Distributed Geometric and Optimization-based Control of Multiple Quadrotors for Cable-Suspended Payload Transport

Khaled Wahba and Wolfgang Hönig

***Abstract*— We consider transporting a heavy payload that is attached to multiple quadrotors. The current state-of-the-art distributed geometric controller has several limitations: it does not avoid collisions between the robots or obstacles and there are almost no successful physical flights. We generalize this controller by introducing an efficient quadratic program for quadrotor load balancing, which can consider additional constraints, such as collision avoidance or weight limits. Since we optimize in the null-space of the prior work, we retain the existing strong stability results. Moreover, we provide an implementation that runs in real-time on an embedded microcontroller at a high rate. Our code is also usable in our software-in-the-loop (SITL) simulation. We demonstrate our new controller both in simulation and on a physical team of small quadrotors carrying a payload.**

I. Introduction

Aerial vehicles can operate in places that are hard to reach by other robots. As such, they are well suited for collaborative assistance in a construction site, removal of rubble in a search-and-rescue scenario, or decommissioning of a nuclear power plant. Cable-driven payload transportation using multi-UAVs are beneficial because they do not require to carry manipulators or grippers onboard the quadrotors, thus allowing to transport heavier objects [1], which makes them in particular interesting to transport tools or supplies [2].

The dynamics of such system can be described explicitly through the interaction forces between the payload, cables and the UAVs [1]. In general, the dynamics are expressed using Newton's equations of motion [3], [4], [5], while augmenting the rotational dynamics of the UAVs [6].

There has been some advancement towards control algorithms for payload collaborative transport. A Leader-Follower paradigm has been proposed in several works [5], [7]. These methods use the interaction forces between the robots and the payload-cable system to ensure that the follower robot would be compliant with the leader. Although these methods do not require any communication between the UAVs, the dependability over the leader is risky in case of its failure. Other methods depend on constructing a cascaded design for the control law using either force [5], [6] or kinematic analysis [8] to provide control over the payload. In particular, the authors in [6], [9] introduced a geometric cascaded controller on $SE(3)$ for cable-driven payload transportation using multi-UAVs. The cascaded design allows the controller to compute the control input of each UAV given a reference trajectory for the payload. The stability of the controlled full dynamics is studied through Lyapunov analysis and it shows that the zero equilibrium of the tracking errors is exponentially stable. A recent paper [10] includes a simulator, an open-source implementation of one the controllers [6], and some real world experiments using their framework with different payloads. However, the cascaded structure imposes challenges in practice. The method requires either the measurement or estimating the accelerations of the payload, which is impractical due to the induced noise from both methods. Moreover, the controller is not designed to avoid robot/robot collision or cable tangling.

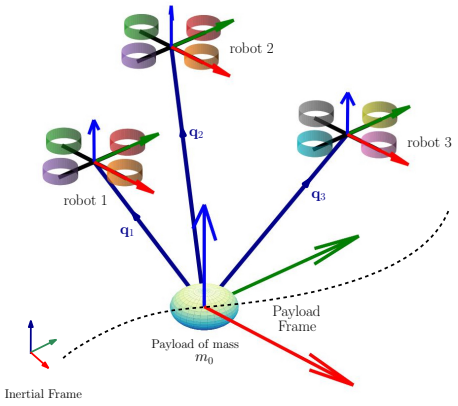

Fig. 1. Three quadrotors carrying a point mass payload.

In this paper, we address these shortcomings by augmenting the existing controllers [6], [9] with (quadratic) optimization, while retaining the existing stability results. Moreover, unlike [6], [9], we demonstrate our approach in physical test flights. Unlike the previous work [10], our simulator supports software-in-the-loop simulation and our efficient embedded controller implementation in C can be executed on significantly smaller UAVs in real-time.

II. Background

A. Single Quadrotor Dynamics

The dynamics of a single multirotor is modeled as a 6 degree-of-freedom floating rigid body with mass m and diagonal moment of inertia $\mathbf{J}$. The single quadrotor's state

The authors are with TU Berlin, Germany. {k.wahba, hoenig}@tu-berlin.de.

Code: https://github.com/IMRCLab/pyCrazyflie, and https://github.com/IMRCLab/crazyswarm2/tree/col-trans.

Video: https://youtu.be/yLbzvuXErXY

The research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 448549715.

comprises of the global position $\mathbf{p} \in \mathbb{R}^3$, global velocity $\mathbf{v} \in \mathbb{R}^3$, attitude rotation matrix $\mathbf{R} \in SO(3)$ and body angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. The dynamics can be expressed using Newton-Euler [11] equations of motion as follows

$$\dot{\mathbf{p}} = \mathbf{v}, \qquad m\mathbf{v} = m\mathbf{g} + \mathbf{R}\mathbf{f}_u, \tag{1a}$$

$$\dot{\mathbf{R}} = \mathbf{R}\mathbf{S}(\boldsymbol{\omega}), \qquad \mathbf{J}\dot{\boldsymbol{\omega}} = \mathbf{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau}_u, \tag{1b}$$

where $\boldsymbol{S}(\cdot)$ is a skew-symmetric mapping; $\mathbf{g} = [0; 0; -g]$ is the gravity vector; $\boldsymbol{f}_u = [0; 0; T]$ and $\boldsymbol{\tau}_u = [\tau_x; \tau_y; \tau_z]$. Denote the total wrench vector applied on the center-of-mass (CoM) of the quadrotor's body as $\boldsymbol{\eta} = [T; \tau_x; \tau_y; \tau_z]$. Where $\boldsymbol{\eta}$ is linearly related to the squared motor rotational rate (i.e., propeller speed) for m motors, where $\boldsymbol{\omega_m} = [w_1^2; \ldots; w_m^2]$. Such that $\boldsymbol{\eta} = \mathbf{B}_0\boldsymbol{\omega_m}$, where $\mathbf{B}_0$ is the actuation matrix [11].

B. Full System Dynamics

Consider a team of n quadrotors that are connected to a payload through massless cables (see Fig. 1). In the following, we summarize results presented in [6], [9] using our own notations for clarity. Throughout this work, the variables related to the payload are denoted by the subscript 0, and the variables for the i-th quadrotor are denoted by the superscript i such that $i \in \{1, \ldots, n\}$. The payload is described as a rigid body with mass m_0 and moment of inertia matrix $\mathbf{J}_0$. The cables are assumed to be always taut (i.e., modeled as rigid rods) each with length l_i. The states of the full system can be described as the global position $\mathbf{p}_0 \in \mathbb{R}^3$ and the velocity $\dot{\mathbf{p}}_0 \in \mathbb{R}^3$ of the payload; the attitude rotation matrix $\mathbf{R}_0 \in SO(3)$ and the body angular velocity $\boldsymbol{w}_0 \in \mathbb{R}^3$ of the payload; the unit directional vector $\mathbf{q}_i \in \mathbb{R}^3$ and the angular velocity $\boldsymbol{\omega}_i \in \mathbb{R}^3$ of each i-th cable, where $\mathbf{q}_i$ points from quadrotor i towards the payload. Hence, the state space has dimension $13 + 6n$.

Without loss of generality, consider the payload to be a point mass model with mass m_0. Given the states of the full system, the position of each quadrotor can be described as $\mathbf{p}_i = \mathbf{p}_0 + l_i\mathbf{q}_i$. Thus, the kinematics of the full system is described as

$$\dot{\mathbf{q}}_i = \boldsymbol{\omega}_i \times \mathbf{q}_i, \qquad \dot{\mathbf{p}}_i = \dot{\mathbf{p}}_0 + l_i\dot{\mathbf{q}}_i. \tag{2}$$

While the dynamics of the full system can be simply expressed using Euler-Lagrange equations.

C. Control Design

1) Single Quadrotor: Consider a single quadrotor control problem: given a tuple reference trajectory $\langle \mathbf{p}_r, \dot{\mathbf{p}}_r, \ddot{\mathbf{p}}_r, \psi_r \rangle$ for the quadrotor CoM, where $\mathbf{p}_r$ and $\boldsymbol{\psi}_r$ are the position and the heading of the quadrotor, respectively. We need a controller that tracks this reference trajectory. The controller in [12] is developed for the single quadrotor case. The controller consists of 2-*loop* cascaded design, where the outer loop computes the desired force control vector, which then computes T and the desired third body z-axis z_d. Thus, using differential flatness along with the the reference heading $\boldsymbol{\psi}_r$, the desired rotational states are computed and $\boldsymbol{\tau}_u$ tracks those desired values [11], [12].

2) Control Design for Rigid Body Payload: In order to achieve a controller for multi-UAVs transporting a shared payload (dynamics as defined in Section II-B), the single control design is not applicable for this case, as it cannot guarantee that the payload will follow a given reference trajectory. Moreover, the single controller will not be able to maintain the cables to be always in a taut mode.

The state-of-the-art controller [6] solves the multi-UAVs shared payload transportation problem, where the control problem is defined as follows. The output is a tuple reference trajectory $\langle \mathbf{p}_{0_r}, \dot{\mathbf{p}}_{0_r}, \ddot{\mathbf{p}}_{0_r}, \mathbf{R}_{0_r}, \boldsymbol{\omega}_{0_r} \rangle$, where $\mathbf{p}_{0_r}, \mathbf{R}_{0_r}, \boldsymbol{\omega}_{0_r}$ are the position, rotation matrix and angular velocity of the payload, respectively. The output is a collaborative/centralized controller that computes the motor signals to track this reference trajectory.

This controller consists of a 4-*loop* cascaded design. The first step in the first loop computes the desired resultant control forces, $\mathbf{F}_d$, and torques, $\mathbf{M}_d$, of the payload to track the payload reference trajectory. Let the control force applied by each i-th quadrotor on its cable be $\mathbf{u}_i = \mathbf{R}_i\mathbf{f}_{u_i}$. Denote that $\mathbf{u}_i^{\parallel} \in \mathbb{R}^3$ and $\mathbf{u}_i^{\perp} \in \mathbb{R}^3$ are the orthogonal projection of $\mathbf{u}_i$ along $\mathbf{q}_i$ and to the plane normal to $\mathbf{q}_i$, respectively, i.e., $\mathbf{u}_i = \mathbf{u}_i^{\parallel} + \mathbf{u}_i^{\perp}$. Then, the first loop outputs the desired cable unit directional vectors $\mathbf{q}_{i_d}$ in order to achieve $\mathbf{F}_d$ and $\mathbf{M}_d$. Let $\boldsymbol{\mu}_{i_d}$ be the desired i-th cable force and $\mathbf{P} \in \mathbb{R}^{6\times 3n}$ be a matrix that maps the forces and torques of the payload CoM to $\boldsymbol{\mu}_{i_d}$. The linear relation between $\boldsymbol{\mu}_{i_d}$ and $\mathbf{F}_d, \mathbf{M}_d$ is defined as

$$\mathbf{P}\begin{bmatrix}(\mathbf{R}_0^T\boldsymbol{\mu}_{1_d})^T & \ldots & (\mathbf{R}_0^T\boldsymbol{\mu}_{n_d})^T\end{bmatrix} = \begin{bmatrix}(\mathbf{R}_0^T\mathbf{F}_d)^T & \mathbf{M}_d^T\end{bmatrix}. \tag{3}$$

For any $\mathbf{F}_d$, $\mathbf{M}_d$, there exists at least one solution for $\boldsymbol{\mu}_{i_d}$, e.g., the minimum-norm solution

$$\boldsymbol{\mu}_{i_d} = diag[\mathbf{R}_0, \ldots, \mathbf{R}_0]\mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\begin{bmatrix}\mathbf{R}_0^T\mathbf{F}_d \\ \mathbf{M}_d\end{bmatrix} \tag{4}$$

Then, we compute $\mathbf{q}_{i_d}$ using $\boldsymbol{\mu}_{i_d}$ by applying the definition

$$\mathbf{q}_{i_d} = \frac{\boldsymbol{\mu}_{i_d}}{||\boldsymbol{\mu}_{i_d}||}. \tag{5}$$

The second loop computes the control force $\mathbf{u}_i$ applied by each quadrotor on the payload. In particular, $\mathbf{u}_i^{\parallel}$ is first computed by projecting $\boldsymbol{\mu}_{i_d}$ on the current cable force vector $\boldsymbol{\mu}_i$, in addition to non-linear terms that linearize the translational dynamics of the payload. In other words, when $\boldsymbol{\mu}_i \to \boldsymbol{\mu}_{i_d}$, then $\mathbf{F}_d, \mathbf{M}_d$ are tracked. In order to track the desired cable forces $\boldsymbol{\mu}_{i_d}$, then $\mathbf{u}_i^{\perp}$ is also used, which is responsible to track the desired cable forces $\boldsymbol{\mu}_{i_d}$ by tracking the desired unit directional vector, i.e., $\mathbf{q}_i \to \mathbf{q}_{i_d}$, of the cables. It is straightforward to verify that when $\mathbf{q}_i = \mathbf{q}_{i_d}$ then the resultant force and moment acting on the payload become identical to their desired values.

After computing $\mathbf{u}_i$ vector, it is used in the third loop that outputs the thrust magnitude T (i.e., $\mathbf{f}_u$) and the desired third body vector $\mathbf{z}_d$. Thus, using differential flatness, we can define the desired rotation matrix and its derivatives. Finally, the fourth loop is responsible to compute $\boldsymbol{\tau}_u$ to track the desired attitude values (i.e., attitude controller).

3) Control Design for Point Mass Payload: We now consider a special case for the model of the payload to be a point mass. Then, only the translational dynamics of the full model are relevant and Eq. (3) is reduced to

$$\mathbf{P} \begin{bmatrix} \boldsymbol{\mu}_{1_d}^T & \dots & \boldsymbol{\mu}_{n_d}^T \end{bmatrix}^T = \mathbf{F}_d. \tag{6}$$

Another solution for the point mass model [9], requires a reference trajectory for the cable directions $\mathbf{q}_{i_d}$ as additional input and offers two modes. The first mode is *Relative Formation Control*, which is similar to virtual structure approaches in formation control and only works if $n \geq 1$. The second mode is *Inertial Formation Control* and only usable with more than two quadrotors. Then the desired directions of the cables are defined by smooth curves $\mathbf{q}_{i_d} \in \mathcal{S}^2$. Thus, the computation of $\mathbf{u}_i^{\parallel}$ and $\mathbf{u}_i^{\perp}$ are the same as the rest of the state-of-the-art controller for the rigid payload case.

III. Approach

The baseline controller [6] for a rigid body payload exhibits certain challenges and issues. In particular, while tracking the reference trajectory for the payload, this controller does not take the size of the payload into account. If the payload's size is small compared to the cable lengths, then collision between the quadrotors might occur, or cables might tangle. Moreover, the baseline does not consider required formation changes between configurations while tracking the payload's reference trajectory, which might cause collisions between the quadrotors and obstacles during loading or unloading of the payload. In other words, this baseline controller includes neither inter-robot nor robot/obstacle collision avoidance.

The authors had prior work on a similar class of problems, where the model of the payload was assumed to be a point mass [9], and this controller allows to design a reference trajectory for the cable direction $\mathbf{q}_{i_d}$, which can be used to avoid inter-robot collisions. However, it will be susceptible to other problems which the controller is not designed to handle. While a desired formation can be provided for the team of robots, there is no guarantee that the load distribution is feasible over the cables and the UAVs while also keeping the quality of the trajectory tracking. In fact, the tracking quality might degrade if the formation is close to singularity formations [8] and thus the full system might become unstable. Moreover, the designed desired forces to be applied by the UAVs on the cables cannot be limited in magnitude, which might cause stability loss due to actuation limits.

Finally, the baseline and its prior work support homogeneous systems only and do not consider heterogeneous team members for the multi-UAVs payload transportation class of problems. By heterogeneous we refer to differences of team members in robot type (multirotor or fixed wing), size, or sensing capabilities.

A. Problem Description

Consider *n* quadrotor that are transporting a rigid body payload with massless cables of length l_i. Given a reference trajectory tuple for the payload $\langle \mathbf{p}_{0_r}(t), \dot{\mathbf{p}}_{0_r}(t), \ddot{\mathbf{p}}_{0_r}(t), \mathbf{R}_{0_r}(t), \boldsymbol{\omega}_{0_r}(t) \rangle$, where $t \in [t_0, t_f]$ and $\mathbf{p}_{0_r}, \mathbf{R}_{0_r}, \boldsymbol{\omega}_{0_r}$ are the position, rotation matrix and angular velocity of the payload, respectively. Our objective is to find the optimal control input of each quadrotor $\boldsymbol{\eta}_i$ that minimizes the error between the states of the payload and the reference trajectory, while respecting the dynamic model of the full system, the quadrotors avoid collision with themselves and the obstacles. Formally:

$$\min_{\boldsymbol{\eta}_i} ||\mathbf{p}_{0_r}(t) - \mathbf{p}_0(t)|| \tag{7}$$
$$\text{s.t.} \begin{cases} \dot{\mathcal{X}} = f(\mathcal{X}, \boldsymbol{\eta}_i) \\ \boldsymbol{\eta}_i \in \mathcal{U}^{(i)} \\ \mathbf{p}_i(t) \in \mathcal{W}_{\text{free}}^{(i)}, \quad i \in \{1, \dots, n\} \end{cases},$$

where the equality constraint represents the dynamic model (Section II-B); the second equation is the control input of each *i*-th quadrotor $\boldsymbol{\eta}_i$ that must belong to its control space $\mathcal{U}^{(i)}$; and the final constraint represents the collision-free workspace $\mathcal{W}_{free}^{(i)}$ where it is defined disjointly for each *i*-th quadrotor and it includes the inter-robot collision constraints.

While Eq. (7) is a centralized formalization, we are only interested in distributed solutions, which means that each *i*-th quadrotor runs the optimization program to solve for its own control input $\boldsymbol{\eta}_i$. In addition, the optimization problem is running online for each time step.

We make the following assumptions. In terms of modeling, it is assumed that the cables are always in taut mode (i.e., they are modeled as massless rigid rods with a predefined length l_i). It is beneficial for the collision-free space to belong to a convex workspace to be able to present faster solutions from the optimization program. For this paper, we assume that the constraints are provided as user input in form of hyperplanes or to be precise *half-spaces*. A hyperplane $\mathcal{H}$ in $\mathbb{R}^d$ can be formulated by a normal vector $\mathbf{n}$ and an offset a as $\mathcal{H} = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{n}^T\mathbf{x} + a = 0\}$. A half-space $\tilde{\mathcal{H}}$ in $\mathbb{R}^d$ is a subset of $\mathbb{R}^d$ that is bounded by a hyperplane such that $\tilde{\mathcal{H}} = \{\mathbf{x} \in \mathbb{R}^d \mid \mathbf{n}^T\mathbf{x} + a \leq 0\}$.

B. Quadratic Optimization Problem

The baseline considers *n* quadrotor that are connected to a payload with massless cables. Given a reference trajectory tuple $\langle \mathbf{p}_{0_r}(t), \dot{\mathbf{p}}_{0_r}(t), \ddot{\mathbf{p}}_{0_r}(t) \rangle$ for a point mass payload, our objective is to track this reference trajectory while taking into account the mentioned problems.

One major issue is caused by the design choice for $\mathbf{q}_{i_d}$. Let us consider the solution provided in the rigid payload case for computing $\boldsymbol{\mu}_{i_d}$ to be applied to the point mass case. It is clear that minimum-norm solution of Eq. (6) will provide a solution for $\boldsymbol{\mu}_{i_d}$. In fact, the minimum-norm solution is generated by solving the optimization problem

$$\min_{\boldsymbol{\mu}_{i_d}} \ \frac{1}{2}||\boldsymbol{\mu}_{i_d}||^2 \tag{8}$$
$$\text{s.t.} \left\{ \mathbf{P} \begin{bmatrix} \boldsymbol{\mu}_{1_d}^T & \dots & \boldsymbol{\mu}_{n_d}^T \end{bmatrix}^T = \mathbf{F}_d \right. .$$

However, the proposed solution presented for the desired cable forces will not be valid in practice for the point mass case. Since the minimum-norm solution does not take into account a proper formation for the UAVs, this will lead to inter-UAV collision.

Consequently, it makes sense to use the null-space of the minimum-norm solution for the desired cable forces vector. In particular, we generalize our approach by adding additional (hard) constraints to the optimization problem. Thus, our approach uses a ***Quadratic Optimization Problem*** (QP) formulation for efficiently solving the inter-UAV collision problem while tracking the reference trajectory with a feasible formation for the quadrotor. In particular, we extend Eq. (8) as follows

$$\min_{\mu_{i_d}} \ \frac{1}{2}||\boldsymbol{\mu}_{i_d}||^2 \tag{9}$$
$$\text{s.t.} \begin{cases} \mathbf{P}\begin{bmatrix}\boldsymbol{\mu}_{1_d}^T & \dots & \boldsymbol{\mu}_{n_d}^T\end{bmatrix}^T = \mathbf{F}_d \\ \begin{bmatrix}\mathbf{n}_1^T & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \mathbf{n}_m^T\end{bmatrix}\begin{bmatrix}\boldsymbol{\mu}_{1_d} \\ \vdots \\ \boldsymbol{\mu}_{n_d}\end{bmatrix} + \begin{bmatrix}a_1 \\ \vdots \\ a_m\end{bmatrix} \leq \mathbf{0}_{m\times 1} \end{cases},$$

where $\mathbf{n_i} \in \mathbb{R}^{3\times 1}$ and $a_i \in \mathbb{R}^{3\times 1}$ for $i \in 1, \dots, m$ define a *half-space* and m is the number of hyperplanes with $m \geq n$.

Our QP formulation is presented for the point mass payload case, however, it can be generalized and extended easily to the rigid case with minor changes. In particular, the first change to be considered is the provided reference trajectory tuple. It must be augmented with a rotational reference trajectory for the attitude dynamics of the rigid payload. Consequently, the desired virtual moments $\mathbf{M}_d$ must be computed. Finally, the last change to be considered is the equality constraint in Eq. (9) must be substituted with Eq. (3). Moreover, the proposed approach will solve the inter-UAV collision problem and cable tangling whenever the rigid payload is small in size.

IV. Experiments

To validate the functionality of our proposed methodology, we now present the results of simulation comparing between our approach with the state-of-the-art baseline [6] as well as experimental results on physical team of two quadrotors transporting a point mass payload.

A. Target Platform

In order to test our approach, we use quadrotors of type Bitcraze Crazyflie 2.1 (CF). These are small ($9\,\mathrm{cm}$ rotor-to-rotor) and lightweight ($34\,\mathrm{g}$) products that are commercially available. The physical parameters are identified in prior work [13]. Our controller and an extended Kalman filter are running on-board the STM32 microcontroller ($168\,\mathrm{MHz}$, $192\,\mathrm{kB}$ RAM). The control algorithm implemented on-board is written using the *C (99)* language.

B. Simulation

1) Simulator Development Process: We found that an iterative development method reduces challenges regarding the sim-to-real gap. We developed and tuned our approach through three stages.

In the first stage, we implement our experiments in simulation to present our results using only *Python*. We start with the dynamic model of a single quadrotor and a state-of-the-art single UAV controller [12]. Then, we extend the simulation to the multi-UAVs payload transportation problem using the dynamics from Section II-B. In all cases we found that a simple **Euler Integration** method and a small timestep provide good results. All physical parameters used in the simulation for the quadrotor are based on an existing system identification [13]. Finally, we implement the baseline and our approach. For our controller, we write the QP program using an open source Python-embedded modeling language for convex optimization problems, CVXPY [14].

In the second stage, we port our baseline controller from Python to C and add it to the quadrotor's firmware. Then, we generate Python bindings for the relevant firmware code using SWIG. We verify the semantic equivalence between the two implementations by using those Python bindings in our simulator in a Software-in-The-Loop (SITL) fashion.

In the third stage, we port our own controller, which requires to solve the QP (Eq. (9)) on-board the STM32 microcontroller. In particular, our objective is to migrate our QP formulation from a Python-embedded CVXPY language to the C language for the STM32 controller. First, we write the same optimization program using the OSQP (Operator Splitting Quadratic Program) solver [15]. The difference between CVXPY and the *Python* interface of OSQP is the type of the matrices used to construct the problem. In CVXPY, the matrices of the constraints are written using *numpy* arrays, while in OSQP, the constraints matrices are written in sparse CSC format. CVXPY is a framework that solves different types of optimization problems using different solvers, while OSQP is a solver which solves convex quadratic programs. In fact, CVXPY can use OSQP as a solver for a QP optimization program. Moreover, OSQP can generate tailored C code that compiles into a fast and reliable solver for the given family of QP problems in which the problem data, but not its dimensions, change between problem instances. Thus we use this fact to migrate the QP written in OSQP *Python* interface, and generate C code which can be augmented to our Crazyflie firmware system. Finally, we test the optimization program and the controller using the firmware Python bindings. Thus, the results and the efficiency of our controller can be tested directly. In addition, this allows us to tune the controller as close as possible to the required actual values.

2) Numerical Example: The following example uses the SITL firmware controller based on the firmware Python bindings. We define a test case using the point mass model for the payload. We consider an example of two Crazyflie quadrotors each of mass $m_i = 34\,\mathrm{g}$, transporting a point

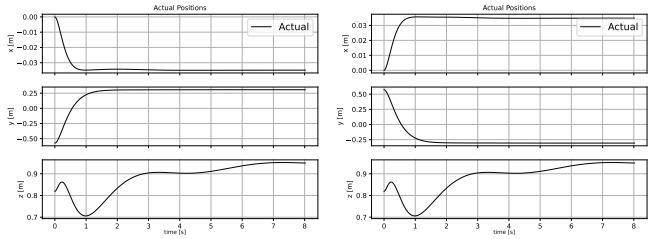

Fig. 2. Position state of both quadrotors when they are solving the QP are executed in a SITL simulation while tracking the hovering reference trajectory of the payload and avoiding inter-robot collision using the predefined hyperplanes.

mass payload of mass $m_0 = 10\,\mathrm{g}$. The length of the both cables are set to $l_i = 1\,\mathrm{m}$ which are attached from the payload to the CoM of the quadrotors.

The initial conditions of the payload position and velocity are chosen as $\mathbf{p}_0(0) = \mathbf{0}_{3\times1}$, $\mathbf{v}_0(0) = \mathbf{0}_{3\times1}$, respectively. Moreover, the initial conditions of the cable directional unit vectors $\mathbf{q}_i(0)$ are chosen to be of $35°$ (i.e., $\frac{7}{36}\pi$ rad) relative to the z-axis of payload frame, such that

$$\mathbf{q}_1(0) = \begin{bmatrix}\frac{7}{36}\pi & 0 & 0\end{bmatrix}^T \mathbf{q}_2(0) = \begin{bmatrix}-\frac{7}{36}\pi & 0 & 0\end{bmatrix}^T. \quad \text{(10a)}$$

The initial conditions of the angular velocities of the cables are set to $\boldsymbol{\omega}_i(0) = \mathbf{0}_{3\times1}$. Then, the initial conditions of the positions and velocities for the quadrotors are

$$\mathbf{p}_i = \mathbf{p}_0 + l_i\mathbf{q}_i \qquad \dot{\mathbf{p}}_i = \dot{\mathbf{p}}_0 + l_i\dot{\mathbf{q}}_i. \quad \text{(11a)}$$

The reference trajectory of the payload's position is hovering at the origin, such that $\mathbf{p}_{0_r} = \dot{\mathbf{p}}_{0_r} = \ddot{\mathbf{p}}_{0_r} = \mathbf{0}_{3\times1}$. The desired hyperplanes are assumed to be predefined such that the two normal directions with an offset $\mathbf{a} = \mathbf{0}_{2\times1}$ are

$$\mathbf{n}_1^T = \begin{bmatrix}0.0194 & -0.1091 & 0.04\end{bmatrix}^T, \quad \text{(12a)}$$
$$\mathbf{n}_2^T = \begin{bmatrix}-0.0194 & 0.1091 & 0.04\end{bmatrix}^T. \quad \text{(12b)}$$

These two normal directions describe the hyperplanes to be $20°$ relative to the z-axis of payload frame. The plots in Fig. 2 show the positions of both quadrotors starting from the initial positions computed from $\mathbf{p}_0(0)$ and $\mathbf{q}_i(0)$ using Eq. (11a), evolving until reaching the final value. The final values are decided by the QP that provides the $\boldsymbol{\mu}_{i_d}$, which then is achieved by the controller when $\mathbf{u}_i$ is computed and tracked by each quadrotor. Notice that with this approach, we can control the distance between the quadrotors and avoid inter-robot collisions, while implicitly (i.e., using $\boldsymbol{\mu}_{i_d}$) providing feasible $\mathbf{u}_i$ for the quadrotors to track. In contrast, using the null-space of the minimum-norm solution to generate $\boldsymbol{\mu}_{i_d}$ in the baseline controllers results in an inter-robot collision, because the minimum-norm solution for $\boldsymbol{\mu}_{i_d}$ in the hovering case will require for each quadrotor to cancel out half the weight of the payload while hovering on top of the payload.

C. Physical Flights

We now verify our approach in physical flight tests with for two Bitcraze Crazyflies. As this is a distributed system, each quadrotor has its own controller running on-board and generating its own control input. We use Crazyswarm2, which is based on Crazyswarm [16] but uses ROS2 [17] to control and send commands for multiple Crazyflies. In order to measure the current state, we equip each quadrotor and the payload with a single reflective marker for position tracking at $100\,\mathrm{Hz}$ using an OptiTrack motion capture system. The angular states of each quadrotor are estimated on-board using an Extended Kalman Filter. The experimental setup for our physical experiment is similar to the simulation environment. In particular, the quadrotors and the payload has the same mass as in simulation. However, the lengths of the cables are slightly different $l_1 = 0.77\,\mathrm{m}$ and $l_2 = 0.705\,\mathrm{m}$ and the gains of the controller are slightly tuned to enhance the tracking of the cables trajectories and of the payload states. We use nylon strings for transporting the point mass payload, that are very thin and lightweight so we can align with the massless cables assumption.

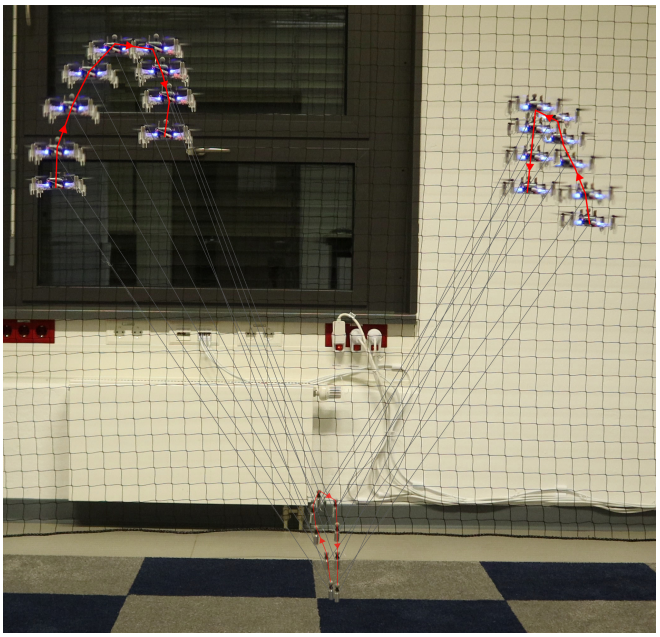

Fig. 3. Physical experimental setup and snapshots of the execution of the hovering reference trajectory for the payload. The red lines and the arrows show the trajectory of both quadrotors and the payload.

1) Physical Experiment Setup: We use the same initial conditions as in Section IV-B.2 and similar hovering reference trajectory but with $\mathbf{p}_{0_r} = [0; 0; 0.4]$. Moreover, the numerical values for the normal of the hyperplanes $\boldsymbol{n}_i$ and the offset $\boldsymbol{a}_i$ are also predefined as in Eq. (12).

2) Challenges: Executing the baseline and our controller on physical quadrotors was challenging (both [9] and [6] were only shown in simulation). The first challenge is setting the payload and the cables (i.e., the quadrotors) to their predefined initial conditions. Our controller assumes that the cables are always taut and modeled as massless rigid rods. Thus, the controller can not be active during the takeoff stage because the cables are still slack and not in full taut mode. In order to solve this challenge, we used the fact the Crazyflie

can switch controllers mid-flight. In particular, we take off with the payload to the desired initial conditions while using the on-board single quadrotor controller and then switch to our controller.

The second challenge is in the second loop of the controller that computes $\mathbf{u}_i^{\parallel}$, which requires either measuring or estimating the acceleration of the payload. This information is needed to linearize the translational dynamics of the payload, see Section II-C.2. Initially, we attempted to estimate the acceleration of the payload by applying numerical differentiation twice on the position of the payload. However, the resulting signal is very noisy and causes the quadrotors to crash. In order to overcome this challenge, we propose two solutions, each with a trade-off. The first solution is to use the reference acceleration of the payload, rather than the estimated current acceleration. This approach results in a numerically stable computation for $\mathbf{u}_i^{\parallel}$. However, this assumes that the payload states track the reference trajectory well, which might not be the case if any external disturbance exists or when the initial conditions of the reference trajectory are not accurate. The second proposed solution is to set the estimated acceleration to zero. This works well for slow motions, but is not stable when the reference trajectory is aggressive.

The third challenge is executing the QP on-board and generating solutions with a frequency close the to that of the controller. We optimize the generated C code from OSQP to produce faster solutions within each loop. In particular, the runtime of the QP for each loop was reduced from $2000\,\mu s$ to $700\,\mu s$ by switching to 32-bit floating point operations.

3) Experimental Results: An example of the actual trajectory of the full system is shown in Fig. 3. The results show the frames and the trajectory of each quadrotor and of the payload (i.e., red line), starting from the defined initial conditions towards the final state. The results of Fig. 3 show that the QP in Eq. (9) generates solution for the cable forces $\boldsymbol{\mu}_{i_d}$ that will respect the reference hovering trajectory while taking into account the inter-robot collision. The final state of the trajectory shows that the quadrotors, and the cable does not exceed the predefined hyperplanes while having the payload hovering at the reference position. Initially, when the quadrotors were taking off, each with the single quadrotor controller [12] (i.e., before switching to our controller), it was noticeable that the payload did not hold a stable position. When we switched to our controller, it was visible that the stability of the payload trajectory tracking improved significantly.

V. Conclusion and Future Work

We present a new approach for multi-UAV cable-suspended payload transportation. Our controller tracks a predefined reference trajectory for the payload while avoiding inter-robot and robot-obstacle collisions. Our optimization-based approach is a generalized and unified method for the baseline controller [6] and its prior work [9]. For efficiency, we use a QP to compute the desired cable forces $\boldsymbol{\mu}_{i_d}$ that considers inter-robot collision and tangling of the cables. Since we did not change any other parts of the existing baseline, the prior stability analysis still holds. We present a physical real experiment for our controller running on-board of two quadrotors carrying a point mass payload. We present an iterative development method that reduces challenges regarding the sim-to-real gap.

In the future, we plan to validate our approach for different reference trajectories with more than two quadrotors. In addition, we want to extend our approach to support rigid body payloads. Moreover, our current work relies on the assumption of having predefined hyperplanes. We will investigate new methods to compute feasible hyperplanes efficiently, e.g. by using support vector machines. In general, we want to demonstrate motion planning and control for payload transport using multiple quadrotors in challenging scenarios.

References

[1] C. Masone, H. H. Bülthoff, and P. Stegagno, "Cooperative transportation of a payload using quadrotors: A reconfigurable cable-driven parallel robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1623–1630.

[2] C. Gabellieri, M. Tognon, L. Pallottino, and A. Franchi, "A study on force-based collaboration in flying swarms," in *Int. Conf. Swarm Intell.*, 2018, pp. 3–15.

[3] P. O. Pereira and D. V. Dimarogonas, "Control framework for slung load transportation with two aerial vehicles," in *Proc. IEEE Conf. Decis. Control*, 2017, pp. 4254–4259.

[4] E. Tuci, M. H. Alkilabi, and O. Akanyeti, "Cooperative object transport in multi-robot systems: A review of the state-of-the-art," *Frontiers in Robotics and AI*, vol. 5, p. 59, 2018.

[5] M. Tognon, C. Gabellieri, L. Pallottino, and A. Franchi, "Aerial co-manipulation with cables: The role of internal force for equilibria, stability, and passivity," *IEEE Trans. Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2577–2583, 2018.

[6] 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.

[7] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto, "Robust collaborative object transportation using multiple mavs," *I. J. Robotics Res.*, vol. 38, no. 9, pp. 1020–1044, 2019.

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

[9] T. Lee, K. Sreenath, and V. Kumar, "Geometric control of cooperating multiple quadrotor uavs with a suspended payload," in *Proc. IEEE Conf. Decis. Control*, 2013, pp. 5510–5515.

[10] G. Li, X. Liu, and G. Loianno, "RotorTM: A flexible simulator for aerial transportation and manipulation," *arXiv preprint arXiv:2205.05140*, 2022.

[11] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.* IEEE, 2011, pp. 2520–2525.

[12] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Proc. IEEE Conf. Decis. Control*, 2010, pp. 5420–5425.

[13] J. Förster, "System identification of the crazyflie 2.0 nano quadrocopter," B.S. thesis, ETH Zurich, 2015.

[14] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[15] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[16] J. A. Preiss, W. Hönig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3299–3304.

[17] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022.