Direct Force Feedback Control and Online Multi-task Optimization for Aerial Manipulators

Gabriele Nava, Quentin Sablé, Marco Tognon, Daniele Pucci, Antonio Franchi

Direct Force Feedback Control and Online Multi-task Optimization for Aerial Manipulators

Gabriele Nava[1,2], Quentin Sablé[3], Marco Tognon[3] , Daniele Pucci[1], Antonio Franchi[3,4]

***Abstract*— In this paper we present an optimization-based method for controlling aerial manipulators in physical contact with the environment. The multi-task control problem, which includes hybrid force-motion tasks, energetic tasks, and position/postural tasks, is recast as a quadratic programming problem with equality and inequality constraints, which is solved online. Thanks to this method, the aerial platform can be exploited at its best to perform the multi-objective tasks, with tunable priorities, while hard constraints such as contact maintenance, friction cones, joint limits, maximum and minimum propeller speeds are all respected. An on-board force/torque sensor mounted at the end effector is used in the feedback loop in order to cope with model inaccuracies and reject external disturbances. Real experiments with a multi-rotor platform and a multi-DoF lightweight manipulator demonstrate the applicability and effectiveness of the proposed approach in the real world.**

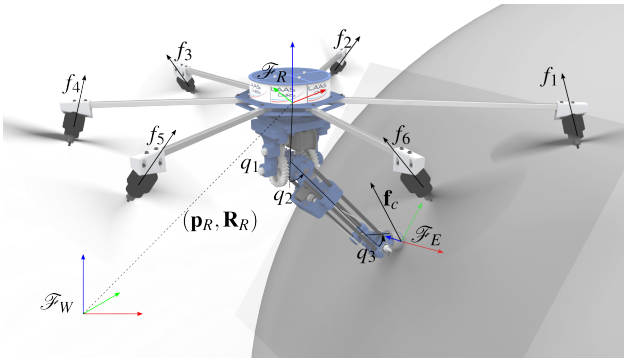

Fig. 1: CAD rendering of the aerial manipulator designed and developed at LAAS-CNRS.

I. Introduction

Nowadays, Unmanned Aerial Vehicles (UAVs) are widespread and employed in several application scenarios such as surveillance, remote monitoring and aerial photography. The recent development of *aerial manipulators* physically interacting with the environment, significantly enlarged the number of feasible tasks spanning from manipulation and grasping of objects to the contact-based inspection [1]. Such *interaction tasks* often require the aerial manipulator to precisely regulate both position of the contact point (possibly moving) and the amount of force that is exerted on it.

There is a rich literature concerning the modeling and control of aerial manipulators [2]. Regarding the motion control, different methods can be applied, e.g., full dynamic inversion [3], flatness-based [4], and adaptive sliding mode [5]. On top of those, if the system is over actuated with respect to the desired task, a nullspace-based behavioral control [6] or a task priority controller [7] can be applied at the kinematic level exploiting the redundancy to achieve secondary tasks (e.g., obstacle avoidance, minimum energy consumption, etc.). In particular, the latter acts as a local motion planner providing the reference trajectory of each degree of freedom of the robot to the low level motion controller. The main limitation of this control architecture is the inability to consider additional hard constraints to be respected, e.g., input/state bounds, dynamics of the system which is not taken into account at the kinematic level, friction cone in case of interaction, etc. The mentioned constraints are very important because, if not respected, they might bring the whole system to instability in real scenarios.

From the interaction control side, one of the most common strategy is to use an admittance filter. In [8] and [9], such technique has been employed to control the interaction force in the case of a fully-actuated platform equipped with a rigid tool, and of an under-actuated platform equipped with a robotic arm, respectively. In [10], a passivity-based controller has been proposed as well. However, in those works, the force-control is only *indirect*. In fact, the force is not directly measured but rather estimated using the robot dynamics. Since the method is strongly model based, it is thus prone to error in case of parameter uncertainties. Furthermore, if the system is affected by external disturbances, it might not be possible to discriminate them from the interaction forces. A first attempt in using a *direct* force feedback can be found in [11] where a force sensor has been attached to the end-effector. Nevertheless, this feedback is not used to precisely control the interaction force but is rather used in an impedance control framework to make the end-effector compliant.

In this work we propose a *whole-body* force control strategy for aerial manipulators that allows, in a unique formulation, to achieve hybrid position/force control, considering multi-task optimization under hard-constraints. We shall show that with the proposed control framework, the aerial manipulator can perform different contact-based tasks using the same strategy, only changing the tasks priority and

[1]Dynamic Interaction Control, Istituto Italiano di Tecnologia, Genova, Italy gabriele.nava@iit.it,
[2]DIBRIS, University of Genova, Genova, Italy,
[3]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France, marco.tognon@laas.fr, quentin.sable@laas.fr
[4]Robotics and Mechatronics lab, Faculty of Electrical Engineering, Mathematics & Computer Science, University of Twente, Enschede, The Netherlands a.franchi@utwente.nl

This research was partially supported by the ANR, Project ANR-18-CE33-0001 'The Flying Coworker'.

contact-constraints. In our framework, we consider several control objectives organized in a weighted prioritization. The controller is implemented in the form of a Quadratic Programming (QP) optimization, which is a rather known technique in the field of humanoid robotics [12], [13], but whose applicability and effectiveness was never demonstrated until now in the challenging field of aerial manipulators – where the physical interaction tasks are made very complex by, e.g., the absence of stabilizing contacts, limited propeller forces, inaccurate and time varying aerodynamics models, and mechanical vibrations.

Furthermore, to provide an accurate interaction control, our controller implements a *direct* force-feedback employing a 6-axis force-torque sensor (FT) mounted on the manipulator end-effector. We shall show through real experiments that the FT sensor feedback ensures the robustness of the proposed control law with respect to disturbances and modeling errors. The effectiveness of the overall control framework has been shown through real experiments for three different tasks: 1) pushing under external disturbances, 2) sliding while pushing and 3) high-acceleration motions exploiting contacts, but of course these are just a few of the many real world tasks can be achieved with the proposed method.

The remainder of the paper is organized as follows: Sec. II recalls the notation, the dynamics model of the aerial manipulator and the rigid contact model. Sec. III details the task-based control strategy and the QP optimization problem. Simulations and experimental results with a real hexarotor equipped with a three degrees of freedom manipulator are presented in Sec. IV. Conclusions and perspectives end the paper.

II. Modeling

We consider an aerial manipulator composed of an aerial platform equipped with a robotic manipulator as in Fig. 1. Given the flying nature of the system, we model it as a *floating base* system [14]. We assume that the robotic arm is composed of $n+1$ links, connected by n actuated joints with a single degree of freedom each.

To describe the state of the aerial manipulator we define a world frame $\mathscr{F}_W = \{O_W, \boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W\}$, with arbitrarily placed origin O_W, and unit axes $(\boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W)$ such that $\boldsymbol{z}_W$ points in the opposite direction of the gravity vector. An additional frame $\mathscr{F}_R = \{O_R, \boldsymbol{x}_R, \boldsymbol{y}_R, \boldsymbol{z}_R\}$ is rigidly attached to the aerial platform base. Finally, we rigidly attach a frame $\mathscr{F}_E = \{O_E, \boldsymbol{x}_E, \boldsymbol{y}_E, \boldsymbol{z}_E\}$ to the end-effector of the robotic arm such that $\boldsymbol{z}_E$ is parallel to the last link of the manipulator.

The robot configuration is then given by $\boldsymbol{q} = (\boldsymbol{p}_R, \boldsymbol{R}_R, \boldsymbol{q}_A) \in \mathscr{C} \subseteq \mathbb{R}^3 \times SO(3) \times \mathbb{R}^n$ where $\boldsymbol{p}_R \in \mathbb{R}^3$ and $\boldsymbol{R}_R \in SO(3)$ represent the position and orientation of $\mathscr{F}_R$ with respect to (w.r.t.) $\mathscr{F}_W$, respectively, while $\boldsymbol{q}_A \in \mathbb{R}^n$ are the manipulator joint angles which characterize the configuration of the robotic arm. The velocity relative to $\boldsymbol{q}$ is given by $\boldsymbol{v} = [\boldsymbol{v}_R^\top \ \boldsymbol{\omega}_R^\top \ \boldsymbol{v}_A^\top]^\top \in \mathbb{R}^{6+n}$, where $\boldsymbol{v}_R \in \mathbb{R}^3$ and $\boldsymbol{\omega}_R \in \mathbb{R}^3$ are the aerial platform linear and angular velocities, and $\boldsymbol{v}_A \in \mathbb{R}^n$ are the joints velocities. More precisely, $\boldsymbol{\omega}_R$ is the angular velocity of $\mathscr{F}_R$ w.r.t. $\mathscr{F}_W$ expressed in $\mathscr{F}_R$, which satisfies[1] $\dot{\boldsymbol{R}}_R = \boldsymbol{S}(\boldsymbol{\omega}_R)\boldsymbol{R}_R$.

The system equations of motions can be computed by applying the Euler-Poincaré formalism [15, Ch. 13.5] :

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{c}(\boldsymbol{q},\boldsymbol{v}) + \boldsymbol{g}(\boldsymbol{q}) = \begin{bmatrix}\mathbf{w}_R^\top \boldsymbol{\tau}_A^\top\end{bmatrix}^\top + \boldsymbol{J}_E(\boldsymbol{q})^\top \mathbf{w}_E, \quad (1)$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(n+6)\times(n+6)}$ is the mass matrix, $\boldsymbol{c}(\boldsymbol{q},\boldsymbol{v}) \in \mathbb{R}^{n+6}$ accounts for the Coriolis and centrifugal effects, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^{n+6}$ is the gravity vector, $\mathbf{w}_R \in \mathbb{R}^6$ is the control wrench (forces and moments) applied to the aerial vehicle by the propellers, and $\boldsymbol{\tau}_A \in \mathbb{R}^n$ are the joint torques of the manipulator. We assume that the interaction with the environment occurs at the manipulator end-effector only. When the robot end-effector is in contact with the environment, the external wrench $\mathbf{w}_E \in \mathbb{R}^6$ has to be included in the equations. The Jacobian $\boldsymbol{J}_E(\boldsymbol{q}) \in \mathbb{R}^{6\times(n+6)}$ is the map between the system velocity $\boldsymbol{v}$ and the linear and angular velocities of the end-effector at the contact location. It is assumed that for the applications proposed in this paper the aerodynamics phenomena such as, e.g., wind effect, blade flapping, cross interference between propellers, and ground/wall effect do not significantly affect the robot dynamics, and therefore can be neglected. Considering the aerial vehicle actuated by a set of $p \in \mathbb{N}$ rigidly attached propellers, the control wrench $\mathbf{w}_R$ can be conveniently rewritten as a function of the propellers angular rates. In particular we consider, as usual, a quadratic relation between propeller angular rates and corresponding generated thrust:

$$\mathbf{w}_R = \boldsymbol{G}_w(\boldsymbol{q})\boldsymbol{\omega}_P^2 := \boldsymbol{G}_w(\boldsymbol{q})(\boldsymbol{\omega}_P \odot \boldsymbol{\omega}_P), \quad (2)$$

where $\boldsymbol{\omega}_P \in \mathbb{R}^p$ is the vector of propellers angular rates, $\odot$ is the component-wise product between two vectors, and $\boldsymbol{G}_w \in \mathbb{R}^{6\times p}$ is the mapping between the propellers square angular rates and the control wrench. In particular we can partition $\boldsymbol{G}_w$ in two blocks, $\boldsymbol{G}_{w1} \in \mathbb{R}^{3\times p}$ and $\boldsymbol{G}_{w2} \in \mathbb{R}^{3\times p}$ such that $\boldsymbol{G}_w = [\boldsymbol{G}_{w1}^\top \ \boldsymbol{G}_{w2}^\top]^\top$. $\boldsymbol{G}_{w1}^\top$ and $\boldsymbol{G}_{w2}^\top$ map the propellers square angular rates into control force and moment, respectively. In this work, we consider both cases of 1) *under-actuated*, and 2) *fully-actuated* vehicles. For under-actuated vehicles, $0 < \text{rank}(\boldsymbol{G}_{w1}) < 3$ which means that the total thrust cannot change in all directions without reorienting the whole platform. This is the case of standard quadrotors where the propellers are all collinear and the thrust direction is fixed w.r.t. $\mathscr{F}_R$. On the contrary, for fully-actuated vehicles, $\text{rank}(\boldsymbol{G}_{w1}) = 3$ (it has to be that $p \geq 6$) which means that the total thrust can change in all directions. In both cases, $\boldsymbol{G}_{w2} = 3$, i.e., there is full control on the moment applied by the aerial platform.

A. Contact Modeling

We assume that the robot interacts with a *rigid* and *planar* environment.[2] In an industrial environment, such assumptions may occur while executing tasks such as polishing,

[1] The skew operator $\boldsymbol{S}(\star) : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ is defined such that for two generic vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$, $\boldsymbol{S}(\boldsymbol{x})\boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y}$

[2] The planarity assumption is adopted here for simplicity and could be replaced by a less stringent assumption of surface smoothness with small changes in the model and control law.

inspection or welding. Possible types of interaction from the end-effector constraints point of view are:

- *Fully constrained*: the end-effector position and orientation remain always constant w.r.t. the inertial frame;
- *Only the position is constrained*: the end-effector can freely rotate, but it cannot change its position. This is the case of a single contact point;
- *Only the normal translation is constrained*: the translation in the direction normal to the contact plane is constrained, while the end-effector is free to move in the perpendicular directions. The end-effector can also rotate;
- *Rotations and the normal translation are constrained*: the end-effector position is constrained as in the previous case. However, the end-effector can rotate about the normal direction only;

We model those interactions by a set of *holonomic constraints* which describe the limitations of the end-effector motion [16], [17]. Often it is easier to express those constraints w.r.t. local reference frame $\mathscr{F}_E$. Differentiating such constraints w.r.t. time one obtains:

$$\boldsymbol{S}_c \begin{bmatrix} {}^E\boldsymbol{v}_E^\top & {}^E\boldsymbol{\omega}_E^\top \end{bmatrix}^\top = \boldsymbol{0},$$

where ${}^E\boldsymbol{v}_E$ and ${}^E\boldsymbol{\omega}_E$ are the end-effector linear and angular velocities w.r.t. $\mathscr{F}_E$, while $\boldsymbol{S}_c \in \mathbb{R}^{n_c \times 6}$ is a selector of the constrained directions of motion. $n_c \in \mathbb{N}_{>0}$ represents the number of motion constraints applied to the end-effector. Note that in the local reference frame the selector matrix $\boldsymbol{S}_c$ usually remains constant during each interaction task. Using the kinematic relation, the contact constraints can be expressed as a function of the robot velocities $\boldsymbol{v}$:

$$\boldsymbol{S}_c \bar{\boldsymbol{R}} \begin{bmatrix} \boldsymbol{v}_E^\top & \boldsymbol{\omega}_E^\top \end{bmatrix}^\top = \boldsymbol{J}_E^c \boldsymbol{v} = \boldsymbol{0}, \tag{3}$$

where[3] $\boldsymbol{J}_E^c \coloneqq \boldsymbol{S}_c \bar{\boldsymbol{R}} \boldsymbol{J}_E \in \mathbb{R}^{n_c \times 6+n}$, $\bar{\boldsymbol{R}} = \text{blkdiag}(\boldsymbol{R}_E, \boldsymbol{R}_E)$ with $\boldsymbol{R}_E \in SO(3)$ being the rotation matrix describing the orientation of $\mathscr{F}_E$ w.r.t. $\mathscr{F}_W$. The equations complementary to (3) represent instead the directions of motion of the end-effector that remain free to move, and can be written as:

$$\boldsymbol{S}_f \bar{\boldsymbol{R}} \begin{bmatrix} \boldsymbol{v}_E^\top & \boldsymbol{\omega}_E^\top \end{bmatrix}^\top = \boldsymbol{J}_E^f \boldsymbol{v} =: \boldsymbol{v}_f, \tag{4}$$

where $\boldsymbol{J}_E^f \coloneqq \boldsymbol{S}_f \bar{\boldsymbol{R}} \boldsymbol{J}_E \in \mathbb{R}^{6-n_c \times 6+n}$ with $\boldsymbol{S}_f \in \mathbb{R}^{6-n_c \times 6}$ the selector matrix complementary to $\boldsymbol{S}_c$.

Rewriting the system dynamics (1) including (2) and taking into account the contact model (3)-(4) gives:

$$\boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{c} + \boldsymbol{g} = \boldsymbol{G}\boldsymbol{u} + \boldsymbol{J}_E^{c\top} \boldsymbol{f}_c + \boldsymbol{J}_E^{f\top} \boldsymbol{f}_f \tag{5a}$$

$$\boldsymbol{J}_E^c \dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_E^c \boldsymbol{v} = \boldsymbol{0}, \tag{5b}$$

where $\boldsymbol{G} = \text{blkdiag}(\boldsymbol{G}_w, \boldsymbol{I}_3)$, $\boldsymbol{u} = [\boldsymbol{\omega}_P^{2\top} \ \boldsymbol{\tau}_A^\top]^\top$, $\boldsymbol{f}_c \in \mathbb{R}^{n_c}$ are the contact forces and/or moments, while $\boldsymbol{f}_f \in \mathbb{R}^{6-n_c}$ represent forces and moments that may arise in the unconstrained directions of motion, e.g., viscous friction during motion. Equation (5b) is the time differentiation of (3) and highlights the constraints on the accelerations $\dot{\boldsymbol{v}}$.

[3]From now on, with the aim of compactness, we avoid to report the state dependence of some quantities as the Jacobian and mass matrix, and so on.

III. Control Design

Let us denote the vector of outputs of interest (called *tasks*) with $\boldsymbol{y} = [y_1 \ \dots \ y_m]^\top \in \mathbb{R}^m$, where $m \in \mathbb{N}_{>0}$. The control method is composed of an *outer loop* and an *inner loop*. The outer-loop assumes that a certain time-derivative for each task, defined by the symbol[4] $\boldsymbol{a} = [y_1^{(r_1)} \ \dots \ y_m^{(r_m)}]^\top$, is directly controllable by a virtual input $\boldsymbol{a}^\star$, i.e.:

$$\boldsymbol{a} = \boldsymbol{a}^\star. \tag{6}$$

Based on this assumption any kind of stabilizing controller (PID, sliding mode, robust control, etc) can be applied by the outer loop designing $\boldsymbol{a}^\star$ such that to steer $\boldsymbol{y}$ along a sufficiently smooth desired trajectory $\boldsymbol{y}^d(t)$. In what follows, we apply to the outer loop a PID controller, whose effectiveness in stabilizing the desired output has been verified experimentally. The role of the inner-loop is instead to compute the real system inputs in order to verify as much as possible (6) via the resolution of a constrained optimization problem [12]:

$$\underset{\boldsymbol{u}}{\text{minimize}} \ (\boldsymbol{a} - \boldsymbol{a}^\star)^\top \boldsymbol{W}_a (\boldsymbol{a} - \boldsymbol{a}^\star), \tag{7}$$

subjected to several constrains, e.g.: 1) input and state boundaries, 2) contact stability constraints, 3) system dynamics, etc. The (semi) positive and diagonal weight matrix $\boldsymbol{W}_a \in \mathbb{R}^{m \times m}$ allows to defines *soft* priorities among all the tasks.

As demonstrated in other robotic fields,[5] this technique can handle complex system dynamics and a large number of tasks. An additional advantage is the flexibility in adding and removing both tasks and constraints. As an example, if *strict* tasks priorities are required, the optimization problem can be easily modified by transforming some of the elements of the cost function into equality constraints.

The input-output asymptotic stability is obtained if it exists $\boldsymbol{u}^\star$ such that $\boldsymbol{a} = \boldsymbol{a}^\star$, i.e., if the optimal control input is a feedback linearizing control. This implies that a careful choice of $\boldsymbol{a}$ is required. However, the definition of $\boldsymbol{a}$ may not be easy in case the robot has to perform a complex operation in a real environment. Furthermore, the presence of several constraints may prevent to find a solution that guarantees (6) for all the components of $\boldsymbol{a}$. In this case, the solution will privilege the high priority tasks while keeping bounded the the error on lower priority tasks. The priority among tasks, and consequently the behavior of the robot, can be modified by properly choosing $\boldsymbol{W}_a$.

In the following, considering a contact-based application requiring a hybrid position/force control of the end-effector, we define the appropriate set of tasks and the implementation of the inner- and outer-loop controllers.

1) Control Task Definition: For our case study, we define the output as $\boldsymbol{y} = (\boldsymbol{p}_R, \boldsymbol{R}_R, \boldsymbol{q}_A, \boldsymbol{p}_f, \boldsymbol{f}_c, \boldsymbol{r}_a)$ and the corresponding time-derivatives to be assigned by the outer loop as $\boldsymbol{a} = \begin{bmatrix} \dot{\boldsymbol{v}}_R^\top & \dot{\boldsymbol{\omega}}_R^\top & \dot{\boldsymbol{v}}_A^\top & \dot{\boldsymbol{v}}_f^\top & \boldsymbol{f}_c^\top & \boldsymbol{r}_a^\top \end{bmatrix}^\top$. The tasks $\boldsymbol{p}_R$, $\boldsymbol{R}_R$ and $\boldsymbol{q}_A$ represent the full aerial manipulator configuration, and can

[4]For a given variable $x \in \mathbb{R}$, $x^{(r)}$ indicates the derivative of order r of x.

[5]As, e.g., in humanoid control.

be used to cover several real world objectives, from simple changes of the overall position, to more complex coordinated maneuvers in cluttered environment. The task $\boldsymbol{p}_f \in \mathbb{R}^{6-n_c}$ and its time-derivative $\boldsymbol{v}_f$ represent the end-effector positions and velocities in the unconstrained directions of motion. The role of $\boldsymbol{v}_f$ and $\boldsymbol{f}_c$ is to specify the hybrid force-motion behavior of the aerial manipulator perform, when in contact. Finally, the term $\boldsymbol{r}_a$ is a propeller regularization task which is defined as a function of the propeller forces and such that $\boldsymbol{r}_a = \boldsymbol{0}$ when all the forces are the same. The regularization term is designed as follows:

$$\boldsymbol{r}_a = \boldsymbol{D}_{\boldsymbol{r}_a} \boldsymbol{\omega}_P^2,$$

where $\boldsymbol{D}_{\boldsymbol{r}_a} \in \mathbb{R}^{p-1 \times p}$ is a matrix whose elements are 1 along the main diagonal and -1 right above the main diagonal, and all other elements are equal to zero. The goal of this task is to balance the propeller forces in order to avoid energetically unfavorable solutions where some propellers deliver almost zero thrust and others are close to saturation.[6] Notice that the virtual input associated to the tasks $\boldsymbol{f}_c$ and $\boldsymbol{r}_a$ are the variable themselves (zero-order time-derivative) being such tasks algebraic functions of the input.

2) Inner-Loop (QP-based Optimization): Let us consider the extended input $\boldsymbol{u}' = [\boldsymbol{u}^\top \ \boldsymbol{f}_c^\top]^\top \in \mathbb{R}^{n+p+n_c}$ in which the contact wrench is added to the real control inputs $\boldsymbol{u}$. In this way, inverting the dynamics (5a) and the time derivative of (4) one can express $\boldsymbol{a}$ as a linear function of $\boldsymbol{u}'$:

$$\boldsymbol{a} = \boldsymbol{H}(\boldsymbol{q})\boldsymbol{u}' + \boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}), \tag{8}$$

where $\boldsymbol{H}(\boldsymbol{q}) \in \mathbb{R}^{m \times (n+p+n_c)}$ and $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^m$ contain all the terms that do not depend on $\boldsymbol{u}'$. More specifically, the matrix $\boldsymbol{H}(\boldsymbol{q})$ and the bias vector $\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v})$ are given by:

$$\boldsymbol{H}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{M}^{-1}\boldsymbol{G} & \boldsymbol{M}^{-1}\boldsymbol{J}_E^{c\top} \\ \boldsymbol{J}_E^f \boldsymbol{M}^{-1}\boldsymbol{G} & \boldsymbol{J}_E^f \boldsymbol{M}^{-1}\boldsymbol{J}_E^{c\top} \\ \boldsymbol{0}_{(n_c \times p)} & \boldsymbol{1}_{(n_c \times n_c)} \\ \boldsymbol{D}_{\boldsymbol{r}_a} & \boldsymbol{0}_{(p-1 \times n_c)} \end{bmatrix},$$

$$\boldsymbol{h}(\boldsymbol{q}, \boldsymbol{v}) = \begin{bmatrix} \boldsymbol{M}^{-1}(\boldsymbol{J}_E^{f\top} \boldsymbol{f}_f - \boldsymbol{c} - \boldsymbol{g}) \\ \boldsymbol{J}_E^f \boldsymbol{M}^{-1}(\boldsymbol{J}_E^{f\top} \boldsymbol{f}_f - \boldsymbol{c} - \boldsymbol{g}) + \dot{\boldsymbol{J}}_E^f \boldsymbol{v} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}.$$

To ensure the satisfaction of the contact constraint the equation (5b) is added to the optimization problem (7) as a constraint. Such input extension avoids the explicit inversion of $\boldsymbol{J}_E^c$ and the related singularity issues that may arise in some configuration.

For the considered interaction-based scenario, the inner-loop control problem (7) may be formulated as

$$\underset{\boldsymbol{u}'}{\text{minimize}} \ \ (\boldsymbol{a} - \boldsymbol{a}^\star)^\top \boldsymbol{W}_a (\boldsymbol{a} - \boldsymbol{a}^\star) \tag{10a}$$

$$\text{subject to:} \quad \boldsymbol{u}_l \leq \boldsymbol{u} \leq \boldsymbol{u}_u \tag{10b}$$

$$\boldsymbol{C}\boldsymbol{f}_c \leq \boldsymbol{b} \tag{10c}$$

$$\boldsymbol{M}\dot{\boldsymbol{v}} + \boldsymbol{c} + \boldsymbol{g} = \boldsymbol{G}\boldsymbol{u} + \boldsymbol{J}_E^{c\top} \boldsymbol{f}_c + \boldsymbol{J}_E^{f\top} \boldsymbol{f}_f \tag{10d}$$

$$\boldsymbol{J}_E^c \dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_E^c \boldsymbol{v} = \boldsymbol{0} \tag{10e}$$

The constraint (10b) takes into account the bounds on the control inputs, such as saturations in the joint torques and propeller velocities, while (10c) ensures the respect of contact stability conditions such as friction cone (approximated with linear inequalities) and positivity of the normal force. Constraints (10e)-(10d) correspond to the dynamics of the aerial manipulator when in contact with the environment.

Observing (10), it is easy to verify that is an instance of a *Quadratic Programming* (QP) problem. In fact, the cost function and constraints are a quadratic and linear functions of the optimization variable $\boldsymbol{u}'$, respectively. The QP problem allows for a fast and efficient solution that provides online the control inputs $\boldsymbol{u}^\star$, such that $\boldsymbol{u}'^\star = [\boldsymbol{u}^{\star\top} \ \boldsymbol{f}_c^{\star\top}]^\top$ is solution of (10).

3) Outer-Loop (Desired Dynamics): Given a desired output trajectory $\boldsymbol{y}^d(t)$ the task virtual inputs $\boldsymbol{a}^\star$ are chosen as follows:

$$\boldsymbol{a}^\star = \begin{bmatrix} \dot{\boldsymbol{v}}_R^d - \boldsymbol{K}_{DR}(\boldsymbol{v}_R - \boldsymbol{v}_R^d) - \boldsymbol{K}_{PR}(\boldsymbol{p}_R - \boldsymbol{p}_R^d) \\ \dot{\boldsymbol{\omega}}_R^d - \boldsymbol{K}_{DR\omega}(\boldsymbol{\omega}_R - \boldsymbol{\omega}_R^d) - \boldsymbol{K}_{PR\omega}\boldsymbol{e}_{R\omega} \\ \dot{\boldsymbol{v}}_A^d - \boldsymbol{K}_{DA}(\boldsymbol{v}_A - \boldsymbol{v}_A^d) - \boldsymbol{K}_{PA}(\boldsymbol{q}_A - \boldsymbol{q}_A^d) \\ \dot{\boldsymbol{v}}_f^d - \boldsymbol{K}_{DF}(\boldsymbol{v}_f - \boldsymbol{v}_f^d) - \boldsymbol{K}_{PF}(\boldsymbol{p}_f - \boldsymbol{p}_f^d) \\ \boldsymbol{f}_c^d - \boldsymbol{K}_{PC}(\boldsymbol{f}_c^m - \boldsymbol{f}_c^d) - \boldsymbol{K}_{IC}\int_0^t (\boldsymbol{f}_c^m - \boldsymbol{f}_c^d)dt \\ \boldsymbol{0} \end{bmatrix} \tag{11}$$

where the $\boldsymbol{K}_*$ are symmetric and positive definite matrices and the rotation error $\boldsymbol{e}_{R\omega} \in \mathbb{R}^3$ is given by $\boldsymbol{e}_{R\omega} = \frac{1}{2}[\boldsymbol{R}_R^\top \boldsymbol{R}_R^d - \boldsymbol{R}_R^{d\top} \boldsymbol{R}_R]_\vee$. Direct feedback from FT sensors measurements is used in the force control task for computing the force error $\boldsymbol{f}_c^m - \boldsymbol{f}_c^d$, with $\boldsymbol{f}_c^m$ being the measured contact force.

IV. Results

A. Experimental Setup

The control framework presented in Sec. III is tested with the *Open Tilted Hexarotor* (OTHex) [18]. The OTHex is a custom-made aerial vehicle composed of six coplanar-center propellers. The tilted arrangement of the propellers allows the multi-directional thrust property and therefore guarantees the local full actuation of its dynamics. However, despite the local full actuation, the propeller saturation still limits substantially its ability to exert a force sideways, as it will be clearly shown in some phases of the experiments, where tilting of the whole OTHex is also requested by the controller in order to get the needed pushing force.

The OTHex has been equipped with a three degrees of freedom serial manipulator as in Fig. 1. The manipulator's end-effector is composed of a pointed tool rigidly attached to a 6-axis force/torque sensor. The manipulator is controlled with a velocity control loop that commands desired motors velocities to three Dynamixel motors. The reference velocities for the velocity control loop are computed by numerically integrating the commanded joints acceleration

[6]This task is particularly important in the case that the number of propellers and their arrangement is redundant for the execution of the remaining tasks. as, e.g., in the case of an underactuated floating base with six or more propellers all pointing in the same direction.

Weights for the selected tasks during experiments						
	p_R	R_R	q_A	p_f	f_c	r_a
Push with dist.	1	1	0.1	0	2	1e-5
Push and slide	0.1	1.5	0.025	2.5	1	1e-5
Push away	2	2	0.05	0	0	1e-5
Simulation	1	0.01	0.01	0	1	1e-5

TABLE I: Weights of the selected tasks for the QP cost function, during the three experiments and in simulation.

$\dot{\boldsymbol{v}}_A^\star$, that are obtained inverting the system's dynamics (5) when $\boldsymbol{u}' = \boldsymbol{u}'^\star$.

All experiments are performed in an indoor arena using a Motion Capture (MoCap) system. The Control algorithm is implemented in Matlab-Simulink and runs on an external PC at a frequency of 250 $[Hz]$. The inner-loop optimization problem in (10) is resolved run-time by means of qpOASES solver [19]. As it often occurs in real world applications, the hard QP equality constraints are softened by moving (10e) into the set of virtual inputs (11). This modification helps to reduce the discontinuities in the control input when the contact constraints are activated/deactivated. Contact constraints are then heavily weighted by properly designing matrix $\boldsymbol{W}_a$, to enforce the achievement of the corresponding task in (11). We performed three different experiments and a simulation, that will be all detailed in the next paragraphs. The QP weights used for the different tasks during each experiment are listed in Table I. The videos of the experiments are available in the attached multimedia material.

B. Pushing with Disturbances

During this experiment, the robot's end-effector gets in contact with a rigid surface. The robot is then required to push against the surface with a normal force of 5 $[N]$. After few seconds, a virtual force disturbance of 3.5 $[N]$ is applied at the OTHex base link. The disturbance persists for 5 seconds, and it is removed after. The purpose of the experiment is to understand the benefit of adding direct force feedback from the force/torque sensor when tracking a reference force in presence of external disturbances. To this purpose, we compared two different scenario: in the *'no FT'* case, the reference output force from (11) is computed as $\boldsymbol{f}_c = \boldsymbol{f}_c^d$, thus not adding any feedback from the measured contact forces. In the *'with FT'* case, the reference output force is computed including feedback terms as in (11).

Figure 2 describes the behavior of the normal force during the push with disturbance experiment for the two different cases. In particular, the plot shows the measured ($f_{c_z}^m$) and commanded ($f_{c_z}^\star$) normal forces during the interaction task. The dashed black line is the desired normal force. The red region denotes the period of time during which the end-effector is in contact with the surface, and the blue region is the time period when the disturbance is applied.

If feedback from force/torque sensors is used in the control law (top plot of Fig. 2), the commanded vertical force (blue line) increases to $6N$ when the disturbance is applied. This due to the presence of force feedback, that attempts at compensating for the exernal disturbance. In fact, as a

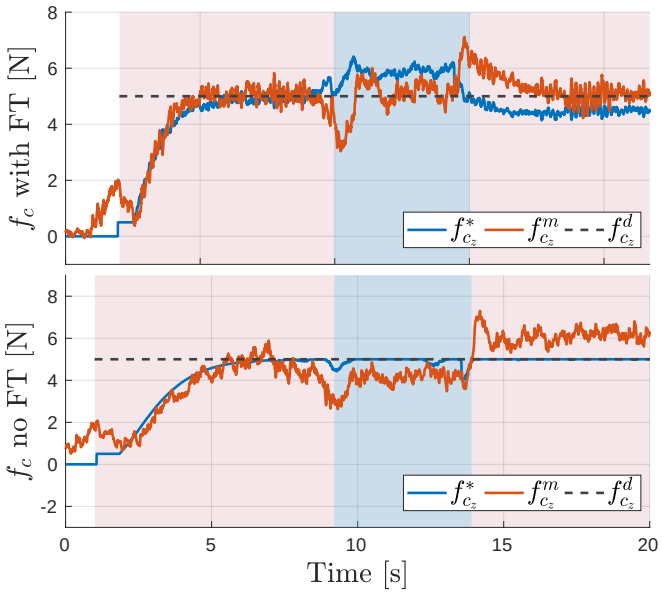

Fig. 2: Normal force at the end-effector when the robot is in contact. With FT sensors feedback (top plot), the measured force remains close to the desired value. With no feedback (bottom plot), the force drifts of ± 1 $[N]$.

consequence of the new commanded force, the measured force $f_{c_z}^m$ (red line) remains close to $5N$ after a short transient phase. When no force feedback is present instead, as in the bottom plot of Fig. 2, the commanded force does not change magnitude and the error between measured and desired force increases up to 1 $[N]$. We recall that in this second case the force/torque sensor information is only used as ground-truth, but is not actively employed in the control algorithm.

The top plot of Fig. 3 shows the the propeller commanded angular rates $\boldsymbol{\omega}_P^\star$ during the experiment with FT sensor feedback. When the external force is applied, three propellers reach saturation, which is represented in the figure by the dotted horizontal lines. Therefore, the solution $\boldsymbol{a} = \boldsymbol{a}^\star$ cannot be achieved anymore, and the QP penalizes tasks with lower priority, such as the OTHex orientation, in the attempt of maintaining a small error on the higher priority task. This effect is visible in the bottom plot of Fig. 3, where the error along the pitch angle of the OTHex increases up to 7 $[deg]$ during the saturation phase.

C. Push and Slide

In this experiment, the robot pushes against the surface with a normal force of 2.5 $[N]$, while sliding along the surface following a reference end-effector trajectory. We made use of FT sensors feedback for improving the tracking of the normal force, and the FT sensor information is also used for compensating the viscous friction forces acting on the surface while sliding.

The top part of Figure Fig. 4 compares the measured and the desired end-effector trajectory along the contact surface. Despite some noise due to vibrations and compliance of the arm, the tracking error always remains small (< 1 $[cm]$

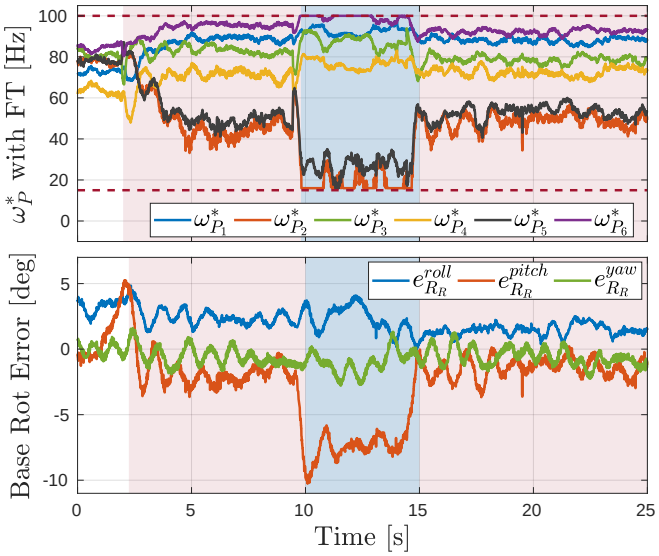

Fig. 3: Propellers commanded angular rates and base rotation error during pushing with disturbance. when the disturbance is applied, the propellers saturate. The robot error along the pitch direction increases to keep small the error of higher priority tasks, such as the contact force one.

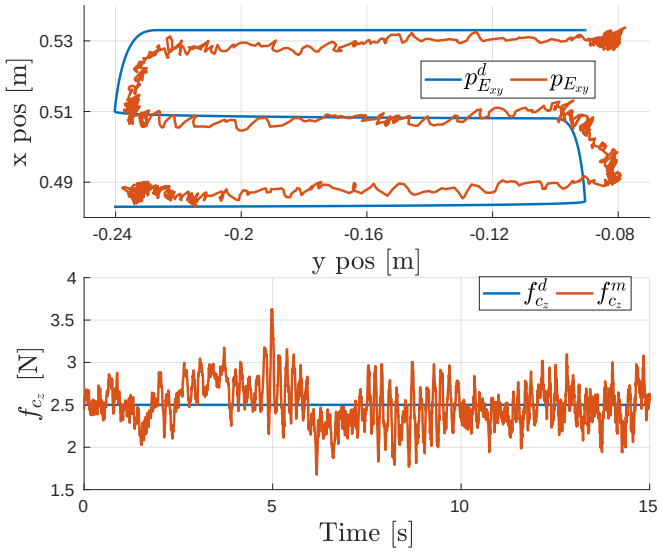

Fig. 4: End-effector measured and reference trajectories on the contact surface (top plot) and measured and reference normal force (bottom plot). The robot achieves good tracking performances while keep pushing with the required force.

along the y direction and $< 0.5[cm]$ along the x direction). The bottom part of figure Fig. 4 shows the desired and measured normal contact force during sliding. The robot is capable of pushing with the required normal force during the whole task. As it is possible to see in Table I, during this experiment the number of active tasks is greater than the number of independent inputs. As for the previous experiment, tasks with lower priority are penalized in order to achieve tasks with higher priority. In fact, Fig. 5 shows that the joints position error increases during the sliding task. The reference joints position is a fixed posture $\boldsymbol{q}_A^d$ which is not related to the end-effector references.

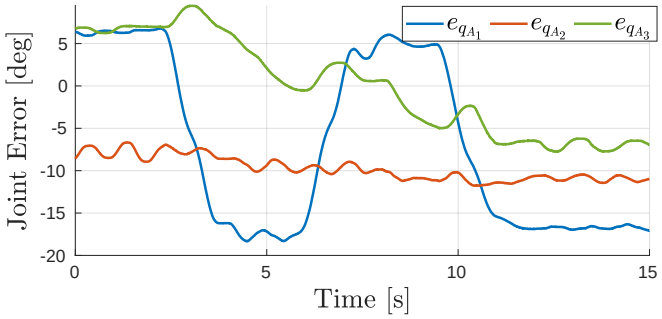

Fig. 5: Joints position error during push and slide task. The low-priority task is penalized to allow the achievement of higher priority tasks.

D. Pushing Away

In this experiment the robot performs an aggressive maneuver while hovering. In particular, the OTHex is required to move horizontally of 35 $[cm]$ in 0.5 $[s]$. The robot performs the fast movement in two scenarios: while hovering with no contact with the environment, and while hovering in contact with a planar surface. In the second case, the contact is also exploited to perform the task. Fig. 11 describes the experiment in the scenario when the robot is in contact.

Fig. 6 points out how the contact is exploited for performing the fast motion: a high normal force is requested by the controller in order to "push away" the robot from the contact. The peak force is achieved by means of the manipulator: in particular, by commanding a high acceleration on the second arm joint (red line), which is absent in the scenario in which the robot is not in contact (blue line). The blue area in the plot indicates the time at which the robot performs the fast movement. At half of the time, the contact constraint equations (10e)-(10c) are removed from the QP to allow the robot to move forward. Fig. 7 is the position tracking error along the direction of the fast movement. Exploiting the contact allowed to reduce by $6[cm]$ the peak error and the robot stabilizes on the desired position in a shorter time. Furthermore, Fig. 8 shows that exploiting the contact also helped in reducing the amount of time the propellers angular rates remains saturated.

E. Force Control with Underactuation

To verify if the proposed control framework can be effectively applied also for controlling underactuated aerial systems, we performed a force control task in simulation with an underactuated hexarotor equipped with a three degrees of freedom manipulator. We control the simulated robot with the same Matlab-Simulink environment used for the experiments, while dynamics integration is performed by Gazebo simulator [20]. A virtual force/torque sensor is simulated inside Gazebo and mounted on the robot's end-effector.

The task is to achieve a desired normal force of 5 $[N]$

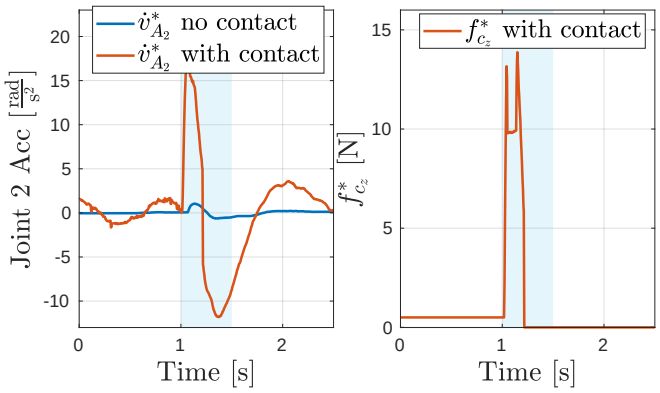

Fig. 6: Joint 2 acceleration (left plot) and normal force (right plot) required by the controller during push away task. The robot exploits the contact to achieve a fast base acceleration.

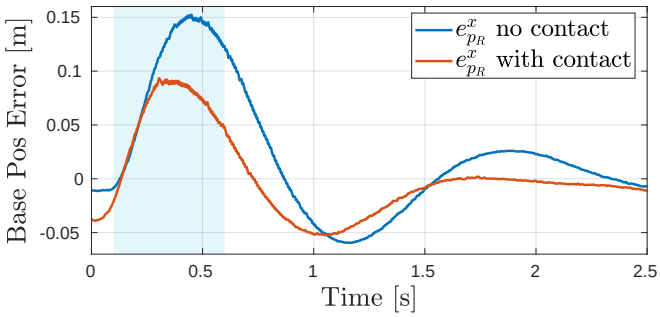

Fig. 7: Position error along the direction of the fast motion during push away task. Exploiting the contact reduces the peak error and allows faster convergence.

while pushing against a rigid surface. The top plot in Fig. 9 shows that the robot is capable of achieving the required force despite the underactuation. The bottom plot depicts the propellers angular rates. While achieving the force task, propellers still remain far from saturation. Fig. 10 depicts instead the error on the base rotation task. Differently from the fully actuated case of Fig. 3, where the rotation error only increased in presence of external disturbances and propellers saturation, because of underactuation the robot anyways needs to tilt of 10 $[deg]$ along the pitch direction in order to be able to execute the force task.

V. Conclusions

In this work we considered the challenging problem of precise position and force control for an aerial manipulator. The proposed method takes inspiration from whole-body control methods applied to humanoid robots. In particular, our control method is based on a multi-task optimization problem, solved with a QP method. This allows considering different control objectives (e.g., interaction force, position of the end-effector, full pose of the robot, etc.) and hard constraints that should be respected to ensure the stability of the system and the feasibility of the problem solution. The precision of the method during interaction is enhanced by a direct force-feedback exploiting a FT sensor integrated at the end-effector.

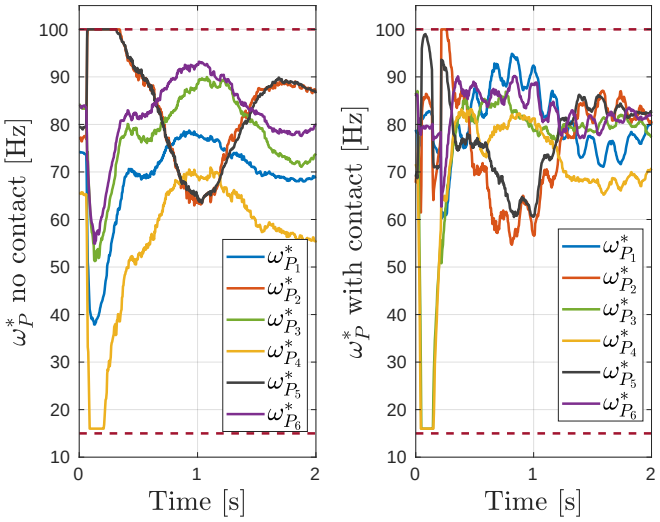

Fig. 8: Propellers angular rates during push away task. Exploiting the contact helped in limiting propellers saturation.

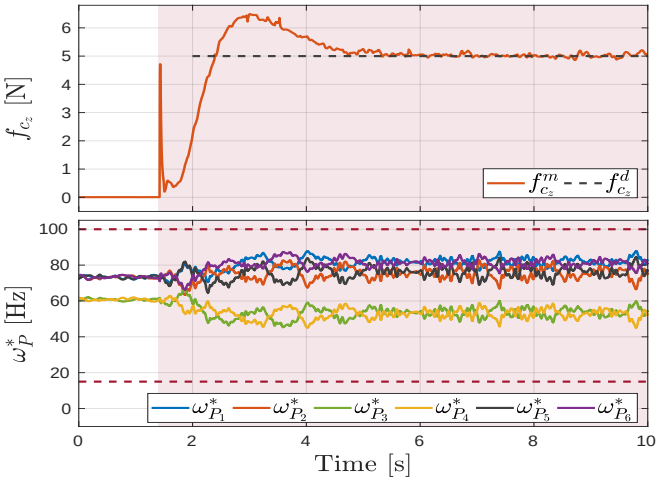

Fig. 9: Normal force during contact (top plot) and propellers velocities (bottom plot) in simulation. The robot can successfully achieve force control tasks despite the underactuation.

We performed experimental and simulation test based on fully- and under-actuated aerial platform respectively, for different tasks: 1) push with disturbances, 2) push and slide, and 3) push away, The corresponding result show the great flexibility of the method and the improvement in the force-tracking thanks to the explicit force-feedback.

In the experiments, only the point contact scenario has been considered. In future work we shall also consider other kind of interactions, and consequently design control algorithms that will include full contact wrench feedback. Furthermore, other challenging tasks will be subject of study, such as manipulation and grasping, and multi-contact interaction. Further experiments will also be carried on with underactuated platforms.

References

[1] M. Tognon, H. A. Tello Chávez, E. Gasparin, Q. Sablé, D. Bicego, A. Mallet, M. Lany, G. Santi, B. Revaz, J. Cortés, and A. Franchi,

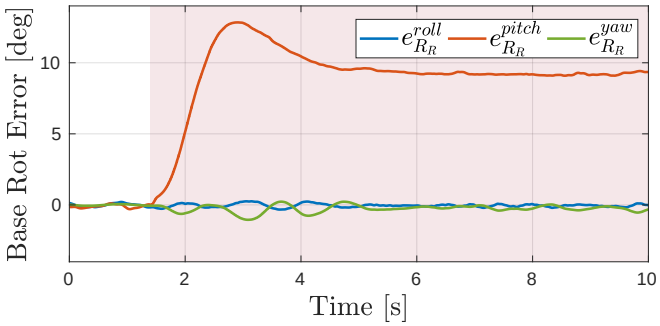

Fig. 10: Base rotation error while performing force control task in simulation. Due to the system's underactuation, the robot needs to tilt in order to achieve the required contact force.

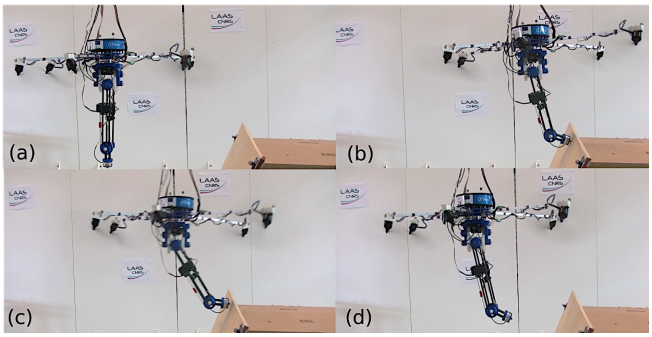

Fig. 11: Pushing away with contact. a) approaching, b) making contact, c) pushing, d) resting

"A truly redundant aerial manipulator system with application to push-and-slide inspection in industrial plants," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1846–1851, 2019.

[2] F. Ruggiero, V. Lippiello, and A. Ollero, "Aerial manipulation: A literature review," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1957–1964, July 2018.

[3] H. Yang and D. J. Lee, "Dynamics and control of quadrotor with robotic manipulator," in *2014 IEEE Int. Conf. on Robotics and Automation*, Hong Kong, China, May 2014, pp. 5544–5549.

[4] M. Tognon, B. Yüksel, G. Buondonno, and A. Franchi, "Dynamic decentralized control for protocentric aerial manipulators," in *2017 IEEE Int. Conf. on Robotics and Automation*, Singapore, May 2017, pp. 6375–6380.

[5] S. Kim, S. Choi, and H. J. Kim, "Aerial manipulation using a quadrotor with a two DOF robotic arm," in *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013, pp. 4990–4995.

[6] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Behavioral control of unmanned aerial vehicle manipulator systems," *Autonomous Robots*, vol. 41, no. 5, pp. 1203–1220, 2017.

[7] A. Santamaria-Navarro, V. Lippiello, and J. Andrade-Cetto, "Task priority control for aerial manipulation," in *2014 IEEE Int. Symp. on Safety, Security and Rescue Robotics*, Oct 2014, pp. 1–6.

[8] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, D. Bicego, and A. Franchi, "6D interaction control with aerial robots: The flying end-effector paradigm," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1045–1062, 2019.

[9] A. Suarez, G. Heredia, and A. Ollero, "Physical-virtual impedance control in ultralightweight and compliant dual-arm aerial manipulators," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2553–2560, July 2018.

[10] R. Rashad, F. Califano, and S. Stramigioli, "Port-hamiltonian passivity-based control on se (3) of a fully actuated uav for aerial physical interaction near-hovering," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4378–4385, 2019.

[11] G. Antonelli, E. Cataldi, G. Muscio, M. Trujillo, Y. Rodriguez, F. Pierri, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Impedance control of an aerial-manipulator: Preliminary results," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Daejeon, South Korea, 2016, pp. 3848–3853.

[12] M. Charbonneau, V. Modugno, F. Nori, G. Oriolo, D. Pucci, and S. Ivaldi, "Learning robust task priorities of QP-based whole-body torque-controllers," in *2018 IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, November 2018.

[13] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *2009 International Conference on Advanced Robotics*, June 2009, pp. 1–6.

[14] R. Featherstone, *Rigid Body Dynamics Algorithms*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[15] J. E. Marsden and T. S. Ratiu, *Introduction to Mechanics and Symmetry: A Basic Exposition of Classical Mechanical Systems*. Springer Publishing Company, Incorporated, 2010.

[16] V. Ortenzi, R. Stolkin, J. Kuo, and M. Mistry, "Hybrid motion/force control: a review," *Advanced Robotics*, vol. 31, no. 19-20, pp. 1102–1113, 2017.

[17] C. C. de Wit, B. Siciliano, and G. Bastin, *Motion and force control*. London: Springer, 1996, pp. 141–175.

[18] N. Staub, D. Bicego, Q. Sablé, V. Arellano-Quintana, S. Mishra, and A. Franchi, "Towards a flying assistant paradigm: the OTHex," in *2018 IEEE Int. Conf. on Robotics and Automation*, Brisbane, Australia, May 2018, pp. 6997–7002.

[19] H. J. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2149 – 2154, 2004.