IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Distributed NMPC for Cooperative Aerial Manipulation of Cable-Suspended Loads

Nicola De Carli[1*], Riccardo Belletti[1*], Emanuele Buzzurro[1],
Andrea Testa[2], Giuseppe Notarstefano[2], Marco Tognon[1]

***Abstract*—In this paper, we address the problem of cooperative manipulation of a cable-suspended load by a team of aerial robots. Unlike classical approaches that rely on centralized controllers, we propose a *Distributed Nonlinear Model Predictive Control* (DNMPC) framework in which the UAVs communicate over a peer-to-peer network a reduced amount of variables. In the proposed method, each robot handles only a small subset of the global optimization problem. The optimal motion computed by the DNMPC loop is then used as a reference for local nonlinear controllers that track the trajectory and compute the robot's actuation inputs. We validate the proposed scheme both through numerical simulations and real-world experiments on the *Fly-Crane* system: a rigid platform connected to three robots by pairs of cables.**

***Index Terms*—Aerial systems, multi-robot systems, cooperating robots**

I. INTRODUCTION

In recent years, there has been increasing interest in using multiple unmanned aerial vehicles (UAVs) for manipulation and transportation tasks [1]. Multi-UAV systems offer advantages over single-robot solutions, such as greater payload capacity and full six-degree-of-freedom (6DoF) control of large objects [2]. Applications include search-and-rescue [3], construction [4], and industrial transportation [5].

To enable UAV teams to manipulate and transport payloads, researchers have explored different attachment mechanisms, including spherical joints [6] and cables [5], [7], [8]. Cable-based systems are particularly attractive thanks to their lightweight nature and cost-effectiveness. Achieving full pose control requires generating a six-dimensional wrench through coordinated adjustments of cable forces and orientations. A minimal configuration consists of three UAVs connected to non-collinear points on the payload [5]. However, to compensate for external wrenches, such as wind disturbances, this setup requires repositioning the UAVs, leading to slow and imprecise responses.

Manuscript received: March 31, 2025; Revised July, 03, 2025; Accepted August, 04, 2025.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the project ANR-23-CE33-0013 "FlyHandyBot". Experiments presented in this paper were carried out thanks to a platform of the Robotex 2.0 French research infrastructure.

[1] Univ Rennes, CNRS, Inria, IRISA, F-35000 Rennes, France ndc@kth.se;marco.tognon@inria.fr

[2] Department of Electrical, Electronic and Information Engineering, Alma Mater Studiorum - Universita' di Bologna, Bologna, Italy, a.testa@unibo.it; giuseppe.notarstefano@unibo.it

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

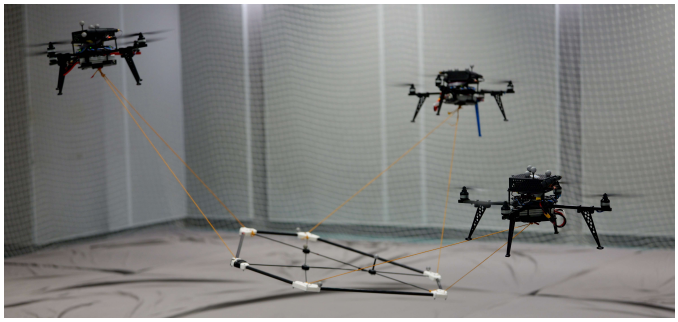

Fig. 1: The Fly-Crane system: A cooperative aerial transport system composed of three quadrotors connected to a payload via cables, enabling controlled manipulation and transportation.

A force-closed design [9] mitigates this issue by enabling rapid compensation of external forces through direct tension adjustments while maintaining cable orientation. This property requires at least six cables. The Fly-Crane system [10] (shown in Fig. 1 and Fig. 2) achieves this by equipping each UAV with two cables, ensuring force closure with a minimal number of robots. While the standard Fly-Crane setup employs three UAVs, the concept can scale to larger teams, increasing load capacity, disturbance rejection, and redundancy while preserving the benefits of force closure [10].

Classical approaches to cooperative transportation often rely on centralized control, where a single computation unit generates planning and control inputs for all robots. These methods require solving large-scale optimization problems and demand high computational and communication resources. As a result, they are typically executed offboard on high-performance computers and rely on high-bandwidth networks. To address these limitations, distributed optimization has emerged as an effective paradigm for multi-robot systems [11], [12]. In this paper, we propose a Distributed Nonlinear Model Predictive Control (DNMPC) framework for collaborative aerial transportation, where a team of UAVs, connected to a load via cables, coordinates in a fully distributed manner.

A. Related Work

Various control strategies have been proposed for cooperative aerial transportation. Centralized approaches include reactive methods [13], [14], which do not explicitly handle constraints, and predictive strategies such as NMPC [7], [15], which optimize trajectories for the full system but require significant computational and communication resources.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

To improve scalability, decentralized and distributed strategies have been explored. Some decentralized methods rely on formation control [16], [17] and leader-follower schemes [18], [19] to stabilize the load to a certain configuration rather than tracking a trajectory.

For the Fly-Crane system, previous works have primarily used reactive strategies, including robust H_∞ control [20] to deal with parameter uncertainties and an admittance framework combined with an inverse kinematic controller [21] to ensure safe task execution in presence of expected or unexpected interactions between the payload and the environment.

Distributed MPC has been investigated for aerial transportation, such as in [22], where quadrotors use rigid rods, though requiring all-to-all communication. In contrast, [23] presents a linear decentralized MPC for quasi-static motions of a bar, hence, not requiring control of the full 6-dimensional pose.

Beyond aerial robotics, distributed optimization has been explored in related domains. For instance, [24] investigates distributed transportation with maritime vessels using a multi-layer control structure. While some stages of the approach are distributed, a central coordinator is required to properly update the trajectory. In [25], the authors apply distributed optimization to trajectory planning for contact-based manipulation, requiring consensus on the full force vector applied by each robot. This consensus mechanism increases computational complexity as the number of robots grows. These studies demonstrate the potential of distributed optimization and MPC techniques in cooperative transportation while also highlighting limitations related to scalability and single points of failure of current approaches.

B. Contributions

This work proposes a Distributed Nonlinear Model Predictive Control (DNMPC) framework for tracking the full pose of a cable-suspended load using a team of UAVs. Each UAV receives reference trajectories—position, velocity, and acceleration—for its low-level controller. The optimal control problem is solved via a partition-based ADMM scheme [26]–[28], in which each UAV optimizes a local subproblem in a peer-to-peer network without a central coordinator. While such methods have been studied in theory and in generic contexts, we tailor the algorithm to the cooperative transportation problem by exploiting its specific sparsity structure: UAVs are coupled only through the shared load. This leads to an efficient and scalable implementation, where the optimization size per UAV remains independent of the team size—unlike in centralized MPC.

The partition-based distributed ADMM NMPC algorithm we use was introduced in [28], and a preliminary version of it had been used for contact-implicit cooperative trajectory optimization in [25]. However, [25] required each robot to optimize the entire vector of forces applied by all robots to the load, thus limiting its scalability. In contrast, our approach considers only the kinematic model of the load, requiring robots to exchange only the trajectory of the load with their neighbors. This significantly reduces communication and ensures the algorithm complexity remains independent of the number of robots.

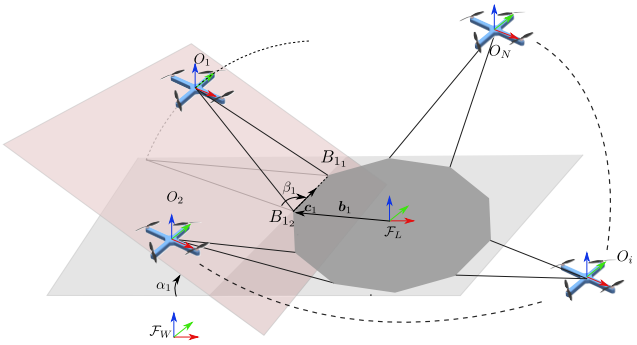

Fig. 2: Schematic representation of the Fly-Crane system with N robots and its key variables. The pink plane illustrates the plane formed by the two cables attached to robot 1, while the gray plane corresponds to the xy-plane of the load.

This work brings a largely theoretical distributed optimal control algorithm into practice, overcoming challenges posed by model uncertainties and hardware constraints. We validate the approach on the Fly-Crane platform through both realistic Software-in-the-Loop simulations and real-world experiments.

II. Preliminaries

A. Problem formulation

We consider a system composed of N UAVs attached to a rigid body (load) by cables. In particular, we refer to the Fly-Crane model (Fig. 2), which, as previously mentioned, exhibits the favourable robustness property of force closure, allowing for precise kinematic control. However, we underline that the same methodology could also be applied to classical configurations with one cable per robot.

In order to describe the system, let us define an inertial frame $\mathcal{F}_W = \{O_W, \boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W\}$, a body frame $\mathcal{F}_i = \{O_i, \boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z}_i\}$ attached to the center of mass of the i-th robot and a body frame $\mathcal{F}_L = \{O_L, \boldsymbol{x}_L, \boldsymbol{y}_L, \boldsymbol{z}_L\}$ attached to the center of mass of the load, where $O_\star$ and $\{\boldsymbol{x}_\star, \boldsymbol{y}_\star, \boldsymbol{z}_\star\}$ are the origin and unit axis of the frame $\mathcal{F}_\star$. The vector ${}^W\boldsymbol{p}_L \in \mathbb{R}^3$ describes the position of O_L with respect to $\mathcal{F}_W$ and the unit quaternion ${}^W\boldsymbol{q}_L = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^\top \in \mathbb{S}^3$ describes the orientation of $\mathcal{F}_L$ with respect to $\mathcal{F}_W$[1]. Quaternion multiplication will be represented by matrix multiplication notations, namely $\boldsymbol{q}_1 \odot \boldsymbol{q}_2 = \boldsymbol{Q}(\boldsymbol{q}_1)\boldsymbol{q}_2$, where $\odot$ indicates the quaternion multiplication, and $\boldsymbol{Q}(\cdot) : \mathbb{S}^3 \to \mathbb{R}^{4\times4}$. The corresponding rotation matrix, denoted as $\boldsymbol{R}_L \in SO(3)$, can be obtained as a function of $\boldsymbol{q}_L$. Moreover, we use the superscript d to indicate the desired value of a variable to be tracked, e.g., $\boldsymbol{p}_L^d$ denotes the desired load position.

Each UAV is tethered to the load by two cables that link the UAV's center of mass to an arbitrary point on the load. These connections are designed to ensure that there are no rotational constraints between the cable-platform and cable-UAV interfaces. As it is common, we assume that the cables have negligible mass and inertia with respect to the other bodies of the system, the deformations due to elasticity are negligible

[1]The left superscript indicates the reference frame. From now on, $\mathcal{F}_W$ is considered as reference frame when the superscript is omitted.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

in the operative conditions and the cables are always taut. As a consequence, the k-th cable is characterized by a constant length $l_k \in \mathbb{R}_{\geq 0}$. These assumptions are introduced to simplify the model and enable a tractable control design. Despite the mismatch with real cable behavior, the effectiveness of the resulting controller is validated through experiments.

We assume that the UAVs are able to communicate among them. In particular, we assume that the communication interaction among the robots is described by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, .., N\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the vertex set and the edge set, respectively. More in detail, $(i, j) \in \mathcal{E}$ if UAV i can communicate with UAV j. We denote as $\mathcal{N}_i$ the set of robots which communicate with robot i. Furthermore, we assume the communication graph $\mathcal{G}$ to be connected.

B. Modeling

Robot model: The position and velocity of the i-th robot are denoted as $\boldsymbol{p}_i \in \mathbb{R}^3$ and $\boldsymbol{v}_i \in \mathbb{R}^3$, respectively. A low-level controller is assumed to be in place, capable of tracking position trajectories while accounting for the robot's full dynamics, e.g., [29], [30]. Thus, we model the closed-loop translational dynamics of the robot as a double integrator:

$$\begin{aligned} \dot{\boldsymbol{p}}_i &= \boldsymbol{v}_i \\ \dot{\boldsymbol{v}}_i &= \boldsymbol{a}_i, \end{aligned} \tag{1}$$

in which the acceleration $\boldsymbol{a}_i \in \mathbb{R}^3$ is the input to the robot. This model provides a reasonable approximation for both fully-actuated vehicles and underactuated ones, assuming motions with small jerk and snap.

Load model: The load dynamics are modeled as a kinematic system in the form:

$$\begin{aligned} \dot{\boldsymbol{p}}_L &= \boldsymbol{v}_L, \\ \dot{\boldsymbol{q}}_L &= \frac{1}{2}\boldsymbol{Q}(\boldsymbol{q}_L) \begin{bmatrix} 0 \\ {}^L\boldsymbol{\omega}_L \end{bmatrix}, \end{aligned} \tag{2}$$

where $\boldsymbol{v}_L$ and ${}^L\boldsymbol{\omega}_L$ are, respectively, the linear and angular velocity of the load, with the latter being expressed in the load body frame.

Cable configuration: In the Fly-Crane configuration, each robot is subject to two geometric constraints, restricting its movement relative to the load to a circular trajectory parameterized by the angle $\alpha_i \in (-\pi, \pi]$ (see Fig. 2), whose kinematics are represented by a single integrator:

$$\dot{\alpha}_i = \omega_{\alpha_i}. \tag{3}$$

The resulting geometric constraint can be written as:

$$\boldsymbol{p}_i = \boldsymbol{p}_L + \boldsymbol{R}_L\, {}^L\boldsymbol{p}_i(\alpha_i), \tag{4}$$

where ${}^L\boldsymbol{p}_i(\alpha_i) = l_i \sin(\beta_i) \boldsymbol{R}_{{}^L\boldsymbol{c}_i}(\alpha_i) \frac{{}^L\boldsymbol{b}_i}{\|{}^L\boldsymbol{b}_i\|}$. Here, β_i is the fixed angle between the segment connecting the drone-cable attachment point O_i to the load-cable attachment point B_{i_2}, and the segment connecting the two load-cable attachment points B_{i_1} and B_{i_2} (see Fig. 2). Additionally, $\boldsymbol{c}_i$ represents the unit vector between the two load-cable attachment points, while $\boldsymbol{b}_i$ denotes the relative displacement between the load center of mass and B_{i_2}. For the system to be more robust to disturbances as well as to avoid inter-robot collisions, we want the cable angles to operate in a certain region

$$\underline{\alpha_i} \leq \alpha_i \leq \overline{\alpha_i}, \tag{5}$$

with $\underline{\alpha_i} > 0$. We express the kinematic relationship linking the robot velocities to the payload velocity twist and cable angular rates:

$$\boldsymbol{v} = \boldsymbol{J}(\boldsymbol{q}_L, \boldsymbol{\alpha}) \begin{bmatrix} \boldsymbol{v}_L \\ {}^L\boldsymbol{\omega}_L \\ \boldsymbol{\omega}_\alpha \end{bmatrix}, \tag{6}$$

where $\boldsymbol{v} = [\{\boldsymbol{v}_i\}_{i=1}^N]$ and $\boldsymbol{\omega}_\alpha = [\{\omega_{\alpha_i}\}_{i=1}^N]$. We use the notation $\left[\{\boldsymbol{v}_i\}_{i=1}^N\right] = \begin{bmatrix} \boldsymbol{v}_1^\top & \boldsymbol{v}_2^\top & \dots & \boldsymbol{v}_N^\top \end{bmatrix}^\top$ to denote the vertical stack of vectors indexed by the integer $i \in \{1, 2, .., N\}$. Moreover, we use $\boldsymbol{J}(\boldsymbol{q}_L, \boldsymbol{\alpha})$ to denote the geometric jacobian matrix, which, for $N \geq 3$, is full column rank (cf. [31]). As a result, given robot velocities that satisfy the system's geometric constraints, the load velocity twist and cable angular velocities are uniquely determined.

Objective: The control objective is to ensure that the load's full pose—both position and orientation—tracks the desired trajectory $(\boldsymbol{p}_L^d, \boldsymbol{q}_L^d, \boldsymbol{v}_L^d, \boldsymbol{\omega}_L^d)$ in a distributed manner.

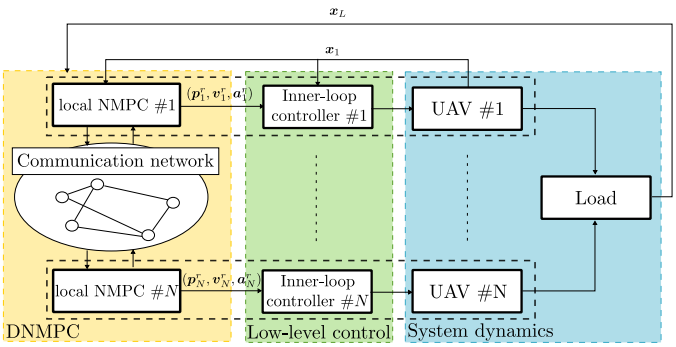

Fig. 3: Diagram of the proposed control framework.

III. Control Methodology

In this section, we describe the proposed control strategy, which includes a distributed NMPC as the outer-loop controller and a low-level controller which tracks the reference trajectory provided by the NMPC (see Fig. 3). We first describe the global optimal control problem in its centralized form and then explain how it can be reformulated in a distributed manner using *partition-based* ADMM.

A. OCP formulation

In the optimal control problem formulation, the control inputs include the robots' linear accelerations and the angular velocities of the cables, grouped as $\boldsymbol{u}_i = \begin{bmatrix} \boldsymbol{a}_i^\top & \omega_{\alpha_i} \end{bmatrix}^\top$, as well as the load's linear and angular velocities, denoted by $\boldsymbol{u}_L = \begin{bmatrix} \boldsymbol{v}_L^\top & {}^L\boldsymbol{\omega}_L{}^\top \end{bmatrix}^\top$. We define the vector containing all system inputs as

$$\boldsymbol{u} = \begin{bmatrix} \{\boldsymbol{u}_i\}_{i=1}^N \\ \boldsymbol{u}_L \end{bmatrix}. \tag{7}$$

The state is given by the robots position, velocity and corresponding cable state $\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{p}_i^\top & \boldsymbol{v}_i^\top & \alpha_i \end{bmatrix}^\top$ as well as the load pose $\boldsymbol{x}_L = \begin{bmatrix} \boldsymbol{p}_L^\top & \boldsymbol{q}_L^\top \end{bmatrix}^\top$.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

We point out that the platform velocity twist $\boldsymbol{u}_L$ and cable velocities ω_{α_i} are ficticious inputs. They are not actually a controlled input, but an algebraic function of the robots velocities, due to the geometric constraint (4). Consistency among the resulting velocities is ensured by enforcing (4) inside the following nonlinear optimal control problem (OCP):

$$\begin{aligned}
\min_{\boldsymbol{u}(\cdot)} \quad & \int_t^{t+T} \Big[N\ell_L(\boldsymbol{x}_L(\tau), \boldsymbol{u}_L(\tau)) \\
& + \sum_{i=1}^{N} \ell_i(\boldsymbol{x}_i(\tau), \boldsymbol{u}_i(\tau)) \Big] d\tau \\
& + N\phi_L(\boldsymbol{x}_L(t+T)) + \sum_{i=1}^{N} \phi_i(\boldsymbol{x}_i(t+T)) \\
\text{s.t.} \quad & \text{Initial conditions:} \\
& \boldsymbol{x}_L(t) = \hat{\boldsymbol{x}}_L(t) \quad \boldsymbol{x}_i(t) = \hat{\boldsymbol{x}}_i(t) \\
& \text{System dynamics:} \quad (1)-(3) \\
& \text{Load dynamics:} \quad (2) \\
& \text{Geometric constraint:} \quad (4) \\
& \text{Inequality constraints:} \quad (5) \\
& \boldsymbol{p}_i(\tau) \notin \mathcal{O}(\tau) \quad \boldsymbol{v}_i(\tau) \in \mathcal{V}_i, \quad \boldsymbol{a}_i(\tau) \in \mathcal{A}_i \\
& \forall \tau \in [t, t+T] \quad i = 1, \dots, N
\end{aligned} \tag{8}$$

where T is the prediction horizon, the factor N in the cost is added in view of later developments, $\hat{\boldsymbol{x}}_L$ and $\hat{\boldsymbol{x}}_i$ are the load and robots state feedback at time t, $\mathcal{O}(t)$ represents the space occupied by obstacles at time t, $\mathcal{V}_i$ and $\mathcal{A}_i$ represent the convex sets in which velocity and accelerations must lie due to saturations, respectively. The terms $\ell_L(\cdot)$ and $\ell_i(\cdot)$ are stage costs, accounting for load tracking error and local variables specific to each robot, respectively.

$$\begin{aligned}
\ell_L(\boldsymbol{x}_L, \boldsymbol{u}_L) &= \left\|\boldsymbol{p}_L^d - \boldsymbol{p}_L\right\|_{\boldsymbol{P}_p}^2 + \|\boldsymbol{e}_{\boldsymbol{q}_L}\|_{\boldsymbol{P}_q}^2 + \left\|\boldsymbol{v}_L^d - \boldsymbol{v}_L\right\|_{\boldsymbol{P}_v}^2 \\
&\quad + \left\|{}^L\boldsymbol{\omega}_L{}^d - {}^L\boldsymbol{\omega}_L\right\|_{\boldsymbol{P}_\omega}^2 \\
\ell_i(\boldsymbol{x}_i, \boldsymbol{u}_i) &= \left\|\alpha_i^d - \alpha_i\right\|_{\boldsymbol{P}_{\alpha_i}}^2 + \left\|\omega_{\alpha_i}^d - \omega_{\alpha_i}\right\|_{\boldsymbol{P}_{\omega_{\alpha_i}}}^2 \\
&\quad + \left\|\boldsymbol{u}_i^d - \boldsymbol{u}_i\right\|_{\boldsymbol{P}_{u_i}}^2,
\end{aligned} \tag{9}$$

with $\boldsymbol{P}_p$, $\boldsymbol{P}_q$, $\boldsymbol{P}_v$, $\boldsymbol{P}_\omega$, $\boldsymbol{P}_{\alpha_i}$, $\boldsymbol{P}_{\omega_{\alpha_i}}$ and $\boldsymbol{P}_{u_i}$ being positive definite weight matrices of appropriate dimensions, and $\boldsymbol{e}_{\boldsymbol{q}_L}$ being the quaternion error, while $\phi_L(\cdot)$ and $\phi_i(\cdot)$ are the respective terminal costs:

$$\begin{aligned}
\phi_L(\boldsymbol{x}_L, \boldsymbol{u}_L) &= \left\|\boldsymbol{p}_L^d - \boldsymbol{p}_L\right\|_{\boldsymbol{P}_{pT}}^2 + \|\boldsymbol{e}_{\boldsymbol{q}_L}\|_{\boldsymbol{P}_{qT}}^2 \\
\phi_i(\boldsymbol{x}_i, \boldsymbol{u}_i) &= \left\|\alpha_i^d - \alpha_i\right\|_{\boldsymbol{P}_{\alpha_i T}}^2.
\end{aligned} \tag{10}$$

The OCP (8) is transcribed into a nonlinear program (NLP) using a multiple-shooting approach, where the input trajectory is parameterized in a finite-dimensional space. We define a sequence of control times t_h separated by intervals of duration $\Delta t = T/H$, where H denotes the number of steps in the prediction horizon. The input is parameterized as piecewise constant: $\boldsymbol{u}(\tau) = \boldsymbol{u}_h(t)$, $\tau \in [t + t_h, t + t_{h+1})$. We then define the discretized load input trajectory which starts at time t as $\bar{\boldsymbol{u}}_L = \left[\boldsymbol{u}_{L,0}^\top, ..., \boldsymbol{u}_{L,(H-1)}^\top\right]^\top$, and, similarly, the load state trajectory as $\bar{\boldsymbol{x}}_L$, the i-th robot input and state trajectories, respectively, as $\bar{\boldsymbol{u}}_i$ and $\bar{\boldsymbol{x}}_i$. Problem (8) contains terms that couple the trajectories of all the robots through the load and the geometric constraints, so that its centralized solution requires an update that uses all the agent trajectories. In the next section, we show how to distribute the problem so that each robot needs to compute only its own state and input trajectories, along with those of the load. Consistency among the load trajectories predicted by different robots is then enforced through a consensus-like mechanism.

B. Distributed OCP formulation

Problem (8) is inherently centralized and not directly separable, as all robots must agree on the optimal load trajectory $(\bar{\boldsymbol{x}}_L, \bar{\boldsymbol{u}}_L)$. To enable a distributed formulation, we follow a partition-based implementation of the ADMM algorithm, as proposed in [26]–[28], adapting it to the specific structure of cooperative transportation. In particular, we exploit the sparsity induced by the fact that agents are only coupled through the shared load, allowing an efficient and scalable distributed formulation.

ADMM follows a decomposition-coordination approach, where a large global optimization problem is split into smaller subproblems that are solved locally, while a coordination mechanism ensures consistency across all solutions [32]. Specifically, the distributed ADMM reformulates the centralized problem by introducing auxiliary variables, enabling each robot to maintain a local copy of the shared optimization variables–in this case, the load trajectory. At each iteration, the algorithm consists of two main steps:

- *Local optimization* (decomposition step): Each robot i solves a small-scale local primal optimization problem.
- *Consensus update* (coordination step): A dual variable update is performed based on local solutions exchanged among neighboring robots. This step enforces agreement between neighboring copies of the shared variables, ensuring consistency across all robots.

In our formulation, each robot maintains a local copy of the load state ($\bar{\boldsymbol{x}}_L$) and input ($\bar{\boldsymbol{u}}_L$) trajectories, denoted as $\hat{\boldsymbol{x}}_{Li} = \begin{bmatrix}\boldsymbol{x}_{Li,0} & ... & \boldsymbol{x}_{Li,H-1}\end{bmatrix}$ and $\hat{\boldsymbol{u}}_{Li} = \begin{bmatrix}\boldsymbol{u}_{Li,0} & ... & \boldsymbol{u}_{Li,H-1}\end{bmatrix}$, respectively. A key feature of the ADMM-based approach is that these local copies are updated iteratively through communication with neighboring robots. If the algorithm converges, all robots reach agreement on their respective copies, ensuring consistency in the load trajectory. To enforce this agreement, we introduce a coupling constraint for each pair $(i, j) \in \mathcal{E}$, which enforces consistency among the local copies of the load trajectory variables

$$\boldsymbol{u}_{Li,h} = \boldsymbol{u}_{Lj,h}, \ \forall (i,j) \in \mathcal{E}, \text{ for } h \in \{0, 1, ..., H-1\}. \tag{11}$$

To be able to split the problem in local subproblems, we introduce the auxiliary variables $\bar{\boldsymbol{c}}_{ij} = \begin{bmatrix}\boldsymbol{c}_{ij,0} & ... & \boldsymbol{c}_{ij,H-1}\end{bmatrix}$, with $\boldsymbol{c}_{ij,h} \in \mathbb{R}^6$ and we rewrite (11) as

$$\boldsymbol{u}_{Li,h} = \boldsymbol{c}_{ij,h} \qquad \boldsymbol{c}_{ij,h} = \boldsymbol{c}_{ji,h} \qquad \boldsymbol{u}_{Lj,h} = \boldsymbol{c}_{ji,h}. \tag{12}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The problem (8) , after transcription to an NLP and introduction of auxiliary variables and associated constraint (12) becomes

$$\begin{aligned}
\min_{\substack{\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{u}}_i,\\ \hat{\boldsymbol{x}}_{Li},\bar{\boldsymbol{x}}_i\\ \boldsymbol{c}_{ij,h},\boldsymbol{c}_{ji,h}\forall j\in\mathcal{N}_i\\ \forall i\in\{1,\dots,N\}}} \quad & \sum_{i=1}^{N} J_i(\hat{\boldsymbol{x}}_{Li},\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i)\\
\text{s.t.} \quad & \boldsymbol{x}_{i,(h+1)} = \boldsymbol{f}_i(\boldsymbol{x}_{i,h},\boldsymbol{u}_{i,h})\\
& \boldsymbol{x}_{Li,(h+1)} = \boldsymbol{f}_L(\boldsymbol{x}_{Li,h},\boldsymbol{u}_{Li,h})\\
& \boldsymbol{p}_{i,h} = \boldsymbol{p}_{Li,h} + \boldsymbol{R}(\boldsymbol{q}_{Li,h})\,{}^L\boldsymbol{p}_i(\alpha_{i,h})\\
& \underline{\alpha_i} \le \alpha_{i,h} \le \overline{\alpha_i}\\
& \boldsymbol{p}_{i,h} \notin \mathcal{O}(h), \boldsymbol{v}_{i,h} \in \mathcal{V}_i, \boldsymbol{a}_{i,h} \in \mathcal{A}_i\\
& \boldsymbol{u}_{Li,h} = \boldsymbol{c}_{ij,h} = \boldsymbol{c}_{ji,h} \quad \forall j \in \mathcal{N}_i\\
& \textbf{for } h = 0,\dots,H-1,\\
& \forall i \in \{1,\dots,N\}
\end{aligned} \tag{13}$$

where

$$\begin{aligned}
&J_i(\hat{\boldsymbol{x}}_{Li},\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i) =\\
&\sum_{h=0}^{H-1}\left[\ell_{L,h}(\hat{\boldsymbol{x}}_{Li,h},\hat{\boldsymbol{u}}_{Li,h}) + \ell_{i,h}(\bar{\boldsymbol{x}}_{i,h},\bar{\boldsymbol{u}}_{i,h})\right],
\end{aligned} \tag{14}$$

with $\ell_{L,h}$ and $\ell_{i,h}$ obtained by discretizing (9) and, similarly for $\boldsymbol{f}_i$ and $\boldsymbol{f}_L$, which are obtained by discretizing (1) and (2), respectively.

The problem in (13) has a structure amenable to be split in local subproblems and be solved using distributed ADMM. In the following, first, we introduce the augmented Lagrangian relaxation of problem (13) corresponding to the constraint (12). From this, the corresponding steps of the ADMM iterations are derived: in the first step, the primal problem [32] is solved; in the second step, after communication, the dual variables are updated. The augmented Lagrangian takes the following form:

$$\begin{aligned}
\mathcal{L} = &\sum_{i=1}^{N}\Big[J_i(\hat{\boldsymbol{x}}_{Li},\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i)\\
&+\sum_{j\in\mathcal{N}_i}\Big(\boldsymbol{\lambda}_{ij}^\top(\hat{\boldsymbol{u}}_{Li}-\bar{\boldsymbol{c}}_{ij}) + \boldsymbol{\mu}_{ij}^\top(\hat{\boldsymbol{u}}_{Li}-\bar{\boldsymbol{c}}_{ji})\\
&+\frac{\rho}{2}\left\|\hat{\boldsymbol{u}}_{Li}-\bar{\boldsymbol{c}}_{ij}\right\|^2 + \frac{\rho}{2}\left\|\hat{\boldsymbol{u}}_{Li}-\bar{\boldsymbol{c}}_{ji}\right\|^2\Big)\Big]\\
&= \sum_{i=1}^{N}\mathcal{L}_i(\hat{\boldsymbol{x}}_{Li},\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i,\{\bar{\boldsymbol{c}}_{ij}\}_{j\in\mathcal{N}_i},\{\bar{\boldsymbol{c}}_{ji}\}_{j\in\mathcal{N}_i},\\
&\{\boldsymbol{\lambda}_{ij}\}_{j\in\mathcal{N}_i},\{\boldsymbol{\mu}_{ij}\}_{j\in\mathcal{N}_i}),
\end{aligned} \tag{15}$$

where $\boldsymbol{\lambda}_{ij}$ and $\boldsymbol{\mu}_{ij}$ are Lagrange multipliers, $\rho > 0$ is a penalty parameter and $\mathcal{L}_i(\cdot)$ has been defined. The standard ADMM would now proceed with a step in which $\mathcal{L}_i(\cdot)$ is minimized with respect to the primal variables $(\hat{\boldsymbol{x}}_{Li},\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i)$ subject to local constraints, followed by a step in which $\mathcal{L}_i(\cdot)$ is minimized with respect to the other primal variables $\bar{\boldsymbol{c}}_{ij}$, concluding with a coordination step which updates the dual variables $\boldsymbol{\lambda}_{ij}$ and $\boldsymbol{\mu}_{ij}$. However, it can be showed after lenghty calculations (see [28] for the details) that, after a change of coordinates of the dual variables $\boldsymbol{q}_i = \sum_{j\in\mathcal{N}_i}(\boldsymbol{\lambda}_{ij}+\boldsymbol{\mu}_{ij})$, the algorithm can be simplified to the following two steps: 1) each robot solves a local constrained optimization problem

$$\begin{aligned}
\begin{bmatrix}\bar{\boldsymbol{u}}_i^{k+1}\\ \hat{\boldsymbol{u}}_{Li}^{k+1}\\ \hat{\boldsymbol{x}}_{Li}^{k+1}\\ \bar{\boldsymbol{x}}_i^{k+1}\end{bmatrix} = \underset{\substack{\boldsymbol{u}_i,\hat{\boldsymbol{u}}_{Li}\\ \hat{\boldsymbol{x}}_{Li},\bar{\boldsymbol{x}}_i}}{\operatorname{argmin}}\Big\{ & J_i(\hat{\boldsymbol{x}}_{Li},\hat{\boldsymbol{u}}_{Li},\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i) + \boldsymbol{q}_i^{k\top}\hat{\boldsymbol{u}}_{Li}\\
&+\rho\sum_{j\in\mathcal{N}_i}\left\|\hat{\boldsymbol{u}}_{Li}-\frac{1}{2}\left(\hat{\boldsymbol{u}}_{Li}^k+\hat{\boldsymbol{u}}_{Lj}^k\right)\right\|^2\Big\}\\
&\text{subject to local constraints from (13),}
\end{aligned} \tag{16}$$

where the exponent k represents the iteration count of the ADMM algorithm; 2) the local variable $\hat{\boldsymbol{u}}_{Li}^{k+1}$ is communicated to neighboring robots (and the one of the neighbors is received) and the following dual update step is performed:

$$\boldsymbol{q}_i^{k+1} = \boldsymbol{q}_i^k + \rho\sum_{j\in\mathcal{N}_i}\left(\hat{\boldsymbol{u}}_{Li}^{k+1}-\hat{\boldsymbol{u}}_{Lj}^{k+1}\right), \tag{17}$$

where $\boldsymbol{q}_i$ can be interpreted as the integral of the consensus error. Notice that the load input trajectory $\hat{\boldsymbol{u}}_{Li}^{k+1}$ is the only information which needs to be communicated to neighboring robots. We remark that the consensus can be arbitrarily reached on velocities or pose variables, or even both (at the price of higher communication load). However, performing the consensus on the orientation trajectory creates an additional complexity related to the non-Euclidean nature of the manifold, which can be avoided by performing the consensus on the angular velocity instead.

C. Robots inner-loop

The DNMPC outputs the current optimization variables $(\bar{\boldsymbol{x}}_i,\bar{\boldsymbol{u}}_i)$ for each robot, computed via (16). From these, each robot extracts its reference trajectory—position $\boldsymbol{p}_i^r$, velocity $\boldsymbol{v}_i^r$, and acceleration $\boldsymbol{a}_i^r$ (see Fig. 3)—which are tracked by onboard low-level controllers handling full system dynamics.

Various low-level controllers can be used, such as the geometric controller from [33] (used here) or the incremental nonlinear dynamic inversion method in [34]. These map reference trajectories to motor commands, ensuring that the robots closely track the desired trajectory while respecting the physical constraints of the system, such as thrust limitations and actuator dynamics.

This hierarchical architecture decouples the high-level NMPC—based on a simplified kinematic model—from the full robot dynamics, improving computational efficiency. Although the NMPC cannot guarantee full dynamic feasibility, this is mitigated by imposing velocity and acceleration constraints consistent with the robots' physical limits, ensuring the trajectories remain trackable.

IV. Results

This section presents results from both realistic simulations in Gazebo and real-world experiments. We first evaluate the proposed algorithm in simulation, comparing it to a centralized kinematic NMPC and showcasing its obstacle avoidance capabilities. We then report real-world experimental results for different reference trajectories.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The controller uses the following parameters: the penalty coefficients for the consensus constraint violations are set to $\rho_p = 20$ for the position and $\rho_\omega = 10$ for the angular velocity. The weight matrices are diagonal, with the following values on the diagonal: $\boldsymbol{P}_p = \mathrm{diag}(200, 200, 200)$, $\boldsymbol{P}_q = \mathrm{diag}(200, 200, 200)$, $\boldsymbol{P}_v = \mathrm{diag}(8, 8, 8)$, $\boldsymbol{P}_\omega = \mathrm{diag}(0.1, 0.1, 0.1)$, $\boldsymbol{P}_i = \mathrm{diag}(1, 1, 1)$, $\boldsymbol{P}_{\alpha_i} = \mathrm{diag}(0.01, 0.01, 0.01)$, $\boldsymbol{P}_{\omega_{\alpha_i}} = \mathrm{diag}(0.01, 0.01, 0.01)$, and $\boldsymbol{P}_{u_i} = \mathrm{diag}(1.0, 1.0, 1.0)$. The maximum acceleration norm is set to $a_{\max} = 6\,\mathrm{m\,s^{-2}}$, while $\underline{\alpha_i} = \pi/4$ and $\bar{\alpha_i} = 4/6\pi$.

A. Experimental and simulation setup

To validate and demonstrate the performance of the proposed controller, simulation and experimental tests were also performed with the Fly-Crane system using three quadrotors in an indoor environment. The simulations were designed to closely replicate the experimental setup through a high-fidelity Software-in-the-Loop (SITL) configuration in Gazebo. In particular, the simulation incorporates the same parameters and software stack used on the real robots—including the state estimator, control architecture, and even the firmware running on the Electronic Speed Controllers (ESCs). Cable dynamics are modeled using a series of piecewise mass-spring-damper elements, ensuring realistic behavior of the suspended load. This careful design minimizes the gap between simulation and reality, providing a reliable testbed for validating the proposed controller.

Each quadrotor weighs $1.3\,\mathrm{kg}$, the load weighs $0.338\,\mathrm{kg}$, and the length of each cable is $1.1\,\mathrm{m}$. The aerial vehicles are equipped with standard flight controllers, four brushless motor controllers to regulate propeller speed in a closed-loop configuration, and an onboard PC running a state estimator and the geometric controller [33]. State estimation is performed onboard at $1\,\mathrm{kHz}$ using an Unscented Kalman Filter (UKF), which fuses Motion Capture (MoCap) system measurements ($120\,\mathrm{Hz}$) with IMU data ($1\,\mathrm{kHz}$). Each cable angle α_i and the corresponding angular velocity ω_{α_i} are determined based on the robot's position and velocity, along with the load's pose, velocity twist, and attachment point positions. All simulations and experiments are conducted using the open-source framework TeleKyb3, which provides real-time control and state estimation capabilities[2].

During the initial phase of each experiment, the load is lifted off the ground, and the system is brought to the initial desired configuration. In this phase, the aerial vehicles are independently controlled using a position controller. Once stabilized, the proposed distributed controller is activated.

The proposed distributed MPC controller is implemented in MATLAB using ACADOS [35], employing a real-time iteration scheme to solve the local primal step (16) of the NMPC. Although the algorithm is designed for distributed execution, our current implementation runs one instance of the distributed controller per agent on a single desktop PC. Each instance sends the desired position, velocity, and acceleration to the corresponding quadrotor's low-level controller at $30\,\mathrm{Hz}$ via a Wi-Fi connection. This setup, commonly adopted in related literature, enables a fast validation of the proposed distributed scheme. In the case of the Fly-Crane, involving three robots, we consider a fully connected communication graph. Although our tests are conducted on this simpler setup, they still serve as a valid proof of concept and demonstrate the effectiveness of our approach in real experiments. Extending the implementation to a sparser communication network across multiple drones would require extensive engineering efforts, which we leave as future work.

[2]TeleKyb3 is available at https://git.openrobots.org/projects/telekyb3.

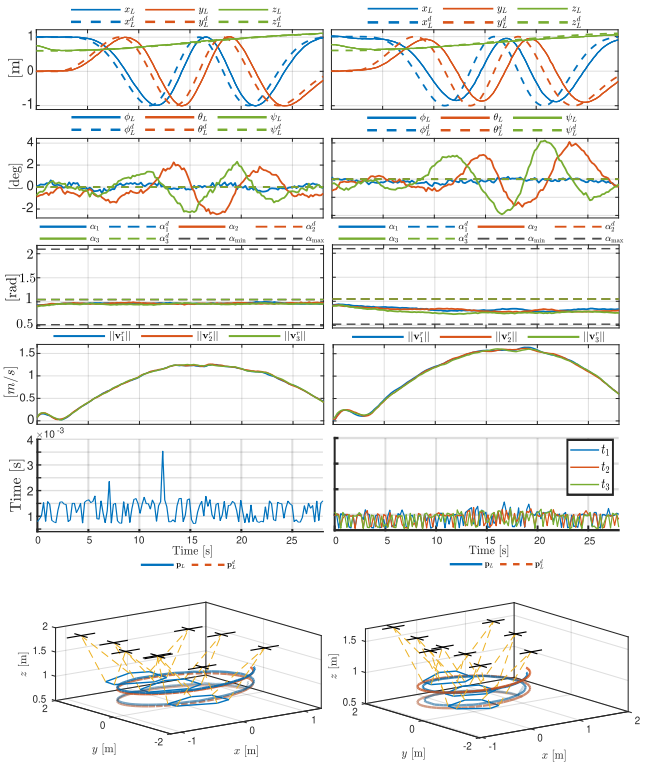

Fig. 4: Comparison performance between results obtained using centralized (on the left column) and distributed (on the right column) NMPC.

B. Numerical simulations

1) Comparison with the centralized: We compare the proposed distributed controller with a centralized counterpart solving (8). The reference trajectory involves a spiral motion along the vertical axis, with load velocities reaching up to $1.5\,\mathrm{m\,s^{-1}}$. As shown in Fig. 4, the centralized controller yields slightly better tracking performance, as expected, but at the cost of higher computation per iteration. Future work will explore how this trade-off evolves with larger robot teams and sparser communication graphs.

2) Obstacle avoidance: We consider a time-polynomial reference trajectory along a straight line, moving the load from approximately [0.0, 0.0, 0.675]m to [0, 5, 1]m, while avoiding a spherical obstacle centered at [0.6, 3.0, 1.0]m with radius $0.5\,\mathrm{m}$. A minimum safety distance of $1\,\mathrm{m}$ from the obstacle center is enforced, causing robot 3 (Fig. 5) to adjust its path accordingly.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

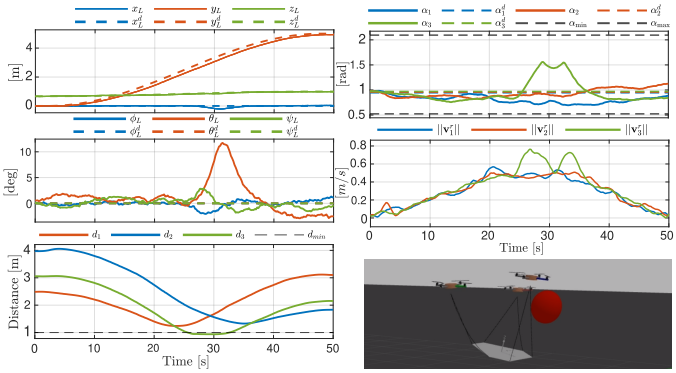

Fig. 5: Results from a simulation where obstacle avoidance intervenes to prevent robot 3 from colliding.

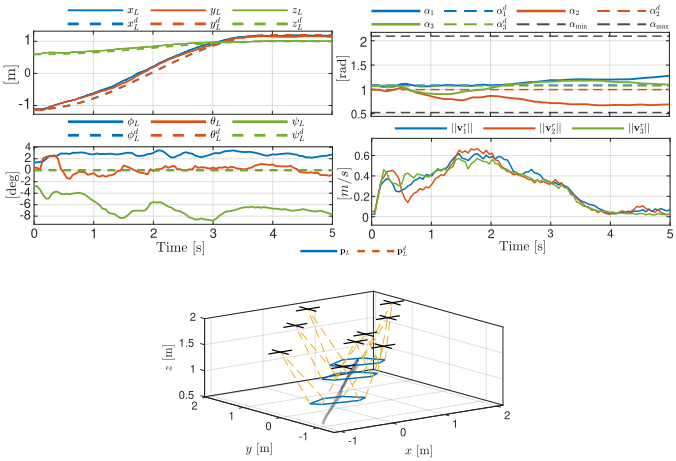

Fig. 6: Results from an experiment in which the desired trajectory is a straight line parameterized by a polynomial function.

Because the desired cable angles α_i are weighted less in the cost function, the controller prioritizes load trajectory tracking. As a result, α_3 deviates from its nominal value (see Fig. 5, upper and lower right). Still, tracking performance remains high, with only a slight pitch deviation (just over $10\,\mathrm{deg}$) from the reference, while satisfying all safety constraints (lower left plot).

C. Experimental Results

We validate the proposed controller on three reference trajectories: (i) a straight line defined by a time-polynomial (Fig.6), (ii) a trajectory that primarily induces a large rotation of the load, highlighting the controller's ability to track the full pose of the payload (Fig.7), and (iii) a fast circular path centered at $[0, 0, 0.6]\mathrm{m}$ with a $1.3\,\mathrm{m}$ radius (Fig.8,9). In the straight-line and circular cases, the desired orientation is fixed at $[0, 0, 0]$ in roll, pitch, and yaw to emphasize position tracking. The aerial robots reach speeds up to $1.5\,\mathrm{m\,s^{-1}}$ during circular motion, demonstrating the controller's ability to handle dynamic trajectories. Performance is evaluated in terms of tracking accuracy for both load position and orientation. We also report cable angles α_i and robots velocity norms as estimated onboard. For the straight and circular trajectories, we include 3D tracking visualizations and snapshots of the system along the path, where transparency denotes time progression.

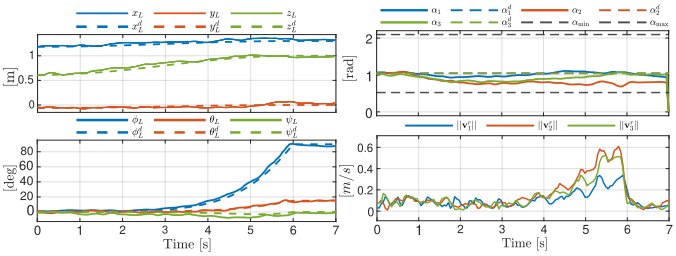

Fig. 7: Results from an experiment in which the desired trajectory require a large yaw rotation and smaller roll and pitch rotations.

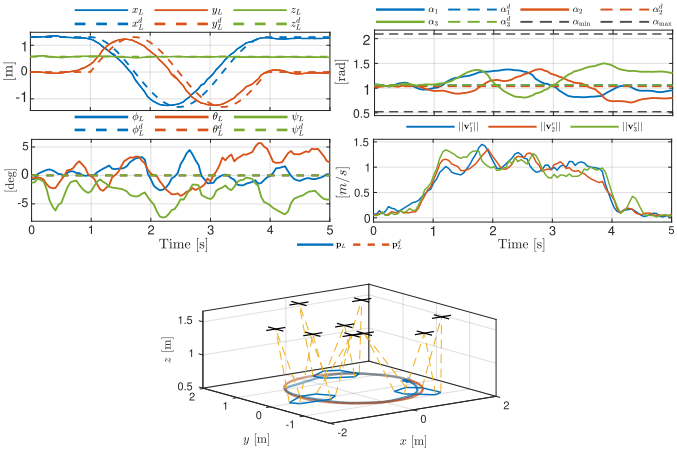

Fig. 8: Results from an experiment in which the desired trajectory is a high speed circumference.

Position tracking errors remain within a few centimeters, mostly due to small geometric mismatches and the underactuated nature of the drones, which require tilting that affects load position. Cable-induced disturbances also contribute to minor deviations. Orientation tracking stays within $8\,\mathrm{deg}$ of the reference. While the tracking of the desired cable angles is assigned a low weight in the cost function—resulting in only approximate convergence to the reference setpoints—the constraints on the minimum and maximum allowable angles are always satisfied. This ensures that safe configurations are maintained throughout the motion, and potential collisions among drones are avoided.

V. CONCLUSIONS

In this work, we proposed a Distributed Nonlinear Model Predictive Control (DNMPC) framework for cooperative transportation with underactuated UAVs tethered to a shared payload. The method employs a partition-based ADMM algorithm

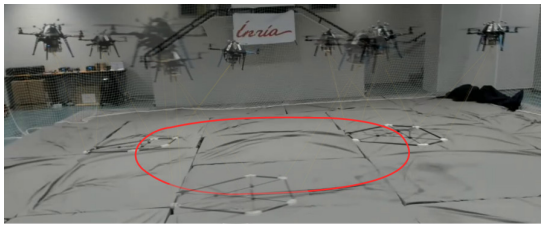

Fig. 9: View of the trajectory performed by the Fly-Crane system.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

to distribute computation across a peer-to-peer network, reducing communication overhead and enabling scalability. Unlike centralized formulations, each UAV solves a local subproblem, and the per-agent computational complexity remains constant regardless of team size.

The framework was validated through real-world experiments on the Fly-Crane system, demonstrating its effectiveness in achieving trajectory tracking for the full pose of the payload. Results highlighted the algorithm's robustness to system constraints and its ability to adapt to dynamic environments while maintaining high precision and satisfying physical limits.

Future work will focus on exploring the viability of extending this approach to incorporate the dynamic model of the system in the DNMPC while keeping the computation and communication complexity independent on the number of robots, exploring more efficient real-time implementations, and investigating its application in fully distributed experimental setups.

REFERENCES

[1] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2021.

[2] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[3] M. Bernard, K. Kondak, I. Maza, and A. Ollero, "Autonomous transportation and deployment with aerial robots for search and rescue missions," *Journal of Field Robotics*, vol. 28, no. 6, pp. 914–931, 2011.

[4] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014.

[5] K. Sreenath and V. R. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," in *Robotics: Science and Systems*, 2013.

[6] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto, "Robust collaborative object transportation using multiple mavs," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1020–1044, 2019.

[7] G. Li and G. Loianno, "Nonlinear model predictive control for cooperative transportation and manipulation of cable suspended payloads with multiple quadrotors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023, pp. 5034–5041.

[8] Y. Wang, G. Yu, W. Xie, W. Zhang, and C. Silvestre, "Robust cooperative transportation of a cable-suspended payload by multiple quadrotors featuring cable-reconfiguration capabilities," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[9] K. Lynch, *Modern Robotics*. Cambridge University Press, 2017.

[10] M. Manubens, D. Devaurs, L. Ros, and J. Cortés, "Motion planning for 6-d manipulation with aerial towed-cable systems," in *Robotics: science and systems*, 2013.

[11] A. Testa, G. Carnevale, and G. Notarstefano, "A tutorial on distributed optimization for cooperative robotics: From setups and algorithms to toolboxes and research directions," *Proceedings of the IEEE*, vol. 113, no. 1, pp. 40–65, 2025.

[12] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," *arXiv preprint arXiv:2103.12840*, 2021.

[13] H.-N. Nguyen, S. Park, J. Park, and D. Lee, "A novel robotic platform for aerial manipulation using quadrotors as rotating thrust generators," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 353–369, 2018.

[14] T. Lee, "Geometric control of quadrotor uavs transporting a cable-suspended rigid body," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 255–264, 2017.

[15] S. Sun, X. Wang, D. Sanalitro, A. Franchi, M. Tognon, and J. Alonso-Mora, "Agile and cooperative aerial manipulation of a cable-suspended load," *arXiv preprint arXiv:2501.18802*, 2025.

[16] K. Klausen, C. Meissen, T. I. Fossen, M. Arcak, and T. A. Johansen, "Cooperative control for multirotors transporting an unknown suspended load under environmental disturbances," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 2, pp. 653–660, 2018.

[17] H. G. De Marina and E. Smeur, "Flexible collaborative transportation by a team of rotorcraft," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 1074–1080.

[18] A. Tagliabue, M. Kamel, S. Verling, R. Siegwart, and J. Nieto, "Collaborative transportation using mavs via passive force control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5766–5773.

[19] C. Gabellieri, M. Tognon, D. Sanalitro, L. Pallottino, and A. Franchi, "A study on force-based collaboration in swarms," *Swarm Intelligence*, vol. 14, no. 1, pp. 57–82, 2020.

[20] D. Sanalitro, H. J. Savino, M. Tognon, J. Cortés, and A. Franchi, "Full-pose manipulation control of a cable-suspended load with multiple uavs under uncertainties," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2185–2191, 2020.

[21] D. Sanalitro, M. Tognon, A. J. Cano, J. Cortés, and A. Franchi, "Indirect force control of a cable-suspended aerial multi-robot manipulator," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6726–6733, 2022.

[22] J. Wehbeh, S. Rahman, and I. Sharf, "Distributed model predictive control for uavs collaborative payload transport," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 11 666–11 672.

[23] R. C. Sundin, P. Roque, and D. V. Dimarogonas, "Decentralized model predictive control for equilibrium-based collaborative uav bar transportation," in *IEEE International Conference on Robotics and Automation*, 2022, pp. 4915–4921.

[24] L. Chen, H. Hopman, and R. R. Negenborn, "Distributed model predictive control for cooperative floating object transport with multi-vessel systems," *Ocean Engineering*, vol. 191, p. 106515, 2019.

[25] O. Shorinwa and M. Schwager, "Distributed contact-implicit trajectory optimization for collaborative manipulation," in *IEEE International Symposium on Multi-Robot and Multi-Agent Systems*, 2021, pp. 56–65.

[26] T. Erseghe, "A distributed and scalable processing method based upon admm," *IEEE Signal Processing Letters*, vol. 19, no. 9, pp. 563–566, 2012.

[27] N. Bastianello, R. Carli, L. Schenato, and M. Todescato, "A partition-based implementation of the relaxed admm for distributed convex optimization over lossy networks," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3379–3384.

[28] O. Shorinwa and M. Schwager, "Distributed model predictive control via separable optimization in multiagent networks," *IEEE Transactions on Automatic Control*, vol. 69, no. 1, pp. 230–245, 2023.

[29] M. Hamandi, M. Tognon, and A. Franchi, "Direct acceleration feedback control of quadrotor aerial vehicles," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 5335–5341.

[30] L. Balandi, P. R. Giordano, and M. Tognon, "Qp-based inner-loop control for constraint-safe and robust trajectory tracking for aerial robots," *IEEE Robotics and Automation Letters*, 2025.

[31] D. Six, S. Briot, A. Chriette, and P. Martinet, "The kinematics, dynamics and control of a flying parallel robot with three quadrotors," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 559–566, 2017.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

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

[34] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, 2020.

[35] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.