DMPC-Swarm: Distributed Model Predictive Control on Nano UAV Swarms

Alexander Gräfe[1*], Joram Eickhoff[1], Marco Zimmerling[2], Sebastian Trimpe[1]

[1]Institute for Data Science in Mechanical Engineering, RWTH Aachen University, Theaterstraße 35-39, Aachen, 52062, Germany.
[2]Networked Embedded Systems Lab, TU Darmstadt, Mornewegstraße 30, Darmstadt, 64293, Germany.

*Corresponding author(s). E-mail(s): alexander.graefe@dsme.rwth-aachen.de;
Contributing authors: joram.eickhoff@dsme.rwth-aachen.de;
marco.zimmerling@tu-darmstadt.de; trimpe@dsme.rwth-aachen.de;

Abstract

Swarms of unmanned aerial vehicles (UAVs) are increasingly becoming vital to our society, undertaking tasks such as search and rescue, surveillance and delivery. A special variant of Distributed Model Predictive Control (DMPC) has emerged as a promising approach for the safe management of these swarms by combining the scalability of distributed computation with dynamic swarm motion control. In this DMPC method, multiple agents solve local optimization problems with coupled anti-collision constraints, periodically exchanging their solutions. Despite its potential, existing methodologies using this DMPC variant have yet to be deployed on distributed hardware that fully utilize true distributed computation and wireless communication. This is primarily due to the lack of a communication system tailored to meet the unique requirements of mobile swarms and an architecture that supports distributed computation while adhering to the payload constraints of UAVs. We present DMPC-SWARM, a new swarm control methodology that integrates an efficient, stateless low-power wireless communication protocol with a novel DMPC algorithm that provably avoids UAV collisions even under message loss. By utilizing event-triggered and distributed off-board computing, DMPC-SWARM supports nano UAVs, allowing them to benefit from additional computational resources while retaining scalability and fault tolerance. In a detailed theoretical analysis, we prove that DMPC-SWARM guarantees collision avoidance under realistic conditions, including communication delays and message loss. Finally, we present DMPC-SWARM's implementation on a swarm of up to 16 nano-quadcopters, demonstrating the first realization of these DMPC variants with computation distributed on multiple physical devices interconnected by a real wireless mesh networks. A video showcasing DMPC-SWARM is available at http://tiny.cc/DMPCSwarm.

Keywords: Robot Swarms, Safety, Collision Avoidance, Distributed Control

1 Introduction

Robot swarms have the potential to transform fields like smart farming, civil protection, and planetary exploration (Trianni et al., 2016; Blender et al., 2016; Cantizani-Estepa et al., 2022; Couceiro et al., 2013; Staudinger et al., 2021; Zhang et al., 2020; Kang et al., 2019). Although currently confined to laboratories, they are expected to become commonplace by 2030 (Dorigo et al., 2020). This holds especially for swarms of unmanned aerial vehicles (UAVs) relevant for tasks such as smart farming (Walter et al., 2017; Trianni et al., 2016; R Shamshiri et al., 2018), aerial surveillance (Saska et al., 2016; Chung et al., 2018), and wildlife observation (Shah et al., 2020).

A fundamental principle in robot swarms is *distributed control*, where agents collaboratively make decisions through distributed computation and communication, unlike *centralized control* by a single agent. Distributed control leverages the collective's computational power, enhances scalability, and improves resilience by avoiding single points of failure (Mondada et al., 2004; Luis and Schoellig, 2019; Ge et al., 2017).

Building upon these advantages, recent work proposed a distributed optimization approach based on *Distributed Model Predictive Control* (DMPC), which is one of the most promising distributed swarm control methods in recent years (Luis and Schoellig, 2019; Luis et al., 2020; Park et al., 2022, 2023; Chen et al., 2023, 2024; Gräfe et al., 2022). These approaches enable dynamic swarm maneuvers by continuously solving distributed optimization problems across the agents. Moreover, by incorporating specific constraints into its optimization problem, they can formally guarantee collision avoidance (Park et al., 2022, 2023; Chen et al., 2023, 2024; Gräfe et al., 2022), which is crucial for ensuring safety in swarm operations.

However, despite promising theoretical analyses and simulations demonstrating DMPC's potential, existing hardware implementations execute DMPC on a central computer, while simulating selected effects of distributed computation and communication (Luis et al., 2020; Park et al., 2022, 2023; Chen et al., 2023, 2024; Gräfe et al., 2022). Although these implementations confirm DMPC's conceptual suitability for swarm control, they do not realize the practical benefits of distributed architectures. Moreover, centralized implementations oversimplify real-world conditions, particularly communication delays and message loss,

rendering them inadequate for evaluating DMPC-controlled swarms in scenarios where the advantages of distributed systems are essential.

To bridge this gap, we develop and implement DMPC-SWARM, the first *distributed realization* of DMPC for swarm control over wireless networks. We identify three key challenges to a real-world realization of DMPC, which we address in this work:

C1 Communication: DMPC requires communication between agents over a wireless network. In DMPC-SWARM, we aim to achieve this communication over self-organizing wireless mesh networks, where agents transmit data for one another based on device-to-device communication. This preserves the distributed nature of the swarm and enhances scalability, reliability, and efficiency compared to a network in which agents are limited to direct communicate with a dedicated base station (Laneman et al., 2004). However, wireless mesh networks introduce complexities, especially as robots move and the network topology continuously evolves (Khan et al., 2019; Oubbati et al., 2019; Namdev et al., 2021). Consequently, achieving the communication required by DMPC approaches becomes even more challenging.

C2 Computation: DMPC necessitates significant computational power because it involves repeatedly solving optimization problems. However, UAVs, especially nano-quadcopters used in applications such as agriculture (Gago et al., 2020), indoor source seeking (Duisterhof et al., 2021; Karagüzel et al., 2023) and indoor visual inspection (Tavasoli et al., 2023), have limited onboard computing capabilities due to constraints in size, weight, and cost. Despite these limitations, most existing DMPC implementations fail to address the computational restrictions inherent in such UAVs. Furthermore, swarm systems often comprise heterogeneous computing architectures with varying computational capabilities, for instance, quadcopters of different sizes, mobile robots, and operator devices. Current DMPC approaches generally do not exploit this diversity in computational resources.

C3 Unrealistic assumptions of DMPC: Moreover, existing DMPC methods often overlook practical challenges such as computational delays due to limited processing power, communication delays

and message loss resulting from unreliable wireless communication. These issues pose significant hurdles, as they may cause the collision avoidance guarantees provided by DMPC methods to fail in practice, thereby limiting their real-world applicability. Solving these challenges on the algorithmic side is crucial for the successful deployment of DMPC in practical swarm applications.

We address these challenges through a novel combination of a stateless communication protocol, a distributed and event-triggered compute architecture leveraging multiple heterogeneous compute nodes, as well as algorithmic extensions to DMPC, achieving the first distributed realization of DMPC.

We note that different concepts and notions of distributed systems exist. In our work, we define "distributed" as collaborative problem-solving among multiple agents, sharing workloads and operating without a single coordinator (Hu et al., 2018; Cheraghi et al., 2022; Lupashin et al., 2014).

One prevalent alternative view on distributed systems emphasizes interaction and information exchange with agents in a neighborhood and peer-to-peer fashion (Antonelli, 2013; Ge et al., 2017). Although such approaches offer scalability benefits, they can also involve downsides in other settings. For example, defining and tracking which agents qualify as neighbors is often problematic, particularly in scenarios involving rapidly moving swarms, which complicates the theoretical analysis of algorithms. This is especially relevant when the interaction involves wireless communication, where the behavior and existence of communication links to neighbors can be difficult to predict due to the significant influence of environmental factors.

Following our view on "distributed" as collaborative problem-solving and distribution of workloads, we consider any system that facilitates these functions as a potential solution, rather than restricting interaction to a peer-to-peer fashion. Accordingly, we design and integrate all three system components, algorithms, computing infrastructure, and communication infrastructure, into an architecture featuring distributed entities at each component. For example the distributed communication solution that we propose is based on physical neighbor-to-neighbor communication, while enabling a many-to-all information exchange and global interaction as part of the solution.

Before detailing our contributions, we make the swarm control problem precise.

1.1 Problem Setting

We consider a swarm $\mathcal{A} = \{1, \cdots, N\}$ of N UAVs. At any time t, each UAV $i \in \mathcal{A}$ has a position $p_i(t) \in \mathbb{R}^3$. A UAV can measure its own position but not the positions of the other UAVs. Each must navigate from its initial position $p_{i,\text{init}} = p_i(0)$ to a target position $p_{i,\text{target}}$ without colliding with other UAVs.

The target positions $p_{i,\text{target}}$ may change over time, e.g., when a UAV receives a new task at a different location. This dynamic environment necessitates frequent real-time trajectory replanning. Such distributed position-following problem is the basis of many swarming tasks and is the standard problem setting of DMPC (Luis and Schoellig, 2019; Luis et al., 2020; Park et al., 2023, 2022; Chen et al., 2023, 2024).

Each UAV can run a low-level trajectory-following controller but lacks the resources to solve the optimization problems in DMPC. However, the network—in addition to the N UAVs—also includes M agents ($M > 1$) with higher computational power, called *compute units* (CUs). We assume $M < N$ to account for limited capacities. Such CUs are common in practice, e.g., in smart farming or manufacturing, UAVs may connect to digital devices (e.g., smartphones or laptops) or rely on edge clouds and computational resources from other agents like ground robots or larger UAVs (Baumann et al., 2020; Sankaranarayanan et al., 2023; Liu et al., 2021).

All UAVs and CUs communicate via a wireless mesh network, each device equipped with a wireless transceiver. Given the limited range of these transceivers, devices can directly communicate only with their immediate neighbors. As the agents move, the network topology changes dynamically, continuously altering the quality and availability of direct communication links, which is known to be challenging for efficient and reliable communication (Khan et al., 2019; Oubbati et al., 2019; Namdev et al., 2021; Chriki et al., 2019; Lv et al., 2023).

Problem Statement. Our goal is to develop a swarm architecture for the safe control of real UAV swarms using DMPC, realized on distributed physical hardware and a real wireless mesh network. The DMPC computations must be distributed across multiple CUs, considering their limited number, and should ensure that the UAVs' positions converge to the target positions over time

$$\forall i \in \mathcal{A} : p_i(t) \to p_{i,\text{target}}, \tag{1}$$

while provably avoiding collisions between UAVs

$$\forall t \in \mathcal{R}_0^+, \, \forall i,j \in \mathcal{A}, \, i \neq j : \qquad (2)$$
$$\left\| \Theta^{-1}[p_j(t) - p_i(t)] \right\|_2 \geq d_{\min},$$

where Θ is a scaling matrix for downwash effects, and $d_{\min}$ is the minimum allowable distance (Luis and Schoellig, 2019; Luis et al., 2020; Chen et al., 2024, 2023; Park et al., 2022, 2023; Gräfe et al., 2022). The DMPC framework should function under real-world conditions, such as communication message loss, delays and limited computational resources.

1.2 Contributions

Our solution, DMPC-SWARM, effectively solves this problem by addressing challenges **C1**–**C3**. Overall, we make the following contributions:

1. We present DMPC-SWARM, a novel swarm architecture combining low-power wireless communication, distributed computing and a new DMPC algorithm. The communication protocol is resilient to rapid device movement and provides the communication required by DMPC. The architecture building on top of it efficiently balances the DMPC computational load across UAVs and CUs.

2. We introduce message-loss-recovery DMPC (MLR-DMPC) as the algorithmic part of DMPC-SWARM.. This DMPC method handles message loss and communication delays, providing collision avoidance guarantees under realistic conditions.

3. We implement DMPC-SWARM, achieving the first distributed hardware realization of DMPC-based swarm control on nano-quadcopters. Our experimental setup features up to 16 nano-quadcopters communicating over a wireless mesh network using Bluetooth Low Energy (BLE). The entire system is based on the popular Crazyflie platform, with its software and hardware files accessible at https://github.com/Data-Science-in-Mechanical-Engineering/DMPC-Swarm. A video demonstrating our results is available at http://tiny.cc/DMPCSwarm.

2 Related Work

Before presenting our solution, we analyze related DMPC methods. For comprehensive reviews of general UAV swarm control methods, we refer to Gugan and Haque (2023); Wang et al. (2024); Yu et al. (2023).

DMPC encompasses a broad category of methods. Peng et al. (2024) provide an overview of various DMPC types specifically applied to swarm control. Among these variants, the approaches presented in Luis and Schoellig (2019); Luis et al. (2020); Park et al. (2022, 2023); Chen et al. (2023, 2024); Gräfe et al. (2022) form the backbone of our work. For brevity, we refer to these approaches simply as DMPC throughout the paper.

It is important to emphasize that alternative DMPC methods also exist. For instance, some approaches solve centralized optimization problems using distributed techniques (Stomberg et al., 2021, 2023; Camponogara et al., 2002). In contrast to the DMPC variant employed here, these methods typically confine information exchange to the local neighborhood of agents.

The considered DMPC variant originated from Augugliaro et al. (2012), which introduced a centralized Sequential Convex Programming approach for trajectory planning by linearizing nonlinear collision constraints and solving Quadratic Programs (QPs). Luis and Schoellig (2019) extended this method to distributed computation by assigning one QP per UAV; after solving their QPs and simulating a step forward, the swarm executes the final trajectories together. Luis et al. (2020) adapted the method for real-time control, synchronizing UAVs into periodic rounds. During such a round, each UAV first runs a QP. Afterwards, the UAVs exchange their solution in many-to-all communication rounds. In parallel, each UAV executes its initial trajectory segment.

Subsequent works extended DMPC to provide theoretical collision avoidance guarantees. Park et al. (2022) utilized Bernstein polynomials for trajectory representation, leveraging their convex hull properties and final state constraints to ensure collision avoidance, recursive feasibility, and static obstacle avoidance. Park et al. (2023) extended this approach to dynamic obstacles, mitigating deadlocks through heuristics that temporarily adjust target positions. Similarly, Chen et al. (2024) showed that soft constraints can prevent deadlocks with theoretical guarantees, which Chen et al. (2023) extended to include

static obstacles. Both use time-varying separating planes, the dynamic extension of Buffered Voronoi Cells (BVC) (Zhou et al., 2017), and final state constraints for collision avoidance and recursive feasibility. Concurrently, Gräfe et al. (2022) studied the feasibility of DMPC for nano UAV swarms by proposing to offload computations to ground agents and using event-triggered scheduling, however, without a distributed hardware implementation. Further, they employ time-varying BVCs with final state constraints for collision avoidance guarantees.

In addition to theoretical advancements, the presented works' hardware experiments have demonstrated DMPC's practical suitability. However, these rely on a *single* central computer that simulates distributed computation and communication, streaming position commands to the UAVs. This setup fails to capture real-world conditions like communication delays from limited communication bandwidth, message loss, and constrained computational power.

In contrast, we present a distributed hardware implementation. Building upon Gräfe et al. (2022), we employ ground-based distributed event-triggered computation. To ensure collision avoidance even in the presence of message loss, a common occurrence in wireless communication, we propose MLR-DMPC, an extension of DMPC. Furthermore, to effectively mitigate deadlocks, we incorporate soft constraints (Chen et al., 2024, 2023) and high-level planning heuristics (Park et al., 2022, 2023).

Finally, we distinguish these methods from others also termed DMPC (Peng et al., 2024). In Stomberg et al. (2021, 2023), for example, centralized optimization problems are solved using distributed techniques. While effective for controlling ground robots via distributed off-board computations, these methods require multiple communication steps per optimization run, making it too slow when running under limited communication bandwidth.

3 DMPC-SWARM — Architecture

This section presents DMPC-SWARM, providing an overview of how it enables communication (**C1**), distributed computation (**C2**), and safe control (**C3**). The subsequent Section 4 details the methodologies of DMPC-SWARM.
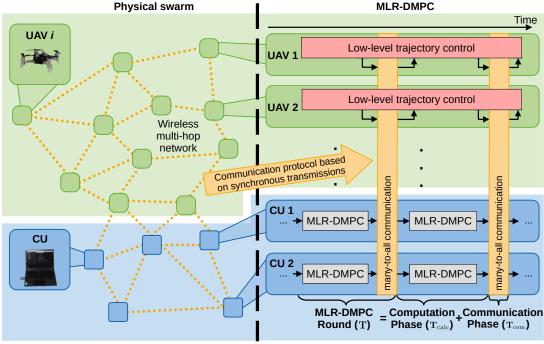
3.1 Communication

As described in Section 2, DMPC requires synchronized round-based many-to-all communication. However, the inherent unreliability of wireless networks, combined with the high dynamics of moving UAVs, makes this challenging.

Over the past decade, wireless mesh protocols based on synchronous transmissions have proven superior to traditional point-to-point routing approaches (Zimmerling et al., 2020). The key insight is that packet collisions from transmission that overlap in time, space and frequency, can be successfully decoded due to the capture effect (Leentvaar and Flint, 1976) and non-destructive interference (Herrmann and Zimmerling, 2023). This has two crucial implications: (*i*) Unlike traditional routing protocols, synchronous transmission methods do not need to track topology changes to avoid packet collisions, making them highly resilient to such changes. (*ii*) Synchronous transmissions enable distributed network nodes to achieve efficient time synchronization with sub-microsecond accuracy (Ferrari et al., 2011; Landsiedel et al., 2013), providing the foundation for real-time communication with formally proven end-to-end deadline guarantees (Zimmerling et al., 2017). As a result, protocols based on synchronous transmissions have enabled a range of powerful wireless control applications (Mager et al., 2019; Baumann et al., 2019; Mager et al., 2021; Trobinger et al., 2021).

In DMPC-SWARM, we build on this prior work by leveraging a synchronous transmissions based protocol called Mixer (Herrmann et al., 2018). Mixer provides many-to-all communication across dynamic wireless mesh networks, essential for DMPC. It does this also efficiently: Mixer achieves order-optimal scaling with the number of messages by integrating synchronous transmissions with random linear network coding (Ho et al., 2006). The network coding approach also provides additional reliability in real networks with fast-moving nodes like UAVs. This high robustness and reliability allows us to treat message losses as exceptional events when designing the control system in DMPC-SWARM.

In DMPC-SWARM, Mixer synchronizes the devices into discrete-time rounds indexed by k and with a fixed duration T (see Figure 1). Each round comprises a computation and a communication phase, which last T_{calc} and T_{com}, respectively. Devices begin computation simultaneously at $t = kT$ and begin to

Fig. 1: DMPC-SWARM overview. *Left: The physical swarm consisting of of UAVs and CUs connected via a wireless mesh network.* **Right:** *Swarm operations are structured in synchronized rounds, alternating between computation and many-to-all communication phases, facilitated by Mixer.*

communicate at $t = kT + T_{calc}$. During communication, devices concurrently send messages in dedicated, synchronized time slots. Mixer efficiently broadcasts these messages, all devices can receive the complete set after T_{com}. This mechanism provides the timely, reliable data exchange essential for MLR-DMPC.

3.2 Distributed Computation

Nano-UAV swarms face a fundamental conflict between limited onboard compute resources and the high computational demands of DMPC. To address this, we leverage the swarm's M CUs for heavy computations (cf. Section 1.1).

Although each CU can compute a trajectory for one UAV at once via DMPC, maintaining a one-to-one ratio of CUs to UAVs ($M = N$) is inefficient—it is costly, overloads the compute network, and demands high communication bandwidth. Therefore, DMPC-SWARM employs significantly fewer CUs ($M < N$). In each round k, the CUs compute new trajectories for M of the N UAVs, while the rest follow their previously planned paths. A distributed, priority-based

event trigger determines which UAVs need new trajectories and schedules computations on the CUs (see Section 4.2.2).

3.3 Overview of MLR-DMPC

DMPC-SWARM leverages Mixer's round-based structure of alternating computation and communication phases for MLR-DMPC (see Figure 1). A key innovation of MLR-DMPC over existing DMPCs is its ability to handle message loss. We provide a brief overview here; details are in Section 4.

To ensure effective DMPC, the CUs must be informed of all UAV actions, which Mixer's many-to-all communication facilitates. The CUs monitor received UAV activities using information trackers.

At the start of each computation phase, each CU checks whether its information trackers are up-to-date. If they are, the CU solves a QP for its assigned UAV and broadcasts the solution during the next communication phase. If crucial information is missing—indicating a critical message loss that prevents safe trajectory calculation—the CU does not

proceed. Instead, the CU identifies the missing information and requests it in the subsequent communication phase. Although such instances are rare due to Mixer's reliability, they must be addressed.

Consequently, UAVs receive only recursively feasible trajectories that ensure their safety—even if they do not receive an update due to message loss or not being selected by the trigger. The UAVs track these trajectories using high-frequency, low-level controllers common in DMPC (Park et al., 2022, 2023; Chen et al., 2023, 2024; Gräfe et al., 2022).

4 Distributed Model Predictive Control with Message Loss Recovery

This section details MLR-DMPC. After describing the UAV model forming the base for MLR-DMPC, we provide an in-depth explanation of MLR-DMPC and prove collision avoidance.

4.1 UAV Model

All N UAVs have nonlinear dynamics

$$\forall i \in \mathcal{A}: \quad \dot{x}_i(t) = f_i(x_i(t), u_i(t)) + v_i(t), \\ x_i(0) = x_{i,0}, \tag{3}$$

where $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state vector, $u_i \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input, $f_i : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, and $v_i \in \mathcal{V} \subseteq \mathbb{R}^n$ represents disturbances. The position of UAV i at time t is given by $p_i(t) = g_{\mathrm{p},i}(x_i(t))$, where $g_{\mathrm{p},i} : \mathcal{X} \to \mathcal{P} \subseteq \mathbb{R}^3$ maps the state to the position. Typically, x_i includes the position, making $g_{\mathrm{p},i}$ straightforward (e.g., $p_i(t) = [I, 0]x_i(t)$). For detailed quadcopter dynamics, see (Mahony et al., 2012; Antal et al., 2023; Panerati et al., 2021). Each UAV must navigate from its initial position $p_{i,\mathrm{init}} = g_{\mathrm{p},i}(x_{i,0})$ to its target $p_{i,\mathrm{target}}$ while avoiding collisions (Equation 2).

Instead of using the complex UAV dynamics f_i, we define a simpler nominal system for each UAV $i \in \mathcal{A}$, as common in (D)MPC (Augugliaro et al., 2012; Luis and Schoellig, 2019; Luis et al., 2020; Chen et al., 2024, 2023; Gräfe et al., 2022)

$$\dot{\hat{x}}_i = \hat{f}_i(\hat{x}_i, \hat{u}_i), \quad \hat{x}_i(0) = \hat{x}_{i,0}, \tag{4}$$

where $\hat{x}_i \in \hat{\mathcal{X}}$ is the nominal state with $\hat{x}_{i,0}$ its initial value, $\hat{u}_i \in \hat{\mathcal{U}}$ and $\hat{f}_i : \hat{\mathcal{X}} \times \hat{\mathcal{U}} \to \hat{\mathcal{X}}$. We define the function $\hat{g}_{\mathrm{p},i} : \hat{\mathcal{X}} \to \mathcal{P}$ that maps the nominal state to the nominal position $\hat{p}_i(t) = \hat{g}_{\mathrm{p},i}(\hat{x}_i(t))$.

We further design a controller $c_i : \mathcal{X} \times \hat{\mathcal{X}} \times \hat{\mathcal{U}} \to \mathcal{U}$ that controls the UAV

$$\dot{x}_i = f_i\big(x_i, c_i(x_i, \hat{x}_i, \hat{u}_i)\big) + v_i, \tag{5}$$

steering it towards the nominal state. In the following, we will not explicitly specify $\hat{f}_i$ and c_i, instead, we will make the following assumption similar to incremental asymptotic stability (Köhler et al., 2018).

Assumption 1. *We assume that for arbitrary inputs $\hat{u}_i \in \hat{\mathcal{U}} \subseteq \mathbb{R}^{\hat{m}}$, there exists a constant $\Delta d_{\min} \in \mathbb{R}_{\geq 0}$ such that if $\|\hat{p}_i(0) - p_i(0)\| = \|\hat{g}_{\mathrm{p},i}(\hat{x}_i(0)) - g_{\mathrm{p},i}(x_i(0))\| \leq \Delta d_{\min}$, then for all $t \in \mathbb{R}_{\geq 0}$:*

$$\|p_i(t) - \hat{p}_i(t)\|_2 \leq \Delta d_{\min}. \tag{6}$$

We can satisfy this assumption by selecting $\hat{f}_i$ as a fourth-order integrator in combination with an appropriate controller leveraging the differential flatness of quadcopters (Mellinger and Kumar, 2011). However, empirical studies show that second- or third-order integrators suffice for generating smooth, trackable trajectories while reducing QP complexity (Augugliaro et al., 2012; Luis and Schoellig, 2019; Luis et al., 2020; Chen et al., 2024, 2023; Gräfe et al., 2022). Therefore, DMPC-SWARM uses a third-order integrator for $\hat{f}_i$, with input $\hat{u}_i$ representing the jerk (Gräfe et al., 2022), balancing simplicity and trajectory smoothness. These trajectories are followed by a standard controller (Mellinger and Kumar, 2011) acting as c_i. As our theoretical analysis does not depend on a specific formulation of $\hat{f}_i$, we will continue with an arbitrary $\hat{f}_i$ without giving a specific formula.

During each round k, a UAV has a predicted reference trajectory $\hat{x}_i(\tau|k)$ and input $\hat{u}_i(\tau|k)$ of the nominal system (4) computed by a CU, where τ denotes the time within the prediction horizon. With these predictions, the UAV generates the references

$$\hat{x}_i(t) := \hat{x}_i(\tau|k), \tag{7}$$
$$\hat{u}_i(t) := \hat{u}_i(\tau|k), \tag{8}$$

and follows them using its low-level controller c_i at time $t = \tau + kT$:

$$u_i(t) = c_i\big[x_i(t), \hat{x}_i(\tau|k), \hat{u}_i(\tau|k)\big]. \tag{9}$$

When a UAV receives a new trajectory $\hat{u}_{i,w}$ from a CU w during the communication phase at round k, it updates its reference trajectory at the beginning of round $k + 1$. Otherwise, it reuses its old trajectory, either because no CU computed a new trajectory for it or because the UAV missed it due to message loss:

$$\hat{u}_i(\tau|k+1) \qquad\qquad\qquad (10)$$
$$= \begin{cases} \hat{u}_{i,w}(\tau + T|k) & \text{if received new tra-} \\ & \text{jectory from a CU } w, \\ \hat{u}_i(\tau + T|k) & \text{otherwise.} \end{cases}$$

4.2 MLR-DMPC Algorithm

Using MLR-DMPC, the CUs concurrently calculate the trajectories $\hat{u}_{i,w}$ for the UAVs. Building upon the brief overview presented in Section 3.3, we now present its details step by step going through Algorithm 1.

Each CU alternates between computation (Lines 8–32) and communication phases (Lines 33–36), with a state machine scheduling operations during computation.

A key component of the computation phase is the information tracker $\mathcal{D}_{iw}(k)$, which CU w uses to store trajectory candidates for UAV $i \in \mathcal{A}$ at time k, considering one trajectory as currently followed by the UAV. We mark trajectories saved in these trackers with a tilde symbol, e.g., $\tilde{x}$. For simplicity and brevity, we abuse the "element of $\mathcal{D}$" notation for states, inputs, and positions ambiguously, i.e., we note $\tilde{u} \in \mathcal{D}_{iw}(k), \tilde{x} \in \mathcal{D}_{iw}(k), \tilde{p} \in \mathcal{D}_{iw}(k)$, although $\tilde{u}$, $\tilde{x}$ and $\tilde{p}$ are of different types and dimensions. Due to the delay T between rounds, each trajectory in $\mathcal{D}_{iw}(k)$ begins at $k - 1$, e.g., $\tilde{u}_i(\cdot \mid k - 1) \in \mathcal{D}_{iw}(k)$.

At the start of each computation phase, the CU updates its information trackers (Line 8) using messages from the previous communication phase (Line 35). If it detects critical message loss, it marks the affected trackers as deprecated, indicating they may not contain the UAV's actual trajectory (details in Section 4.2.1).

If all information trackers are up-to-date, the CU executes a DMPC step: selecting a UAV i using the event trigger (Lines 14 and 15; Section 4.2.2) and solving an optimization problem (Line 18) to compute the new reference trajectory $\hat{x}_{i,w}(\tau \mid k)$ and $\hat{u}_{i,w}(\tau \mid k)$. The optimization problem uses the information trackers to formulate anti-collision constraints (details in Section 4.2.3).

If any information tracker is deprecated, the CU initiates MLR (Lines 22–32) by requesting a missing trajectory from a UAV (Line 25). In the next round, the UAV transmits its current reference trajectory, using the communication resources typically reserved for the CU because its own are occupied with state information and target position. Accordingly, the CU refrains from transmitting during this round to free communication resources for the UAV (Line 30).

At the end of each step, during the communication phase, the CU broadcasts a message containing the MLR-DMPC results and the computed UAV priorities for the next round's event trigger to all other CUs and UAVs (Line 35), unless it has donated its communication resources to a UAV.

4.2.1 Information-Tracker Updates

Algorithm 2 details how each CU updates its information trackers. It first extracts the UAVs' current reference trajectories from the received messages. Each UAV's message includes unique metadata, specifically, the calculation time and the responsible CU, about its current reference trajectory. The CU compares this metadata with that in its information trackers; if they match, it retains the trajectory as the one the UAV is following (Line 6).

If no new trajectory was planned for a UAV in the previous round, the stored trajectory is the current one. However, if a CU did plan a new trajectory, the UAV may be following either the new trajectory (if it received it) or the old one (if it missed the message). The CU therefore retains both trajectories in the information tracker (Line 22).

With this information, the CU checks for critical message loss. If it received fewer CU messages than the total number of CUs, it might lack the actual trajectory of an unknown UAV, so it marks all information trackers as deprecated (Line 18). If the CU did not receive a required message from a UAV with necessary metadata to determine which trajectory the UAV followed, it marks that UAV's information tracker as deprecated (Line 13).

4.2.2 Event-Trigger

The event-trigger aims to assign M of the N UAVs to the CUs in a distributed manner, avoiding a single point of failure, using priorities similar to Mager et al. (2021); Gräfe et al. (2022).

At each round $k - 1$, each CU w calculates a priority $J_{i,w}(k - 1)$ for every UAV i (see Algorithm 1,

Algorithm 1 MLR-DMPC running on CU w.

```
1:  k ← 0                                                                    ▷ Current round
2:  ∀i ∈ A 𝒟_iw(k) ← {}   𝒟_w(k) ← {𝒟_iw(k)|∀i ∈ A}              ▷ Init information-trackers as empty.
3:  ∀i ∈ A setDeprecated(𝒟_iw(k))
4:  state ← REQUEST_TRAJECTORY                                ▷ Because information-trackers are empty.
5:  UAV_messages, cu_messages ← ∅
6:  while true do
```
<span style="background:#d9c8d6">
```
7:  ───────────────────────── Trajectory-Tracker Update ─────────────────────────
8:      𝒟_w(k) ← updateInformationTrackers(UAV_messages, cu_messages, 𝒟_w(k − 1))        ▷ Algorithm 2
9:      if allInformationTrackersUpToDate() then state ← RUN_DMPC
10:     elif state == RUN_DMPC then state ← WAIT
11:     endif
```

<span style="background:#c9c9c9">
```
12: ───────────────────────────────── DMPC ─────────────────────────────────
13:     if state == RUN_DMPC then
14:         prios ← prioConsensus(cu_messages)                                        ▷ Event-trigger
15:         selected_UAV ← selectUAV(prios)
16:         tx_message ← EmptyMessage
17:         if |𝒟_selected_UAV,w(k)| == 1 then
18:             x̂_{i,w}(τ|k), û_{i,w}(τ|k) ← solveOptimization(selected_UAV, 𝒟_w(k))
19:             tx_message ← TrajectoryMessage(x̂_{i,w}(τ|k), û_{i,w}(τ|k))
20:         endif
```

<span style="background:#8fb3d9">
```
21: ───────────────────────────────── MLR ─────────────────────────────────
22:     elif state == WAIT then            ▷ Immediately after entering MLR, we do not exactly know
                                             which trajectories are unknown.
23:         tx_message ← EmptyMessage
24:         state ← REQUEST_TRAJECTORY
25:     elif state == REQUEST_TRAJECTORY then
26:         requested_UAV ← selectUAVWithDeprecatedInformationTracker(𝒟_w(k))
27:         tx_message ← TrajectoryRequestMessage(requested_UAV)
28:         state ← WAIT_FOR_UPDATE
29:     elif state == WAIT_FOR_UPDATE then
30:         tx_message ← ∅              ▷ Do not send anything, because requested UAV uses commu-
                                          nication resources.
31:         state ← REQUEST_TRAJECTORY
32:     endif
```

<span style="background:#f5c98a">
```
33: ───────────────────────────── Communication Phase ─────────────────────────────
34:     tx_message ← {tx_message, calcPrio(D_w(k))} if tx_message ! = ∅
35:     UAV_messages, cu_messages = wirelessBusRound(tx_message)
36:     k ← k + 1
```

```
37: endwhile
```

Line 34). Appendix A outlines three triggers with their priority calculation: round-robin (RR) selecting UAVs periodically; distance-based (DB) selecting UAVs farthest from their targets; and a hybrid trigger (HT) combining RR and DB, selecting UAVs based on both their distance to targets and time since last assigned to a CU.

Message loss (e.g., missed target positions) can cause discrepancies among CUs, leading them to compute different priorities for the same UAVs. To address this, CUs employ a consensus algorithm. During communication, they share their computed priorities. CUs unify them by taking the maximum across all CUs:

$$J_i(k) = \max_w J_{i,w}(k-1). \qquad (11)$$

9

Algorithm 2 Algorithm that updates the trajectory information-trackers of the MLR-DMPC.

1: **procedure** UPDATEINFORMATIONTRACKERS(UAV_messages, cu_messages, $\mathcal{D}_w(k-1)$)
2:      $\forall \mathcal{D}_{iw}(k-1) \in \mathcal{D}_w(k-1) : \tilde{\mathcal{D}}_{iw}(k) \leftarrow \mathcal{D}_{iw}(k-1)$
3:      **for** $(i, \text{message}) \in \text{enumerate}(\text{UAV\_messages})$ **do**   ▷ Which trajectories received in the last round
4:          **for** trajectory $\in \tilde{\mathcal{D}}_{iw}(k)$ **do**
5:              **if** trajectory.metadata == message.metadata **then**
6:                  $\tilde{\mathcal{D}}_{iw}(k) \leftarrow \{\text{trajectory}\}$
7:                  setInformationTrackerUpToDate(i)
8:                  **break**
9:              **end if**
10:          **end for**
11:      **end for**
12:      **if** $\exists i \in \mathcal{A}$ s.t. $|\tilde{\mathcal{D}}_{iw}(k)| > 1$ **then**
13:          setAllInformationTrackersDeprecated()
14:      **end if**
15:
16:      $\mathcal{D}_{iw}(k) \leftarrow \tilde{\mathcal{D}}_{iw}(k)$
17:      **if not** length(cu_messages) == M **then**
18:          setAllInformationTrackersDeprecated()
19:      **end if**
20:      **for** $(\text{message}, i) \in \text{enumerate}(\text{cu\_messages})$ **do**
21:          **if** type(message) **is** TrajectoryMessage **then**
22:              $\mathcal{D}_{iw}(k) \leftarrow \mathcal{D}_{iw}(k) \cup \{\text{message}\}$
23:          **end if**
24:      **end for**
25:      **return** $\{\mathcal{D}_{iw}(k)|\forall i \in \mathcal{A}\}$
26: **end procedure**

They sort the UAVs based on these unified priorities and store the top M UAVs in the set $\mathcal{A}_{\text{ET}}(k)$. Since all active CUs receive the same priority information, they compute the same set $\mathcal{A}_{\text{ET}}(k)$.

Each CU w selects the UAV at position $((k + w) \bmod M)$ in $\mathcal{A}_{\text{ET}}(k)$[1]. The set of UAVs for which new trajectories are computed is denoted by $\tilde{\mathcal{A}}_{\text{ET}}(k) \subseteq \mathcal{A}_{\text{ET}}(k)$, since some CUs may be in MLR mode and not compute a new trajectory.

4.2.3 Trajectory Calculation

The trajectory computed by CU w for UAV i has a piecewise-constant input with sampling time T_{s} and horizon h_{s}. The constant input from time $T_{\text{s}}\kappa$ till $T_{\text{s}}(\kappa+1)$ for $\kappa \in \{0, ..., h_{\text{s}}-1\}$ is denoted as $u_{i,w,\kappa|k}$. Formally, the trajectory of the system is

$$\hat{u}_{i,w}(\tau + T|k) = \sum_{\kappa=0}^{h_{\text{s}}-1} \Gamma_{T_{\text{s}}}(\tau - \kappa T_{\text{s}})u_{i,w,\kappa|k}, \quad (12)$$

where $\Gamma_{T_{\text{s}}}(t) = 1$ if $0 \leq t \leq T_{\text{s}}$ and $\Gamma_{T_{\text{s}}}(t) = 0$ otherwise (Gräfe et al., 2022). Hence, $\Gamma_{T_{\text{s}}}(\tau - \kappa T_{\text{s}})u_{i,w,\kappa|k} = u_{i,w,\kappa|k}$, between $T_{\text{s}}\kappa$ till $T_{\text{s}}(\kappa + 1)$,

---

[1]The modulo operation ensures each priority rank is selected, even when a CU misses multiple messages.

and it is zero otherwise. We require $\frac{T}{T_{\text{s}}} = \frac{h_{\text{s}}}{H}$, where $H \in \mathbb{N}$ is the prediction horizon, so the prediction extends to time $kT + HT + T$. An offset of T accounts for the latency of one round.

The CU aims to determine this input trajectory by minimizing the quadratic distance between the state trajectory and its target, scaled by positive definite matrices Q_i and R_i. This trajectory is constrained by the nominal UAV model $\hat{f}_i$ (Assumption 1). The initial state $\tilde{x}_i(2T|k-1)$ is the sole entry in $\mathcal{D}_{i,w}(k)$ (cf. Algorithm 1, Line 17). Both the input and state are confined to $\hat{\mathcal{U}}$ and $\hat{\mathcal{X}}$, respectively, thereby restricting the movement area of the UAV. Through an equality constraint, the CU ensures that the UAV halts at the end of the computed trajectory, thereby guaranteeing recursive feasibility of the optimization problem (Gräfe et al., 2022; Park et al., 2022; Chen et al., 2024, 2023). Lastly, to ensure collision-free trajectories, the CU imposes time-varying BVC constraints described below.

In summary, the CU solves:

$$\min_{\hat{u}_{i,w,\cdot|k}} \sum_{\kappa=0}^{h_{\text{o}}} \big[||\hat{x}_{i,w}(\kappa T_{\text{o}} + T|k) - \hat{x}_{i,\text{target}}||^2_{Q_i}$$
$$+ ||\hat{u}_{i,w}(\kappa T_{\text{o}} + T|k)||^2_{R_i} \big] \quad (13\text{a})$$

10

$$\text{s.t. } \dot{\hat{x}}_{i,w}(t|k) = \hat{f}_i(\hat{x}_{i,w}(t|k), \hat{u}_{i,w}(t|k)) \tag{13b}$$
$$\hat{x}_{i,w}(T|k) = \tilde{x}_i(2T|k-1)$$

$$\hat{u}_{i,w,\ell|k} \in \hat{\mathcal{U}} \ \forall \ell \in \{0, ..., h_{\text{s}} - 1\} \tag{13c}$$
$$\hat{x}_{i,w}(\kappa T_{\text{b}} + T|k) \in \hat{\mathcal{X}} \ \forall \kappa \in \{0, ..., h_{\text{b}}\} \tag{13d}$$
$$0 = \hat{f}_i(\hat{x}_{i,w}(HT + T|k), 0) \tag{13e}$$

$$\forall j \in \mathcal{A}\backslash\{i\} : \tag{13f}$$
$$A_{i,j,\text{c}} \begin{bmatrix} \hat{p}_{i,w}(T|k) \\ \hat{p}_{i,w}(T + T_{\text{c}}|k) \\ \vdots \\ \hat{p}_{i,w}(T + h_{\text{c}}T_{\text{c}}|k) \end{bmatrix} \leq b_{i,j,\text{c}}(\mathcal{D}_{j,w}(k))$$

where $\hat{x}_{i,\text{target}}$ satisfies $p_{i,\text{target}} = \hat{g}_{\text{p},i}(\hat{x}_{i,\text{target}})$.
Constraint (13f) prevents collisions between the UAVs. The matrices $A_{i,j,\text{c}}$ and $b_{i,j,\text{c}}$ are constructed using time-variant BVC (Gräfe et al., 2022; Van Parys and Pipeleers, 2017; Chen et al., 2024, 2023). However, we relax the constraints in case a UAV is not recomputed. The collision avoidance constraints are computed between the UAV i and every other UAV $j \in \mathcal{A}\backslash\{i\}$. The CU uses the reference positions $\hat{p}_i(\cdot|k-1)$ that UAV i is currently following (known because $\mathcal{D}_{i,w}(k)$ contains only one element, cf. Alg. 1 Line 17) and all trajectories $\tilde{p}_j(\cdot|k-1) \in \mathcal{D}_j(k)$ in the information-tracker for the other UAV j. Thus, the CU constrains the position of UAV i based on all trajectories which it guesses UAV j might fly. First, we define the difference vector

$$n_{ij}(hT_{\text{c}} + 2T|k-1) \tag{14}$$
$$= \Theta^{-1}[\tilde{p}_j(hT_{\text{c}} + 2T|k-1) - \hat{p}_i(hT_{\text{c}} + 2T|k-1)],$$

which is the normal vector of a plane spanned between the UAVs. UAV i is constrained to stay on its side of the plane

$$n_{0,ij}(hT_{\text{c}} + 2T|k-1)^T \Theta^{-1} \tag{15}$$
$$\times [\tilde{p}_j(hT_{\text{c}} + 2T|k-1) - \hat{p}_{i,w}(hT_{\text{c}} + T|k)]$$
$$\geq \begin{cases} \frac{1}{2}(\hat{d}_{\text{min}} \\ \quad + ||n_{ij}(hT_{\text{c}} + 2T|k-1)||_2) & \text{if } j \in \mathcal{A}_{\text{ET}}(k) \\ \hat{d}_{\text{min}} & \text{else} \end{cases}$$

with $n_{0,ij} = n_{ij}/||n_{ij}||_2$ and $\hat{d}_{\text{min}}$ as the minimum distance between UAVs' reference positions with some safety gap to d_{min} (cf. Section 4.3.2). If

UAV $j \in \mathcal{A}_{\text{ET}}(k)$, we span the plane midway between it and UAV i. Else, relaxing the constraints, we move this plane close to j, because j remains on its old trajectory.

In this work, we assume a UAV flying space devoid of static obstacles, such as buildings. However, incorporating such obstacles into MLR-DMPC is straightforward using established methods from other DMPCs (Park et al., 2022, 2023; Chen et al., 2023).

4.3 Theoretical Analysis

Our theoretical analysis of DMPC-SWARM is based on the system model derived under Assumption 1. We first analyze the information trackers $\mathcal{D}$, then demonstrate that the reference trajectories $\hat{p}$ are collision-free, and finally prove the same for the actual positions p.

4.3.1 Content of the Information Trackers

We present two properties of the information trackers $\mathcal{D}_{iw}(k)$, which follow directly from the construction of Algorithm 2 and are proven in Appendix C.

Lemma 1. *If the information-tracker $\mathcal{D}_{\text{iw}}(k)$ is not deprecated, then it contains the true trajectory $\hat{p}_i$, the UAV i is following.*

Lemma 2. *If after executing Algorithm 2 the two information-trackers $\mathcal{D}_{\text{iw}}(k)$ and $\mathcal{D}_{\text{iv}}(k)$ are not deprecated for different w, v, then $\mathcal{D}_{\text{iw}}(k) = \mathcal{D}_{\text{iv}}(k)$.*
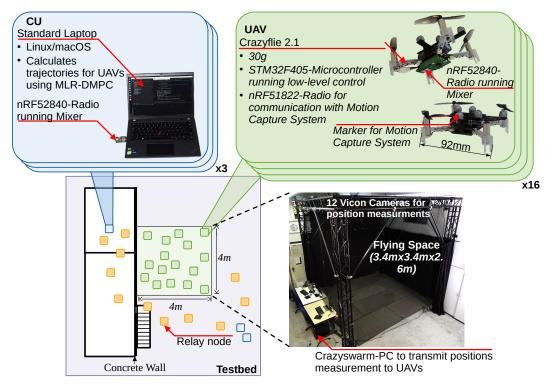
The CUs hence perform the DMPC step with the same information, which is crucial for guaranteeing collision avoidance. We thus write $\mathcal{D}_{\text{iw}}(k) = \mathcal{D}_{\text{iv}}(k) = \mathcal{D}_{\text{i}}(k)$ when a CU is not deprecated. In the following, we treat $\mathcal{D}_{\text{i}}(k)$ as the global knowledge shared by all CUs executing the DMPC. This simplifies the notation for subsequent derivations.

When all CUs are in the MLR state, we consider $\mathcal{D}_{\text{i}}(k)$ as the information tracker of a virtual CU that does not compute new trajectories but listens to all messages without message loss. Since no new trajectories are recomputed in this case, the information tracker remains unchanged.

4.3.2 Collision Avoidance of MLR-DMPC

The reference trajectories under constraints (13f) are collision free as Appendix C proves.

Corollary 1. *If $i, j \in \tilde{\mathcal{A}}_{\text{ET}}(k)$ and constraint (13f) is fulfilled for both i and j on CUs w and v, then for all*

Fig. 2: Hardware Implementation of DMPC-SWARM. *Sixteen Crazyflie 2.1 quadcopters operate in a space of $3.4m \times 3.4m \times 2.6m$. Three laptops serving as CUs compute trajectories using MLR-DMPC. The quadcopters and CUs communicate over a wireless mesh network with at least two hops using the Mixer protocol.*

$$h \in \{1, 2, \cdots, h_c\}$$

$$||\Theta^{-1}[\hat{p}_{j,v}(hT_c + T|k) - \hat{p}_{i,w}(hT_c + T|k)]||_2 \geq \hat{d}_{\min}. \tag{16}$$

Lemma 3. *If $i \in \tilde{\mathcal{A}}_{ET}(k)$, $j \notin \tilde{\mathcal{A}}_{ET}(k)$ and constraint (13f) is fulfilled, then for all $\tilde{p}_j \in \mathcal{D}_j(k+1)$ and for all $h \in \{1, 2, \cdots, h_c\}$*

$$||\Theta^{-1}[\tilde{p}_j(hT_c + T|k) - \hat{p}_{i,w}(hT_c + T|k)]||_2 \geq \hat{d}_{\min}. \tag{17}$$

Lemma 4. *If for all $i \notin \tilde{\mathcal{A}}_{ET}(k)$, $j \notin \tilde{\mathcal{A}}_{ET}(k)$, for all $\tilde{p}_j \in \mathcal{D}_j(k)$ and $\tilde{p}_i \in \mathcal{D}_j(k)$ and for all $h \in \{1, 2, \cdots, h_c\}$*

$$||\Theta^{-1}[\tilde{p}_j(hT_c + T|k-1) - \tilde{p}_i(hT_c + T|k-1)]||_2 \geq \hat{d}_{\min}, \tag{18}$$

then for all $\tilde{p}_j \in \mathcal{D}_j(k+1)$ and $\tilde{p}_i \in \mathcal{D}_i(k+1)$ and for all $h \in \{1, 2, \cdots, h_c\}$

$$||\Theta^{-1}[\tilde{p}_j(hT_c|k) - \tilde{p}_i(hT_c|k)]||_2 \geq \hat{d}_{\min}. \tag{19}$$

We now can prove collision avoidance of all reference trajectories at discrete timepoints.

Theorem 1. *If for all pairwise different $i, j \in \mathcal{A}$, $\forall \tilde{p}_i \in \mathcal{D}_i(k)$, $\forall \tilde{p}_j \in \mathcal{D}_j(k)$ and $\forall h \in \{0, 1, \cdots, h_c - 1\}$*

$$||\Theta^{-1}[\tilde{p}_j(hT_c + 2T|k-1) - \tilde{p}_i(hT_c + 2T|k-1)]||_2$$
$$\geq \hat{d}_{\min}, \tag{20}$$

and all $\tilde{x}_i \in \mathcal{D}_i(k)$ satisfy constraints (13b)–(13e), then the optimization problems (13) are feasible for arbitrary message loss. Additionally, for all pairwise different $i, j \in \mathcal{A}(k-1)$, $\forall \tilde{p}_i \in \mathcal{D}_i(k+1)$, $\forall \tilde{p}_j \in \mathcal{D}_j(k+1)$ and for all $h \in \{1, 2, \cdots, h_c\}$

$$||\Theta^{-1}[\tilde{p}_j(hT_c + T|k) - \tilde{p}_i(hT_c + T|k)]||_2 \geq \hat{d}_{\min} \tag{21}$$

and all $\tilde{p}_i$ also fulfill constraints (13b)–(13d).

Proof. We split the UAVs into two sets. The first set contains all UAVs that were not recomputed, and the second the UAVs that were recomputed. For the first set, we know that the shifted trajectories $\tilde{x}_i(t+T|k) =$

12

$\tilde{x}_i(t+2T|k-1)$ fulfill the constraints (13b)–(13d) for all $t < (H-1)T$, as T is a multiple of the sampling times T_s and T_b (Gräfe et al., 2022). For $t \geq (H-1)T$, it is $\tilde{x}_i(t+T|k) = \tilde{x}_i((H-1)T+T|k) = \tilde{x}_i((H-1)T+2T|k-1)$ and thus also (13b)–(13d) and (13e) are fulfilled.

For the UAVs that are recomputed, with the same argumentation, we can show that the shifted trajectory $\bar{x}_i(t+T|k) = \tilde{x}_i(t+2T|k-1)$ fulfills constraints (13b)–(13d). Also it holds that for all $j \in \mathcal{A}\backslash\{i\}$ and all $\tilde{p}_j \in \mathcal{D}_j(k)$

$$
\begin{aligned}
& n_{0,ij}(hT_c+2T|k-1)^T\Theta^{-1} \\
& \quad \times [\tilde{p}_j(hT_c+2T|k-1) - \bar{p}_i(hT_c+T|k)] \\
& = n_{0,ij}(hT_c+2T|k-1)^T\Theta^{-1} \\
& \quad \times [\tilde{p}_j(hT_c+2T|k-1) - \tilde{p}_i(hT_c+2T|k-1)] \\
& = ||\Theta^{-1}[\tilde{p}_j(hT_c+2T|k-1) \\
& \quad - \bar{p}_i(hT_c+2T|k-1)]||_2 \\
& \geq \frac{1}{2}(\hat{d}_{\min} + ||n_{ij}(hT_c+2T|k-1)||_2) \\
& \geq \hat{d}_{\min}. \quad (22)
\end{aligned}
$$

Hence, $\bar{x}$ also fulfills constraint (13f) and is a candidate solution optimization problem (13). The optimization problem is thus feasible.

Algorithm 2 then incorporates the solutions the CUs computed into the information-trackers of every CU after a communication step. First, it deletes some trajectories in the information-trackers (Lines 2–11). Because the remaining trajectories were already in the former information-tracker for CUs not in the MLR-state (no deprecated information-tracker), they fulfill the constraints (13b)–(13e) as argued above. In Lines 15–24, the algorithm adds new trajectories to the information-tracker, resulting in the information-trackers $\mathcal{D}_i(k+1), \forall i \in \mathcal{A}$ stored on the CUs with non-deprecated information-trackers. These also fulfill constraints (13b)–(13f) as they are solutions to the corresponding optimization problem.

Corollary 1, Lemma 3 and 4 then lead to equation (21). $\qquad\square$

From this, we can derive that the UAVs also fly on collision-free trajectories.

Theorem 2. *If the reference trajectories of all UAVs fulfill (13b)–(13e) at $k = 0$ and Equation (20), then for all k, arbitrary message loss, all pairwise different*

$i, j \in \mathcal{A}$ *and and for all* $h \in \{0, 1, \cdots, h_c - 1\}$

$$
||\Theta^{-1}[\hat{p}_j(hT_c+kT) - \hat{p}_i(hT_c+kT)]||_2 \geq \hat{d}_{\min}. \quad (23)
$$

Proof. First, we apply induction to Theorem 1, which is possible as the non-deprecated information-trackers are all equal to $\mathcal{D}_i$ due to Lemma 2, and T is a multiple of T_c (Gräfe et al., 2022). Thus, for pairwise different $i, j \in \mathcal{A}$, the information-tracker $\mathcal{D}_i$ contains trajectories, which are collision-free to all other trajectories in the information-tracker $\mathcal{D}_j$. Because of Lemma 1, each UAV follows one trajectory in the information-tracker, and its reference positions are collision-free. $\qquad\square$

Based on Assumption 1, we now derive guarantees for the actual UAV positions at continuous times t.

Theorem 3. *Let*

$$
\Delta d_{\min,\text{cont}} = \hat{d}_{\min} \quad (24)
$$
$$
- \min_{\substack{\hat{x}_{0,i}, \hat{x}_{0,j} \in \hat{\mathcal{X}} \\ \tau \in [0, T_c] \\ \hat{u}_i(t), \hat{u}_j(t) \in \hat{\mathcal{U}}}} ||\Theta^{-1}(\hat{p}_i(\tau) - \hat{p}_j(\tau))||_2
$$
$$
s.t. \quad \dot{\hat{x}}_i = \hat{f}_i(\hat{x}_i(t), \hat{u}_i(t)), \hat{x}_i(0) = \hat{x}_{0,i}
$$
$$
\dot{\hat{x}}_j = \hat{f}_i(\hat{x}_j(t), \hat{u}_j(t)), \hat{x}_j(0) = \hat{x}_{0,j}
$$
$$
\hat{p}_i(t) = g_i(\hat{x}_i(t))
$$
$$
\hat{p}_j(t) = g_i(\hat{x}_j(t))
$$
$$
||\Theta^{-1}(\hat{p}_i(0) - \hat{p}_j(0))||_2 \geq \hat{d}_{\min}
$$
$$
||\Theta^{-1}(\hat{p}_i(T_c) - \hat{p}_j(T_c))||_2 \geq \hat{d}_{\min}.
$$
$$
(25)
$$

If

$$
\hat{d}_{\min} \geq d_{\min} + \Delta d_{\min} + \hat{d}_{\min,\text{cont}}, \quad (26)
$$

then Equation (2) holds, i.e., the UAVs are collision free.

Proof. Follows directly from Assumption (2), construction of $\Delta d_{\min,\text{cont}}$ and triangle inequality. $\qquad\square$

For collision avoidance guarantees, $\Delta d_{\min}$ and $\Delta d_{\min,\text{cont}}$ must be known. Solving optimization problem (24) directly yields $\Delta d_{\min,\text{cont}}$, while $\Delta d_{\min}$ must be determined through experiments or engineering intuition.

Remark 1. *Under strong external disturbances like gusts, Assumption 1 may not hold. If a UAV's state deviates significantly from the reference, we reinitialize its current state and assign a high priority J_i to ensure replanning (Luis et al., 2020). However, under*

these conditions, the guarantees of MLR-DMPC do not hold, as such disturbances may lead the UAV onto a collision course.

Remark 2. *Theorem 3 requires collision-free initial trajectories. In practice, UAVs typically start their flight by hovering while maintaining a safe distance from one another. Therefore, the initial trajectories can often be initialized as this constant hovering state.*

5 Experimental Results

We demonstrate DMPC-SWARM's ability to safely control real UAV swarms using DMPC. Specifically, we present:

1. The overall performance of DMPC-SWARM as the first implementation of DMPC with distributed computation and wireless communication (Section 5.2).
2. An evaluation of DMPC-SWARM's distributed event-triggered computation architecture (Section 5.3).
3. DMPC-SWARM's robustness under message loss (Section 5.4).

5.1 Experimental Setup

Using the described architecture as a base, we achieve a distributed DMPC implementation (see Figure 2). It includes 16 Crazyflie 2.1 quadcopters and three laptops as CUs, which could correspond to end-user devices in an application scenario. All devices are equipped with nRF52840 $2\,\mathrm{Mbit\,s^{-1}}$ BLE transceivers running Mixer.

The swarm operates within a $3.4 \times 3.4 \times 2.6\,\mathrm{m}$ obstacle-free space. A Vicon motion capture system with Crazyswarm (Preiss et al., 2017) provides position measurements, which in real application could be substituted by distributed solutions such as Ultra-Wideband or GPS localization.

We chose $\hat{d}_{\min} = 0.25\,\mathrm{m}$ through experimentation. We began with $\hat{d}_{\min} = 0.35\,\mathrm{m}$ from Luis and Schoellig (2019). Through further testing, we found that reducing the distance to $\hat{d}_{\min} = 0.25\,\mathrm{m}$ allowed the swarm to operate reliably. Distances smaller than this proved unfeasible not due to physical collisions between UAVs, but because rotor-generated turbulence reduced lift, leading to drones crashing into the ground unexpectedly.

By positioning the CUs approximately 15 meters apart and separating them with a concrete wall to prevent direct communication, the network has at least two hops. The MLR-DMPC frequency is $5\,\mathrm{Hz}$ ($T_{\mathrm{calc}} = 105\,\mathrm{ms}$, $T_{\mathrm{com}} = 95\,\mathrm{ms}$), while the Crazyflies' low-level control runs at $500\,\mathrm{Hz}$.

5.2 Performance of DMPC-SWARM

Our video (http://tiny.cc/DMPCSwarm) showcases key experimental outcomes, including various swarm maneuvers. It demonstrates that DMPC-SWARM successfully implements DMPC-based swarm control in hardware, executing the entire DMPC algorithm in a distributed manner across the quadcopters and CUs.

Despite challenges such as changing wireless properties due to robot movement, communication among up to 27 devices in a mesh network, and interference from external Wi-Fi and Bluetooth devices, the communication network remains functional and all devices stay synchronized. The event-trigger mechanism enables dynamic swarm control while conserving resources, even when there are more drones than CUs. Importantly, no collisions occur during any maneuvers, despite network latency and message loss.
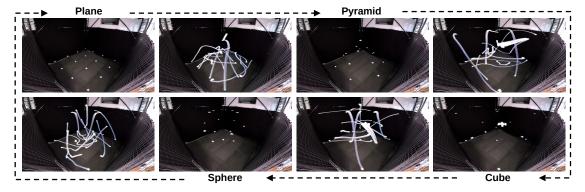
5.3 Distributed Event-Triggered Computation

We evaluated our approach through two experiments: (1) examining the influence of the number of CUs (i.e., available computational power), and (2) assessing the effect of the event trigger on swarm performance. Both experiments used the same setup (Figure 3a): 16 quadcopters underwent multiple formation changes, starting from a planar configuration and sequentially forming a pyramid, cube, sphere, and finally returning to the original plane. Results are presented in Figure 3 and in a video (http://tiny.cc/DMPCSwarmComputation).
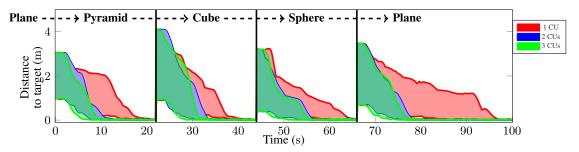
Number of CUs We control the swarm using one, two, and three CUs. As shown in Figure 3b, increasing the number of CUs accelerates the formation changes. The most significant improvement occurs when increasing from one to two CUs, while adding a third CU yields diminishing returns. Thus, adding computational power enhances swarm performance.

Using fewer CUs conserves resources but reduces performance; the extent of this trade-off depends on the number of CUs.
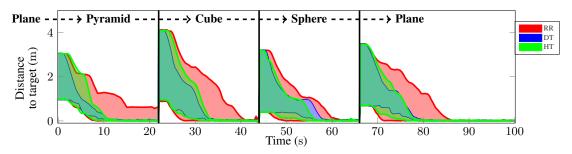
Event-Trigger We can influence swarm performance by choosing how event-trigger priorities are calculated

(a) The swarm's maneuvers in the experiments.

(b) Performance of the swarm with respect to the number of CUs.

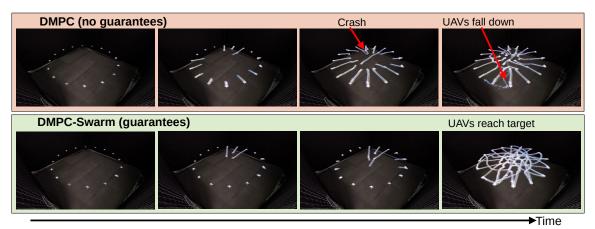(c) Performance of the swarm with respect to the event-trigger.

Fig. 3: Hardware experiment results on distributed computation. *(a) Visualization of flown formations. (b) Swarm performance improves with more CUs, illustrating the trade-off between resource efficiency and performance. (c) Comparison of event triggers: RR periodically selects all UAVs; DT selects based on proximity to targets; HT combines both. RR performs worst overall, while DT and HT each excel in different scenarios. A video of the experiments is available at http://tiny.cc/DMPCSwarmComputation.*

(cf. Section 4.2.2) (Gräfe et al., 2022). We compare RR, DB and HT.

Figure 3c presents the results. In three of the four formations, the swarm using RR is the slowest to reach targets because RR often assigns resources to agents already at their targets. DB and HT generally exhibit similar performance across formations. With

HT, some quadcopters reach targets earlier than others; with DB, all quadcopters reach targets at approximately the same time, as DB exclusively assigns resources to those farthest from their targets.

The event-trigger enables dynamic control of the swarm, even with many more drones than CUs. Selecting the appropriate event-trigger is crucial for achieving optimal performance.

15

Fig. 4: Comparison of MLR-DMPC with DMPC ignoring message loss. *Sixteen UAVs fly to the opposite side of a circle. We jam the wireless channel for the first two seconds. Using existing methods (Gräfe et al., 2022) that ignore message loss, the UAVs crash. With MLR-DMPC, the UAVs remain safe. A video of the maneuver is available at http://tiny.cc/DMPCSwarmMessageLoss.*

5.4 Robustness under Message Loss

To highlight the importance of DMPC-Swarm's collision avoidance under message loss, we compare MLR-DMPC with a DMPC that does not account for message loss (Gräfe et al., 2022).

In this experiment, sixteen quadcopters controlled by three CUs exchange their positions along a circle's circumference (see Figure 4 and http://tiny.cc/DMPCSwarmMessageLoss). To make our experiment reproducable, we simulate a jammed communication channel, introducing a two-second period during which the CUs experience message loss—they can transmit but cannot receive data.

We conduct the experiment twice: first with CUs running MLR-DMPC, then with standard DMPC. With standard DMPC, two quadcopters collide, causing the swarm to fail its task. In contrast, under MLR-DMPC control, all quadcopters remain safe and successfully exchange positions without collision, demonstrating the critical role of our MLR module.

We note that DMPC-Swarm would have been safe for arbitrarily long message-loss periods. In this case, the UAVs would have come safely to a halt after the length of the MPC's prediction horizon.

6 Conclusion

We introduced DMPC-Swarm, the first hardware implementation of DMPC-based swarm control with distributed computation and wireless mesh communication. DMPC-Swarm employs the communication protocol Mixer, showing that its synchronized many-to-all communication structure, robust to rapid device movement, is ideal for realizing DMPC. DMPC-Swarm uses ground-based, event-triggered distributed computations on CUs, enabling flexible scaling with the number of UAVs and efficient resource usage, even with limited bandwidth and hardware resources. Finally, we developed a novel DMPC algorithm that provably prevents collisions even under message loss. Our hardware experiments demonstrate that this combination enables DMPC on nano UAV swarms for the first time.

Our findings on the suitability of synchronous transmission protocols for UAV swarm communication extend beyond DMPC. Future research could explore the benefits of this communication architecture for other UAV swarm control methods.

Although our distributed control approach is efficient for small and medium-sized swarms, it does not readily scale to huge swarms of hundreds to thousands of agents. First, the communication rounds would take too long due to the many-to-all structure. Second, solving the optimization problems would also take a long time as MLR-DMPC includes all UAVs in anti-collision constraints. In such large-scale systems, a many-to-all communication approach is not necessary. For example, Chen et al. (2024) showed that for DMPC, UAVs only need to consider other UAVs in their neighborhood. Although directly integrating this

16

idea into MLR-DMPC is feasible, there is currently no efficient means of achieving local many-to-many communication via synchronous transmissions, which is an interesting challenge for future work.

Another interesting future challenge is to eliminate Assumption 1. For strong external disturbances, like sudden gusts, this Assumption 1 is most likely not valid or too conservative (i.e., $\Delta d_{\min}$ would be very large). One possible way to eliminate this assumption would be to include recent event-triggered robust MPC (Gräfe and Trimpe, 2025).

Acknowledgments

References

Antonelli, G.: Interconnected dynamic systems: An overview on distributed control. IEEE Control Systems Magazine **33**(1), 76–88 (2013)

Antal, P., Péni, T., Tóth, R.: Modelling, identification and geometric control of autonomous quadcopters for agile maneuvering. arXiv preprint arXiv:2306.09651 (2023)

Augugliaro, F., Schoellig, A.P., D'Andrea, R.: Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. IEEE/RSJ international conference on Intelligent Robots and Systems (2012)

Blender, T., Buchner, T., Fernandez, B., Pichlmaier, B., Schlegel, C.: Managing a mobile agricultural robot swarm for a seeding task. IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society (2016)

Baumann, D., Mager, F., Jacob, R., Thiele, L., Zimmerling, M., Trimpe, S.: Fast feedback control over multi-hop wireless networks with mode changes and stability guarantees. ACM Transactions on Cyber-Physical Systems (2019)

Baumann, D., Mager, F., Wetzker, U., Thiele, L., Zimmerling, M., Trimpe, S.: Wireless control for smart manufacturing: Recent approaches and open challenges. Proceedings of the IEEE (2020)

Cantizani-Estepa, J., Bravo-Arrabal, J., Fernández-Lozano, J., Fortes, S., Barco, R., García-Cerezo, A., Mandow, A.: Bluetooth low energy for close detection in search and rescue missions with robotic platforms: An experimental evaluation. IEEE Access (2022)

Chen, Y., Guo, M., Li, Z.: Deadlock resolution and recursive feasibility in mpc-based multi-robot trajectory generation. IEEE Transactions on Automatic Control (2024)

Camponogara, E., Jia, D., Krogh, B.H., Talukdar, S.: Distributed model predictive control. IEEE control systems magazine **22**(1), 44–52 (2002)

Chung, S.-J., Paranjape, A.A., Dames, P., Shen, S., Kumar, V.: A survey on aerial swarm robotics. IEEE Transactions on Robotics (2018)

Couceiro, M.S., Portugal, D., Rocha, R.P.: A collective robotic architecture in search and rescue scenarios. Proceedings of the 28th Annual ACM Symposium on Applied Computing, 64–69 (2013)

Cheraghi, A.R., Shahzad, S., Graffi, K.: Past, present, and future of swarm robotics. Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 3 (2022)

Chriki, A., Touati, H., Snoussi, H., Kamoun, F.: Fanet: Communication, mobility models and security issues. Computer Networks (2019)

Chen, Y., Wang, C., Guo, M., Li, Z.: Multi-robot trajectory planning with feasibility guarantee and deadlock resolution: An obstacle-dense environment. IEEE Robotics and Automation Letters (2023)

Duisterhof, B.P., Krishnan, S., Cruz, J.J., Banbury, C.R., Fu, W., Faust, A., Croon, G.C., Reddi, V.J.: Tiny robot learning (tinyrl) for source seeking on a nano quadcopter. (2021)

Dorigo, M., Theraulaz, G., Trianni, V.: Reflections on the future of swarm robotics. Science Robotics (2020)

Ferrari, F., Zimmerling, M., Thiele, L., Saukh, O.: Efficient network flooding and time synchronization with glossy. Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, 73–84 (2011)

Gago, J., Estrany, J., Estes, L., Fernie, A.R., Alorda, B., Brotman, Y., Flexas, J., Escalona, J.M., Medrano, H.: Nano and micro unmanned aerial vehicles (uavs): a new grand challenge for precision agriculture? Current protocols in plant biology (2020)

Gräfe, A., Eickhoff, J., Trimpe, S.: Event-triggered and distributed model predictive control for guaranteed collision avoidance in uav swarms (2022). 9th IFAC Conference on Networked Systems NECSYS 2022

Gugan, G., Haque, A.: Path planning for autonomous drones: Challenges and future directions. Drones (2023)

Gräfe, A., Trimpe, S.: Event-triggered robust model predictive control under hard computation resource constraints. arXiv preprint arXiv:2504.19540 (2025)

Ge, X., Yang, F., Han, Q.-L.: Distributed networked control systems: A brief overview. Information Sciences (2017)

Hu, J., Bruno, A., Zagieboylo, D., Zhao, M., Ritchken, B., Jackson, B., Chae, J.Y., Mertil, F., Espinosa, M., Delimitrou, C.: To centralize or not to centralize: A tale of swarm coordination. arXiv preprint arXiv:1805.01786 (2018)

Ho, T., Médard, M., Koetter, R., Karger, D.R., Effros, M., Shi, J., Leong, B.: A Random Linear Network Coding Approach to Multicast. IEEE Transactions on Information Theory (2006)

Herrmann, C., Mager, F., Zimmerling, M.: Mixer: Efficient many-to-all broadcast in dynamic wireless mesh networks. 16th ACM Conference on Embedded Networked Sensor Systems (2018)

Herrmann, C., Zimmerling, M.: Rssispy: Inspecting concurrent transmissions in the wild. In: Proceedings of the 2022 International Conference on Embedded Wireless Systems and Networks (2023)

Kang, C.-k., Fahimi, F., Griffin, R., Landrum, D.B., Mesmer, B., Zhang, G., Lee, T., Aono, H., Pohly, J., McCain, J., et al.: Marsbee-swarm of flapping wing flyers for enhanced mars exploration. Technical report (2019)

Köhler, J., Müller, M.A., Allgöwer, F.: A novel constraint tightening approach for nonlinear robust model predictive control. Annual American control conference (ACC) (2018)

Khan, M.A., Qureshi, I.M., Khanzada, F.: A hybrid communication scheme for efficient and low-cost deployment of future flying ad-hoc network (fanet). Drones (2019)

Karagüzel, T.A., Retamal, V., Cambier, N., Ferrante, E.: From shadows to light: A swarm robotics approach with onboard control for seeking dynamic sources in constrained environments. IEEE Robotics and Automation Letters (2023)

Leentvaar, K., Flint, J.H.: The Capture Effect in FM Receivers. IEEE Transactions on Communications (1976)

Landsiedel, O., Ferrari, F., Zimmerling, M.: Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale. Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (2013)

Lupashin, S., Hehn, M., Mueller, M.W., Schoellig, A.P., Sherback, M., D'Andrea, R.: A platform for aerial robotics research and demonstration: The flying machine arena. Mechatronics (2014)

Luis, C.E., Schoellig, A.P.: Trajectory generation for multiagent point-to-point transitions via distributed model predictive control. IEEE Robotics and Automation Letters (2019)

Laneman, J.N., Tse, D.N.C., Wornell, G.W.: Cooperative diversity in wireless networks: Efficient protocols and outage behavior. IEEE Transactions on Information Theory (2004)

Luis, C.E., Vukosavljev, M., Schoellig, A.P.: Online trajectory generation with distributed model predictive control for multi-robot motion planning. IEEE Robotics and Automation Letters (2020)

Liu, J., Xiang, J., Jin, Y., Liu, R., Yan, J., Wang, L.: Boost precision agriculture with unmanned aerial vehicle remote sensing and edge intelligence: A survey. Remote Sensing (2021)

Lv, C., Yu, R., Cao, J., Gong, C., Wu, W., Wang, X.: A survey on unmanned aerial vehicle swarm communication and navigation. Advances in Guidance, Navigation and Control: Proceedings of 2022 International Conference on Guidance, Navigation and Control (2023). Springer

Mager, F., Baumann, D., Herrmann, C., Trimpe, S., Zimmerling, M.: Scaling Beyond Bandwidth Limitations: Wireless Control With Stability Guarantees Under Overload. ACM Trans. Cyber-Phys. Syst. (2021) 2104.07989

Mager, F., Baumann, D., Jacob, R., Thiele, L., Trimpe, S., Zimmerling, M.: Feedback control goes wireless: Guaranteed stability over low-power multi-hop networks. 10th ACM/IEEE International Conference on Cyber-Physical Systems (2019)

Mellinger, D., Kumar, V.: Minimum snap trajectory generation and control for quadrotors. (2011)

Mahony, R., Kumar, V., Corke, P.: Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. IEEE robotics & automation magazine (2012)

Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I.W., Floreano, D., Deneubourg, J.-L., Nolfi, S., Gambardella, L.M., Dorigo, M.: Swarm-bot: A new distributed robotic concept. Autonomous robots (2004)

Namdev, M., Goyal, S., Agarwal, R.: An optimized communication scheme for energy efficient and secure flying ad-hoc network (fanet). Wireless Personal Communications (2021)

Oubbati, O.S., Atiquzzaman, M., Lorenz, P., Tareque, M.H., Hossain, M.S.: Routing in flying ad hoc networks: Survey, constraints, and future challenge perspectives. IEEE Access (2019)

Preiss, J.A., Honig, W., Sukhatme, G.S., Ayanian, N.: Crazyswarm: A large nano-quadcopter swarm. IEEE International Conference on Robotics and Automation (ICRA) (2017)

Park, J., Kim, D., Kim, G.C., Oh, D., Kim, H.J.: Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor. IEEE Robotics and Automation Letters (2022)

Park, J., Lee, Y., Jang, I., Kim, H.J.: Dlsc: Distributed multi-agent trajectory planning in maze-like dynamic environments using linear safe corridor. IEEE Transactions on Robotics (2023)

Peng, Y., Yan, H., Rao, K., Yang, P., Lv, Y.: Distributed model predictive control for unmanned aerial vehicles and vehicle platoon systems: a review. Intelligence & Robotics (2024)

Panerati, J., Zheng, H., Zhou, S., Xu, J., Prorok, A., Schoellig, A.P.: Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. IEEE/RSJ International Conference on Intelligent Robots and Systems (2021)

R Shamshiri, R., Weltzien, C., Hameed, I.A., J Yule, I., E Grift, T., Balasundram, S.K., Pitonakova, L., Ahmad, D., Chowdhary, G.: Research and development in agricultural robotics: A perspective of digital farming (2018)

Shah, K., Ballard, G., Schmidt, A., Schwager, M.: Multidrone aerial surveys of penguin colonies in antarctica. Science Robotics (2020)

Sankaranarayanan, V.N., Damigos, G., Seisa, A.S., Satpute, S.G., Lindgren, T., Nikolakopoulos, G.: Paced-5g: Predictive autonomous control using edge for drones over 5g. (2023)

Stomberg, G., Engelmann, A., Faulwasser, T.: A distributed active set method for model predictive control. IFAC-PapersOnLine (2021)

Stomberg, G., Ebel, H., Faulwasser, T., Eberhard, P.: Cooperative distributed mpc via decentralized real-time optimization: Implementation results for robot formations. Control Engineering Practice (2023)

Saska, M., Vonásek, V., Chudoba, J., Thomas, J., Loianno, G., Kumar, V.: Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. Journal of Intelligent & Robotic Systems (2016)

Staudinger, E., Zhang, S., Pöhlmann, R., Dammann, A.: The role of time in a robotic swarm: A joint view on communications, localization, and sensing. IEEE Communications Magazine (2021)

Trobinger, M., Albuquerque Gleizer, G., Istomin, T., Mazo, M., Murphy, A.L., Picco, G.P.: The wireless control bus: Enabling efficient multi-hop event-triggered control with concurrent transmissions. ACM Trans. Cyber-Phys. Syst. (2021)

Trianni, V., IJsselmuiden, J., Haken, R.: The saga concept: swarm robotics for agricultural applications. Technical report (2016)

Tavasoli, S., Pan, X., Yang, T.: Real-time autonomous indoor navigation and vision-based damage assessment of reinforced concrete structures using low-cost nano aerial vehicles. Journal of Building Engineering (2023)

Van Parys, R., Pipeleers, G.: Distributed model predictive formation control with inter-vehicle collision avoidance. 11th Asian Control Conference (ASCC) (2017). IEEE

Walter, A., Finger, R., Huber, R., Buchmann, N.: Smart farming is key to developing sustainable agriculture. Proceedings of the National Academy of Sciences (2017)

Wang, L., Huang, W., Li, H., Li, W., Chen, J., Wu, W.: A review of collaborative trajectory planning for multiple unmanned aerial vehicles. Processes (2024)

Yu, D., Li, J., Wang, Z., Li, X.: An overview of swarm coordinated control. IEEE Transactions on Artificial Intelligence (2023)

Zimmerling, M., Mottola, L., Kumar, P., Ferrari, F., Thiele, L.: Adaptive real-time communication for wireless cyber-physical systems. ACM Trans. Cyber-Phys. Syst. (2017)

Zimmerling, M., Mottola, L., Santini, S.: Synchronous transmissions in low-power wireless: A survey of communication protocols and network services. ACM Computing Surveys (CSUR) (2020)

Zhang, S., Pöhlmann, R., Wiedemann, T., Dammann, A., Wymeersch, H., Hoeher, P.A.: Self-aware swarm navigation in autonomous exploration missions. Proceedings of the IEEE (2020)

Zhou, D., Wang, Z., Bandyopadhyay, S., Schwager, M.: Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. IEEE Robotics and Automation Letters (2017)

Appendix A Event Triggers

To minimize communication overhead in the event-trigger mechanism, we quantize priorities to 8-bit unsigned integers instead of 32-bit floats. Due to the one-round delay, a recalculated UAV's priority may not reflect its updated state. To address this, if a CU has just recalculated UAV i, it sets its priority to zero. We adjust the maximum operation in Equation (11) to return zero if any element is zero; otherwise, it returns the maximum. The CU calculates priorities using the following equations.

A.1 Round-Robin Event Trigger (RR)

The RR calculates priorities as:

$$J_{iw}(k) = k - k_{i,\mathrm{calc}}(k), \tag{A1}$$

where $k_{i,\mathrm{calc}}(k)$ is the last round in which UAV i's trajectory was calculated. If the CU's database contains multiple trajectories, it uses the first one.

A.2 Distance-Based Event Trigger (DT)

The DT calculates priorities as:

$$J_{iw}(k) = \left\| p_{i,\mathrm{target}} - p_i\big(T \,|\, (k-1)T\big) \right\|. \tag{A2}$$

A.3 Hybrid Event Trigger (HT)

The HT combines the previous methods:

$$J_{iw}(k) = \big\| p_{i,\text{target}} - p_i\big(T \,|\, (k-1)T\big)\big\|$$
$$\times [k - k_{i,\text{calc}}(k)]. \tag{A3}$$

A.4 Deadlock-Aware Triggering

Since recalculating a deadlocked UAV's trajectory will not change it, we introduce a deadlock-aware triggering mechanism. The CU uses (Chen et al., 2024, Theorem 1, Equation (13)) to detect deadlocks. If a UAV is in deadlock, the CU sets its priority to one.

Appendix B Deadlock Avoidance

The main idea behind these constraints is to incorporate rotation into the swarm constraints (Chen et al., 2023). To achieve this, we modify the left side of constraint (15):

$$n_{0,ij}(hT_{\text{c}} + 2T|k-1)^T \Theta^{-1} \tag{B4}$$
$$\times [\tilde{p}_j(hT_{\text{c}} + 2T|k-1) - \hat{p}_{i,w}(hT_{\text{c}} + T|k)]$$
$$\geq \begin{cases} \frac{1}{2}(\hat{d}_{\min} \\ \quad + ||n_{ij}(hT_{\text{c}} + 2T|k-1)||_2) & \text{if } j \in \mathcal{A}_{\text{ET}}(k) \\ \quad + \epsilon \\ \hat{d}_{\min} + \epsilon & \text{else} \end{cases},$$

where $\epsilon \in \mathbb{R}_{\geq 0}$ is an additional optimization variable. $\epsilon \geq 0$ hereby ensures that the constraints in the limit of $\epsilon = 0$ are the same as the constraints (15), ensuring that the guarantees still hold Chen et al. (2023)

Following Chen et al. (2023), we increase the weight of ϵ in the objective function when UAV j is to the right of UAV i, pushing each UAV to its right and causing approaching UAVs to rotate around each other. Unlike Chen et al. (2023), we keep the soft constraint hyperparameters constant, as they perform well without adaptation.

B.1 High-Level Path Planner

We combine the soft constraints with a high-level planner similar to Park et al. (2022). In normal operation, the desired target state $x_{i,\text{target}}$ in optimization problem (13) equals the UAV's actual target. When a deadlock occurs—detected when all velocities fall below a threshold—a CU activates the high-level planner, and $x_{i,\text{target}}$ is set to an intermediate target calculated by the planner.

Each CU runs its portion of the high-level planner for its assigned subset of UAVs in parallel with the communication round, avoiding additional computation time. The intermediate target positions are communicated in the subsequent communication round.

The high-level planner classifies UAVs into those continuing towards their targets and those that should make room for others. For each assigned UAV i, the CU determines if it should make room for another UAV j based on the following conditions:

- UAV i is within a certain distance of UAV j,
- UAV j is not closer to its target than UAV i,
- UAV i is moving towards UAV j, or UAV i lies between UAV j and its target.

The CU notes all UAVs for which UAV i should make room and adjusts UAV i's path for the closest such UAV. Following Park et al. (2022), UAV i sets its new target to its current position plus a vector pointing away from UAV j, adding random noise to resolve symmetric deadlocks.

UAVs follow these intermediate targets until the deadlock resolves or the conditions necessitating making room no longer apply. The high-level planner then stops, and all UAVs resume flying to their actual target positions.

Appendix C Additional Proofs

Proof of Lemma 1. If UAV i was not recalculated in the previous round, then $|\mathcal{D}_{iw}| = 1$. If the CU received the trajectory metadata, it directly knows the trajectory UAV i is following. If the metadata was not received due to message loss, since the database is not deprecated (it would have been marked otherwise), the CU still knows the trajectory from prior rounds.

If UAV i was recalculated in the previous round, then at Line 18 in Algorithm 2, the database contains the trajectory UAV i was following two rounds ago. After processing, it includes this trajectory and the new one calculated and transmitted in the last round. Depending on whether UAV i received the new trajectory, it is following one of these two trajectories. Thus, the CU knows the possible trajectories UAV i may be following. □

Proof of Lemma 2. For UAVs not recalculated in the last round, the intermediate databases $\tilde{\mathcal{D}}_{iv}(k)$ and $\tilde{\mathcal{D}}_{iw}(k)$ contain their actual trajectories (Lines 2–11, Lemma 1). For UAVs recalculated in the last round, the databases contain their trajectories from the last or second-to-last round, depending on recalculations

due to the event trigger. After updating $\tilde{\mathcal{D}}$, the newly calculated trajectories are added (Lines 15–24). The databases are not deprecated only if all new trajectories are received, ensuring $\mathcal{D}_{iw}(k) = \mathcal{D}_{iv}(k)$. □

Proof of Corrolary 1. As both UAVs are selected for computation, we know $|\mathcal{D}_j| = 1$. The corollary then follows from Lemma 1 in (Gräfe et al., 2022). □

Proof of Lemma 3. We know that because UAV j was not recalculated $\tilde{p}_j(\cdot|k) = p_j(\cdot+T|k-1) \in \mathcal{D}_{\mathrm{j}}(k-1)$ (Equation (10)). It is

$$
\begin{aligned}
||\Theta^{-1}&[\tilde{p}_j(hT_{\mathrm{c}} + T|k) - \hat{p}_{i,w}(hT_{\mathrm{c}} + T|k)]||_2 \\
&= ||n_{0,ij}(hT_{\mathrm{c}} + 2T|k-1)||_2||\Theta^{-1} \\
&\quad \times [\tilde{p}_j(hT_{\mathrm{c}} + T|k) - \hat{p}_{i,w}(hT_{\mathrm{c}} + T|k)]||_2 \\
&\geq n_{0,ij}(hT_{\mathrm{c}} + 2T|k-1)^T\Theta^{-1} \\
&\quad \times [p_j(hT_{\mathrm{c}} + T|k) - \hat{p}_{i,w}(hT_{\mathrm{c}} + T|k)] \\
&= n_{0,ij}(hT_{\mathrm{c}} + 2T|k-1)^T\Theta^{-1} \\
&\quad \times [p_j(hT_{\mathrm{c}} + 2T|k-1) - \hat{p}_{i,w}(hT_{\mathrm{c}} + T|k)] \\
&\geq \hat{d}_{\min} \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\text{(C5)}
\end{aligned}
$$

□

Proof of Lemma 4. Since no new entries are added to the databases, there exists a trajectory $\tilde{p}_i(\cdot+T|k-1) \in \mathcal{D}_i(k)$ with $\tilde{p}_i(\cdot|k-1) = \tilde{p}_i(\cdot+T|k)$ (same for j), and thus collision freeness follows directly. □