchanged through ROS 2 (DDS). At 100 Hz, each Jetson receives messages containing the agent ID, three-dimensional position, and velocity of the other agents, which are used together with its own state for collision avoidance planning.

Receding Horizon Replanning Strategy: Tracking errors, state estimation errors, and model mismatch can cause the executed trajectory to deviate from the planned trajectory. To account for these deviations, we use the trajectory replanning framework shown in Figure 12. The single-trajectory case is treated as open-loop at the trajectory-planning layer because the trajectory generated at the initial planning step is tracked throughout the maneuver without re-optimizing it using

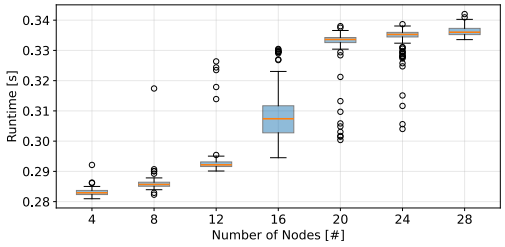


Figure 13: Computation time analysis with respect to remaining nodes on Jetson Orin Nano (PILS, Scenario P-1).

subsequent state feedback. In contrast, the replanning case is treated as closed-loop at the trajectory-planning layer because the remaining horizon is repeatedly re-optimized using state feedback during execution.

To compensate for planner latency, the replanning trigger is advanced by the mission-specific average computation time measured on the Jetson board. At each trigger time, the measured state is propagated forward over this computation time, and the predicted state at the intended update instant is used as the initial condition for the remaining-horizon optimization. This allows the updated velocity command to be applied close to the intended replanning instant.

The process is defined as follows:

1. *Initialization:* The initial full trajectory (T_0 nodes) is generated.
2. *Horizon Shift:* Replanning occurs at a fixed period w_{replan}. The corresponding number of nodes is $T_{\text{replan}} = w_{\text{replan}}/\Delta t$. At each replanning step, the elapsed T_{replan} nodes are discarded.
3. *Optimization:* The state predicted at the scheduled update time is used as the new initial condition, and the trajectory is re-optimized from that point. The remaining segment of the previous solution is shifted to the new time origin and used as a warm start.
4. *Update:* The updated velocity command is transmitted to the PX4 controller. The remaining horizon is reduced as $T_{n+1} = T_n - T_{\text{replan}}$, and the process continues until $T_{n+1} \leq T_{\text{replan}}$.

3.2.1. Scenario P-1: Head-on Encounter

The PILS parameters are listed in Table 6. For the initial trajectory generation, the numbers of CCP and ADMM iterations were set to 5 and 300, respectively. During the replanning phase, since the previous trajectory (shifted in time) provides an excellent warm-start guess, the number of iterations was reduced to 3 CCP iterations and 100 ADMM iterations.

To verify the computational timing, we analyzed the execution time relative to the receding horizon length. Figure 13 presents the computation time measured over 100 trials on the Jetson Orin Nano. To ensure safety and continuity, the replanning trigger was scheduled by subtracting the average computation time from the target replanning timestamp.

Figures 14 and 15 compare the single-trajectory open-loop planning mode and the closed-loop replanning mode.

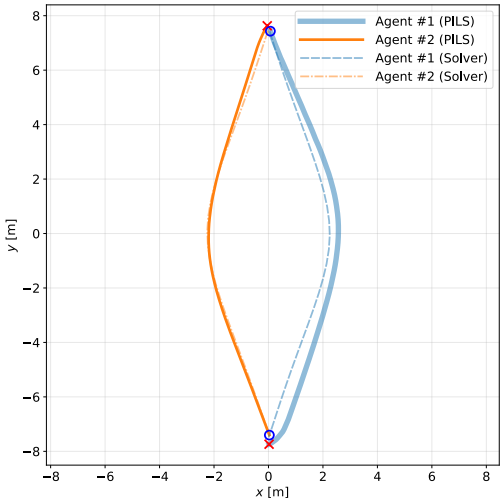


Figure 14: Trajectory (PILS, Scenario P-1).

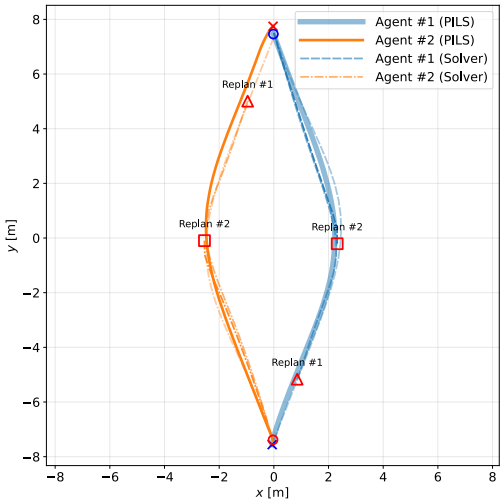


Figure 15: Replanned trajectory (PILS, Scenario P-1).

The dashed curves represent the optimal reference trajectories generated by the solver, while the solid curves represent the flight trajectories executed in the PILS environment.

Figure 16, which shows the relative distances between vehicles over time, confirms that the safety distance constraints are satisfied in both approaches. Furthermore, the velocity profiles presented in Figure 17 remain within the predefined limits. These results demonstrate that both the single-trajectory-based approach and the replanning-based approach yield valid solutions that satisfy the collision-avoidance constraints. These results show that both trajectory planning modes operate properly in the PILS environment while maintaining the prescribed safety distance and speed bounds.

3.2.2. Scenario P-2 : Multi-Agent Intersection

The parameters used in the intersection scenario are listed in Table 7. Similar to Scenario P-1, the numbers of CCP and ADMM iterations were: 5 CCP / 300 ADMM iterations for

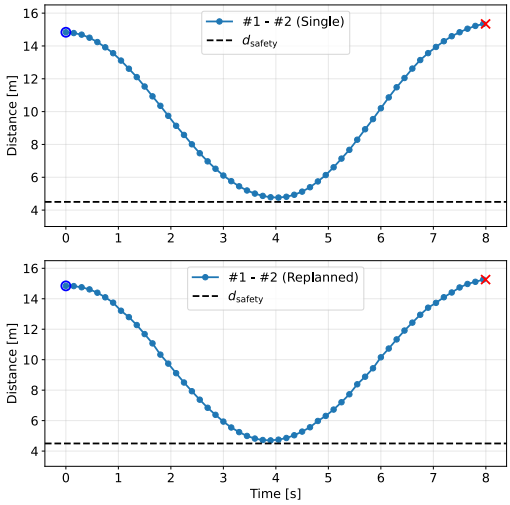


Figure 16: Relative distance (PILS, Scenario P-1).

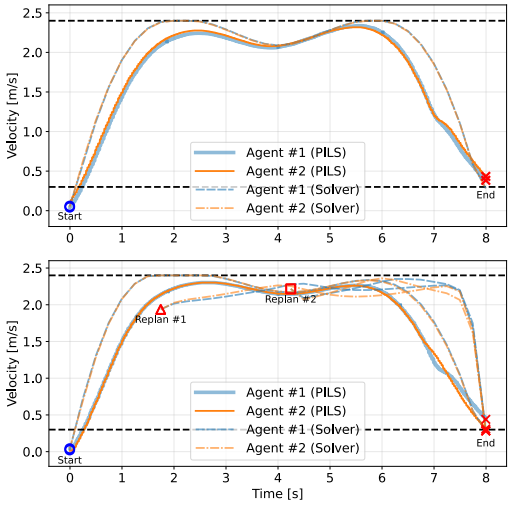


Figure 17: Velocity profile (PILS, Scenario P-1).

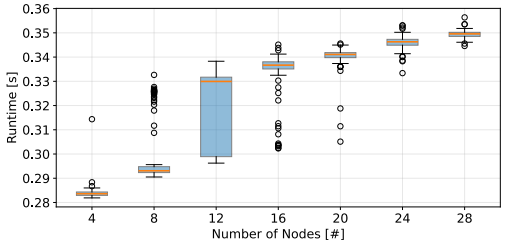


Figure 18: Computation time analysis with respect to remaining nodes on Jetson Orin Nano (PILS, Scenario P-2).

initial generation, and 3 CCP / 100 ADMM iterations for replanning.

Similar to head-on scenario, the computation time relative to the number of remaining nodes was measured over 100 iterations, as illustrated in Figure 18. The replanning process was configured to execute at a point adjusted by subtracting the average computation time from the replanning step.

Table 7
PILS parameters (PILS, Scenario P-2).

Parameter	Agent #1	Agent #2	Agent #3	Unit
Initial & Terminal Conditions				
p_0	(0, -7.5)	(-6, 6)	(6, 6)	m
p_{des}	(0, 7.5)	(6, -6)	(-6, -6)	m
v_0	(0, 0)	(0, 0)	(0, 0)	m/s
v_{des}	(0, 0)	(0, 0)	(0, 0)	m/s
Constraints & Solver Settings				
d_{safety}	4.5			m
$(v_{\text{lb}}, v_{\text{ub}})$	(0.3, 2.4)			m/s
T_0	32			-
Δt	0.25			s
w_{replan}	3			s
$(\rho_{\text{init}}, \rho_{\text{replan}})$	(100, 50)			-

*The lower speed bound was enforced only during the cruise portion.

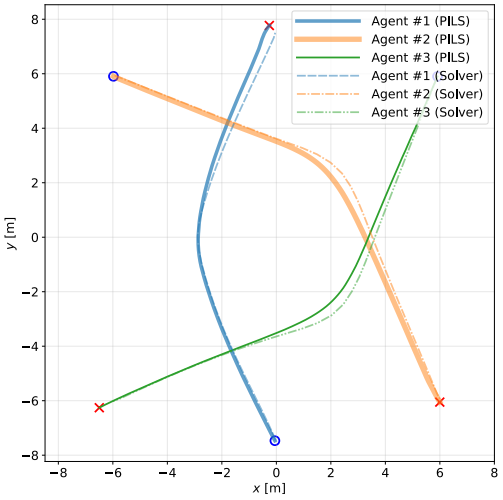


Figure 19: Trajectory (PILS, Scenario P-2).

Figures 19 and 20 compare the single-trajectory open-loop planning mode and the closed-loop replanning mode in the three-agent intersection scenario. Compared with Scenario P-1, this case requires simultaneous satisfaction of multiple pairwise collision avoidance constraints near the shared crossing region. Figure 21 shows that all pairwise relative distances remain above the prescribed safety distance in both planning modes. Figure 22 shows that the velocity profiles follow the planned references and remain within the prescribed operating range during the main maneuver. These results show that both the single-trajectory-based approach and the replanning-based approach yield valid solutions that satisfy the collision-avoidance constraints in the PILS environment.

3.3. Indoor Flight Experiments

To validate the proposed algorithm in a physical environment, flight experiments were conducted using multiple Crazyflie 2.1 quadrotors (Giernacki et al., 2017). The purpose of these experiments is to evaluate the proposed

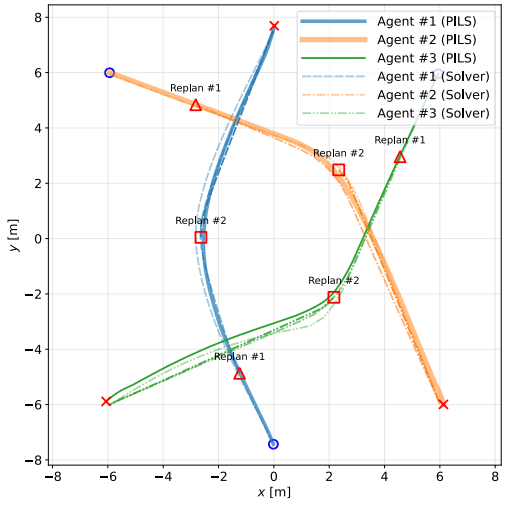


Figure 20: Replanned trajectory (PILS, Scenario P-2).

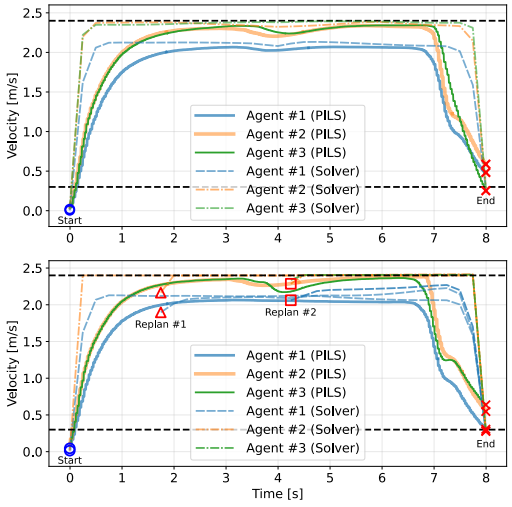


Figure 22: Velocity profile (PILS, Scenario P-2).

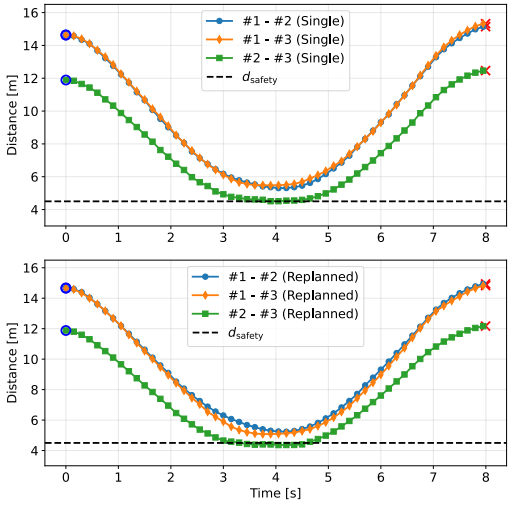


Figure 21: Relative distance (PILS, Scenario P-2).

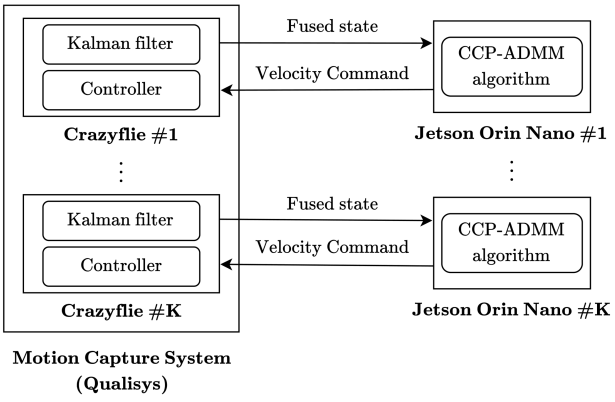


Figure 23: Schematic of the indoor flight test environment integrating motion capture and distributed computing.

framework with trajectory replanning under practical indoor flight conditions, including motion-capture feedback, RF communication, and tracking errors associated with small aerial platforms. The experimental setup uses a Qualisys motion capture system for external state feedback. During flight, the measured state feedback is used to update the remaining trajectory, allowing the CCP-ADMM algorithm to account for deviations between the planned and executed trajectories. The flight altitude was fixed at approximately 0.5 m, and the validation focuses on horizontal collision avoidance, safety-distance maintenance, and planned velocity-command bound satisfaction on the physical testbed.

Hardware Architecture: Since the Crazyflie's onboard STM32 microcontroller is not suitable for solving the CCP-ADMM optimization at the required update rate, we adopted an agent-wise offboard computing architecture. As shown in Figure 24, each quadrotor is paired with a dedicated NVIDIA Jetson Orin Nano board. The Jetson board receives real-time position and velocity data from the Qualisys system via a ROS 2 network. At each replanning step, the Jetson board executes the proposed CCP-ADMM algorithm and updates the remaining reference trajectory. The resulting velocity references are transmitted to the Crazyflie at 100 Hz via RF communication.

Control Strategy: Compared with the PX4-based PILS environment in Section 3.2, the physical Crazyflie platform exhibits larger tracking errors due to its small size, limited low-level control authority, battery-dependent thrust response, and RF communication interface. To compensate for this hardware limitation and improve trajectory tracking, an external position control loop was implemented on the Jetson side using the Qualisys motion capture feedback. In this control scheme, the position profile generated by the CCP-ADMM algorithm serves as the reference setpoint, while the velocity profile acts as a feedforward term to improve dynamic response. Additionally, to ensure operational safety, a fail-safe logic was implemented to trigger an automatic landing once the vehicle enters a 30 cm radius of its destination. The

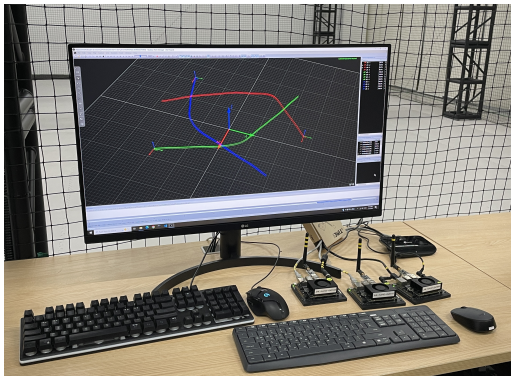


Figure 24: Hardware configuration: Distributed architecture with Jetson Orin Nano boards and Qualisys motion capture interface.

Table 8
Flight test parameters (Flight test, Scenario F-1).

Parameter	Agent #1	Agent #2	Unit
Initial & Terminal Conditions			
p_0	(0, 2.5)	(0, -2.5)	m
p_{des}	(0, -2.5)	(0, 2.5)	m
v_0	(0, 0)	(0, 0)	m/s
v_{des}	(0, 0)	(0, 0)	m/s
Constraints & Solver Settings			
d_{safety}	1.5		m
$(v_{\text{lb}}, v_{\text{ub}})$	(0.1, 0.8)		m/s
T_0	40		-
Δt	0.25		s
w_{replan}	3		s
$(\rho_{\text{init}}, \rho_{\text{replan}})$	(30, 50)		-

*The lower speed bound was enforced only during the cruise portion.

overall experimental configuration is depicted in Figures 23–24. Flight tests were conducted for head-on and intersection scenarios using the same encounter topologies as in the numerical and PILS validations, scaled to the indoor flight-test environment.

3.3.1. Scenario F-1: Head-on Encounter

The flight test parameters for the head-on scenario are listed in Table 8. Consistent with previous PILS tests, the solver settings were: 5 CCP / 300 ADMM iterations for initial generation, and 3 CCP / 100 ADMM iterations for replanning.

The replanning flight trajectory for the head-on encounter is shown in Figure 25. The dashed lines represent the reference trajectories generated by the CCP-ADMM algorithm, while the solid lines indicate the actual flight paths.

Figure 26 shows the corresponding state profiles. The upper panel presents the inter-agent relative distance, confirming that the prescribed safety threshold was maintained throughout the flight. The lower panel compares the planned velocity commands with the measured velocity responses. Compared to the PILS results, the measured velocity response in the flight test exhibits more noticeable overshoot relative to

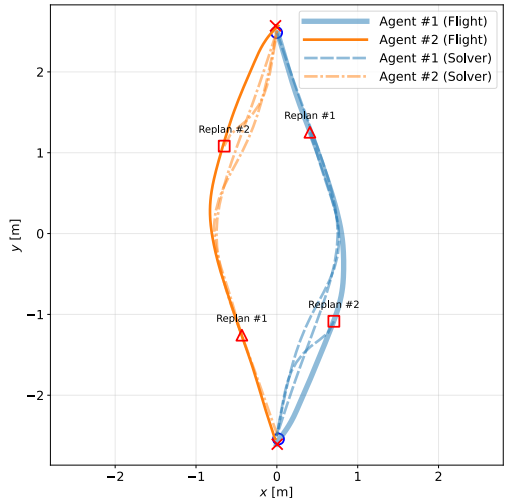


Figure 25: Replanned trajectory (Flight test, Scenario F-1).

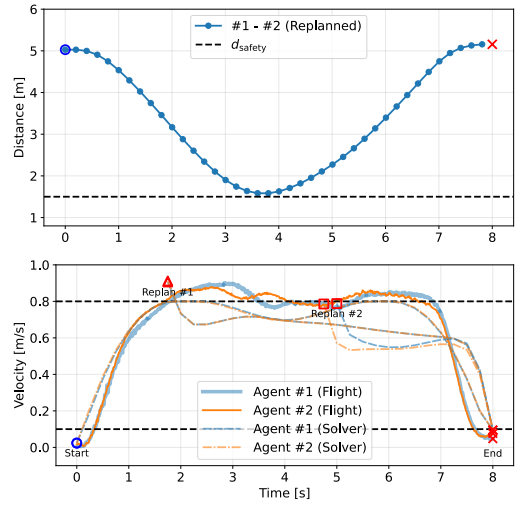


Figure 26: Relative distance and velocity profile (Flight test, Scenario F-1).

the planned command. This discrepancy is mainly attributed to the tracking limitations of the Crazyflie platform and the offboard command interface, including battery-dependent thrust response and simplified thrust mapping in the low-level controller. Nevertheless, the velocity commands generated by CCP-ADMM remained within the prescribed bounds $(v_{\text{lb}}, v_{\text{ub}})$. The changes in the command profile at the replanning instants reflect the updated remaining-horizon solutions.

Overall, the replanning-based flight experiment was successfully executed while maintaining the prescribed safety distance. The result shows that the proposed framework can update the remaining reference trajectory during flight in response to deviations from the previously planned trajectory.

Table 9
Flight test parameters (Flight test, Scenario F-2).

Parameter	Agent #1	Agent #2	Agent #3	Unit
Initial & Terminal Conditions				
p_0	(0, -2.5)	(-2, 2)	(2, 2)	m
p_{des}	(0, 2.5)	(2, -2)	(-2, -2)	m
v_0	(0, 0)	(0, 0)	(0, 0)	m/s
v_{des}	(0, 0)	(0, 0)	(0, 0)	m/s
Constraints & Solver Settings				
d_{safety}		1.5		m
$(v_{\text{lb}}, v_{\text{ub}})$		(0.1, 0.8)		m/s
T_0		40		-
w_{replan}		3		s
Δt		0.25		s
$(\rho_{\text{init}}, \rho_{\text{replan}})$		(100, 50)		-

*The lower speed bound was enforced only during the cruise portion.

3.3.2. *Scenario F-2: Multi-Agent Intersection*

The intersection scenario parameters are detailed in Table 9. Solver settings remained consistent with the previous flight test.

The replanning flight trajectory for the three-quadrotor intersection scenario is shown in Figure 27. The dashed lines represent the reference trajectories generated by the CCP-ADMM algorithm at the initial planning and replanning instants, while the solid lines indicate the measured flight paths. The changes in the reference segments at the replanning instants show that the remaining trajectories are updated during flight execution.

Figure 28 shows the corresponding state profiles. The upper panel presents the measured pairwise relative distances, confirming that all pairwise distances remained above the prescribed safety threshold throughout the flight. The lower panel compares the planned velocity commands with the measured velocity responses. As in Scenario F-1, the measured velocity responses exhibit overshoot due to the tracking limitations of the Crazyflie platform and the offboard command interface. Nevertheless, the velocity commands generated by CCP-ADMM remained within the prescribed bounds $(v_{\text{lb}}, v_{\text{ub}})$. The command profile changes at the replanning instants reflect the newly optimized remaining-horizon solutions.

In conclusion, the flight experiments demonstrate that the replanning-based CCP-ADMM framework can be executed on the Crazyflie–Jetson–Qualisys testbed while maintaining the prescribed safety distance in the tested head-on and intersection scenarios. Specifically, the results show that the algorithm can generate trackable collision-avoidance reference trajectories for small quadrotors under practical tracking errors and communication effects. Furthermore, the replanning results indicate that the proposed framework can update the remaining trajectory during flight execution using measured state feedback, supporting practical implementation in multi-agent indoor flight tests.

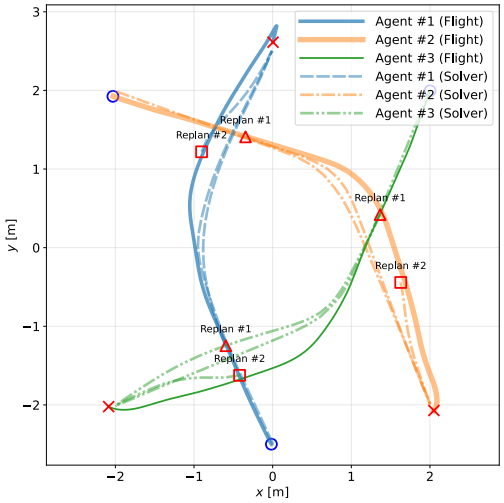


Figure 27: Replanned trajectory (Flight test, Scenario F-2).

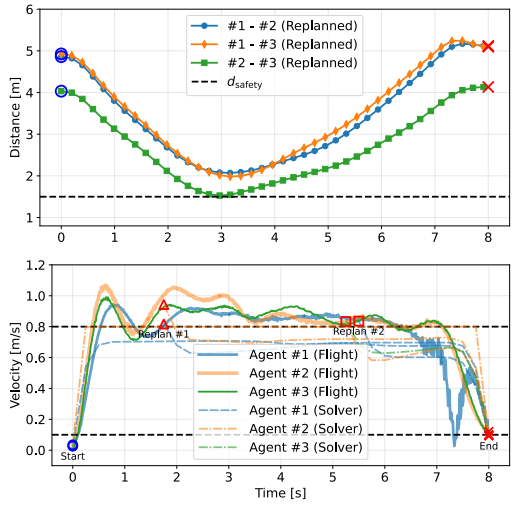


Figure 28: Relative distance and velocity profile (Flight test, Scenario F-2).

4. CONCLUSION

This paper presented a real-time finite-horizon trajectory optimization framework for multi-UAV collision avoidance, designed for embedded implementation. The proposed CCP-ADMM method combines CCP-linearized pairwise separation constraints with reusable linear-system solves, radial speed projections, and pairwise half-space corrections. This structured update scheme reduces online computational overhead while explicitly handling dynamics, terminal conditions, speed bounds, and pairwise separation constraints.

The proposed CCP-ADMM method was evaluated through numerical benchmarks, K-scaling Monte Carlo simulations, PILS, and indoor flight experiments. In the head-on and intersection benchmarks, CCP-ADMM produced trajectory, velocity, and relative distance profiles close to those of the CCP-MOSEK while satisfying the prescribed safety and velocity constraints. The Monte Carlo results

showed lower and more predictable runtime than CCP-MOSEK and higher overall success rates than the compared baselines in the tested randomized multi-agent cases.

PILS simulation confirmed that the planner can run within the embedded software stack and demonstrated both single-trajectory execution and receding-horizon replanning. Indoor flight experiments further demonstrated replanning on a physical testbed while maintaining the prescribed safety distance in the tested scenarios. Although the measured velocity responses showed tracking overshoot, the planned commands remained within the imposed bounds. These results demonstrate the practical applicability of the proposed CCP-ADMM framework as a real-time embedded planner for multi-UAV collision avoidance.

Acknowledgements

This work was supported in part by Korea Research Institute for defense Technology planning and advancement (KRIT) grant funded by the Korea government (Defense Acquisition Program Administration) (No. 20-105-E00-005(KRIT-CT-23-010), VTOL Technology Research Center for Defense Applications, 2025), and in part by Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, Republic of Korea (No. NRF-2021M3C1C1A02099428).

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Arul, S.H., Manocha, D., 2020. DCAD: Decentralized collision avoidance with dynamics constraints for agile quadrotor swarms. IEEE Robotics and Automation Letters 5, 1191–1198.

Arul, S.H., Manocha, D., 2021. SwarmCCO: Probabilistic reactive collision avoidance for quadrotor swarms under uncertainty. IEEE Robotics and Automation Letters 6, 2437–2444.

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

Baca, T., Hert, D., Loianno, G., Saska, M., Kumar, V., 2018. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of Unmanned Aerial Vehicles, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE. pp. 6753–6760.

Bareiss, D., Van den Berg, J., 2013. Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles, in: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE. pp. 3847–3853.

Van den Berg, J., Lin, M., Manocha, D., 2008. Reciprocal velocity obstacles for real-time multi-agent navigation, in: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE. pp. 1928–1935.

Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., 2011. Distributed optimization and statistical learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine Learning 3, 1–122.

Boyd, S., Vandenberghe, L., 2004. Convex optimization. Cambridge University Press.

Castillo-Lopez, M., Sajadi-Alamdari, S.A., Sanchez-Lopez, J.L., Olivares-Mendez, M.A., Voos, H., 2018. Model predictive control for aerial collision avoidance in dynamic environments, in: 2018 26th Mediterranean Conference on Control and Automation, IEEE. pp. 1–6.

Choi, J., Kim, J.H., 2024. Powered descent guidance via first-order optimization with expansive projection. IEEE Access 12, 46232–46240.

Choi, J., Kim, J.H., 2025a. A distributed method for first-order optimization with expansive projection for powered descent guidance, in: AIAA SCITECH 2025 Forum, p. 1507.

Choi, J., Kim, J.H., 2025b. Liftproj: Physics-informed Koopman lifting and projection for nonlinear optimal control via first-order optimization. IEEE Control Systems Letters .

Choi, J., Lee, D., Kim, J.H., 2026. Neural projection operators for real-time 6-DoF powered descent guidance, in: AIAA SCITECH 2026 Forum, p. 1169.

Fiorini, P., Shiller, Z., 1998. Motion planning in dynamic environments using velocity obstacles. The International Journal of Robotics Research 17, 760–772.

Fox, D., Burgard, W., Thrun, S., 2002. The Dynamic Window Approach to collision avoidance. IEEE Robotics & Automation Magazine 4, 23–33.

Giernacki, W., Skwierczyński, M., Witwicki, W., Wroński, P., Kozierski, P., 2017. Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering, in: Proceedings of the International Conference on Methods and Models in Automation and Robotics, IEEE. pp. 37–42.

Huang, Z., Shen, S., Ma, J., 2023. Decentralized iLQR for cooperative trajectory planning of connected autonomous vehicles via dual consensus ADMM. IEEE Transactions on Intelligent Transportation Systems 24, 12754–12766.

Ji, J., Khajepour, A., Melek, W.W., Huang, Y., 2016. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Transactions on Vehicular Technology 66, 952–964.

Kim, Y.J., Choi, J., Choi, J., Kim, J.H., 2026. A first-order approach for nonlinear optimal control under nonconvex constraints. Optimization and Engineering 27.

Lanckriet, G., Sriperumbudur, B.K., 2009. On the convergence of the concave-convex procedure. Advances in Neural Information Processing Systems 22.

Lipp, T., Boyd, S., 2016. Variations and extension of the convex–concave procedure. Optimization and Engineering 17, 263–287.

Matiussi Ramalho, G., Carvalho, S.R., Finardi, E.C., Moreno, U.F., 2018. Trajectory optimization using sequential convex programming with collision avoidance. Journal of Control, Automation and Electrical Systems 29, 318–327.

Missura, M., Bennewitz, M., 2019. Predictive collision avoidance for the Dynamic Window Approach, in: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE. pp. 8620–8626.

Niu, G., Wu, L., Gao, Y., Pun, M.O., 2022. Unmanned aerial vehicle (UAV)-assisted path planning for unmanned ground vehicles (UGVs) via disciplined convex-concave programming. IEEE Transactions on Vehicular Technology 71, 6996–7007.

Parikh, N., Boyd, S., et al., 2014. Proximal algorithms. Foundations and Trends in Optimization 1, 127–239.

Park, G., Choi, J., Jeong, D.H., Kim, J.H., 2024. Optimal impact angle guidance via first-order optimization under nonconvex constraints, in: Proceedings of the American Control Conference, IEEE. pp. 778–784.

Van Den Berg, J., Snape, J., Guy, S.J., Manocha, D., 2011. Reciprocal collision avoidance with acceleration-velocity obstacles, in: Proceedings of the IEEE International Conference on Robotics and Automation, IEEE. pp. 3475–3482.

Yuille, A.L., Rangarajan, A., 2003. The Concave-Convex procedure. Neural Computation 15, 915–936.

Zheng, L., Yang, R., Wang, M.Y., Ma, J., 2024. Barrier-enhanced parallel homotopic trajectory optimization for safety-critical autonomous driving. IEEE Transactions on Intelligent Transportation Systems 26, 2169–2186.