Control-Aware Motion Planning for Task-Constrained Aerial Manipulation

Marco Tognon, Elisabetta Cataldi, Hermes Amadeus A Tello Chavez, Gianluca Antonelli, Juan Cortés, Antonio Franchi

Control-Aware Motion Planning for Task-Constrained Aerial Manipulation

M. Tognon[1], E. Cataldi[2], H. A. Tello Chavez[1], G. Antonelli[2], J. Cortés[1], A. Franchi[1]

***Abstract*— This paper presents a new method to address the problem of task-constrained motion planning for aerial manipulators. We propose a control-aware planner based on the paradigm of tight coupling between planning and control. Such paradigm is especially useful in aerial manipulation, where the separation between planning and control is not advisable. The proposed sampling based motion planner uses a controller composed of a second-order inverse kinematics algorithm and a dynamic tracker, as a local planner, thus allowing a more natural consideration of the closed-loop system dynamics. For task constrained motions, this method lets to i) sample directly in the reduced and more relevant task space, ii) predict the behavior of the controller avoiding motions that bring to singularities or large tracking errors, and iii) guarantee the correct execution of the maneuver. The method is tested in simulation for a multidirectional-thrust vehicle endowed with a two-DoF manipulator. The proposed approach is very general, and could be applied to ground and underwater robotic systems to perform manipulation or inspection tasks.**

I. INTRODUCTION

The interest in Unmanned Aerial Vehicles (UAVs) has increased exponentially in the last decade. This comes from their large applicability in diverse real-world scenarios. A very recent and promising application field is aerial manipulation. In this case, UAVs are not simply used as remote sensors in a free-flight configuration. Instead, the robot needs to physically interact with the environment, exchanging forces. An interesting application is the inspection by contact of industrial installations. In this context, taking measurements requires physical contact between the sensor and the inspected part. For example, in the context of the EU project Aeroarms, one of the goals is to develop an aerial manipulator able to take ultrasonic measurements of a metallic pipe to detect flaws [1].

Compared to ground manipulation, physical interaction using aerial vehicles endowed with rigid tools or robotic arms is a much more challenging problem, mainly due to the use of a floating base. The aerial platform, usually thrusted by one or several propellers, cannot instantaneously react to interaction forces between the robotic arm and the environment. In addition, aerodynamic effects and model uncertainties yield to inaccurate positioning, thus increasing the challenging nature of the problem. Several works in the literature presented various types of dynamic controllers for aerial manipulators to stabilize the system and track a desired trajectory of the degrees of freedom. They range from completely decoupled approaches [2], [3], to model-based approaches [1], [4] and to more recent differential-flatness-based method [5].

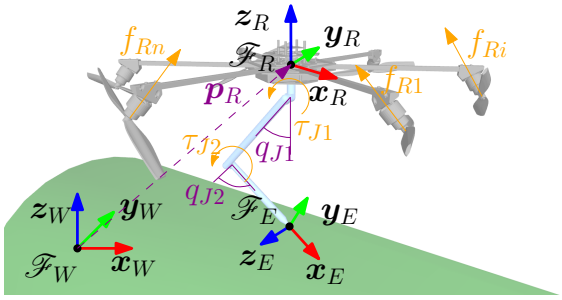


Fig. 1: Schematic representation of an aerial manipulator inspecting a pipe (green surface) by physical contact. The aerial vehicle is an hexarotor with tilted propellers, endowed with a two-link arm.

Some relevant applications impose motion constraints derived from the task. For example, one could require the end-effector to follow a given trajectory or to move while keeping contact with a surface that has to be inspected. Figure 1 shows an example where an aerial manipulator has to inspect the surface of a pipe. To accomplish this type of task, one of the possible methods is based on inverse kinematics control [6]. If the system is over-actuated with respect to (w.r.t.) the desired task, one can also exploit the redundancy to locally optimize some behaviors (e.g., obstacle avoidance, minimum energy consumption, etc.), using null-space-based behavioral control (NSB) [7]. However, due to the local nature of those approaches, the system can get trapped in some local minima. This problem often implies the failure of the sought task.

Techniques using (global) task-constrained motion planning methods have been proposed in order to overcome limitations of purely reactive (control-based) methods. Many works on *task-constrained motion planning*, as [8]–[10], use a projection strategy to sample configurations that respect the task constraints. However they consider the system at a pure kinematic level, i.e., they assume that the robot can track any velocity reference, even if discontinuous. By doing so, they cannot guarantee that the robot will accurately execute the planned trajectory. This is an important issue for aerial manipulators that can be easily destabilized by large tracking errors when dynamic effects are not properly considered.

A dynamic model of the system can be considered at the planning level using *kinodynamic motion planning* ap-

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France, mtognon@laas.fr, htelloch@laas.fr, jcortes@laas.fr antonio.franchi@laas.fr

[2]University of Cassino and Southern Lazio, 03043 Cassino, Italy, gianluca.antonelli@unicas.it, e.cataldi@unicas.it

This work has been funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 644271 AEROARMS.

proaches, as in [11]. However, this significantly increases the complexity of the problem, which is in practice tractable only for simple systems. Aerial manipulators are in general characterized by a high number of degrees of freedom, and thus, the dimension of the state space is too high for the application of current, general purpose, kinodynamic motion planners.

Interesting methods have been proposed to circumvent the complexity of kinodynamic approaches for task-constrained motion planing problems [12], [13]. However, even if the planned trajectory is feasible w.r.t. the dynamics of the system, one cannot guarantee that a given controller will be able to accurately track it during execution. This strongly depends on the employed controller. For example, robots controlled kinematically might show big tracking errors for certain motions. In those cases, when the available controller does not allow to precisely track the planned motion, the separation between planning and control is not suitable. This is the case for many robotic fields, as aerial manipulation or humanoid robotics, where the complexity of the system does not always allow to obtain very precise controllers.

In this paper, we propose an approach to reinforce the connection between motion planning and control in the context of aerial manipulation. The underlying idea is conceptually very simple. It consists in using the controller as a local method to connect neighboring states within a (global) motion planning algorithm. More precisely, our method is based on a sampling-based motion planning algorithm that uses the controller as local planner. The computed trajectories are guaranteed to satisfy task constraints, in addition to other geometric, kinematic and dynamic constraints. Assuming an accurate model and an appropriate control method, the use of the controller inside the planner guarantees the feasibility of the trajectory for the real system and also allows to better predict the behavior of the closed loop system as singularities or big tracking errors.

Another advantage of the proposed approach is that the use of control methods that directly treat the redundancy of the system allows the planner to search for a solution directly in the reduced and more relevant task space. Planning directly at the task level permits a more straightforward formulation of task-constrained motion planning problems, and in general reduces the dimensionality of the search space. This idea has been often exploited in related works (see, e.g., [13]). Finally, by properly defining the task one can choose a good trade-off between the dimension of the search space and the delegation of the redundant degrees of freedom to the local controller. It should be noted here that, although the proposed method has been firstly conceived for aerial manipulators, it can be applied to other types of system thanks to the generality of the paradigm.

The paper is organized as follow: we model the considered robotic system in Sec. II. The planning problem and the control-aware planning paradigm are described in Sec. III and Sec. IV, respectively. The proposed control strategy and planning algorithm are explained in details in Sec. V and Sec. VI, respectively. In Sec. VII we show the numerical results. Conclusions and future development are finally discussed in Sec. VIII.

II. MODEL

This section presents the mathematical model of an aerial vehicle equipped with a manipulator. Figure 1 shows an instance of the generic robotic system under study. We define an inertial frame $\mathscr{F}_W = \{O_W, \boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W\}$, with origin O_W and unit axes $(\boldsymbol{x}_W, \boldsymbol{y}_W, \boldsymbol{z}_W)$. The vector $\boldsymbol{z}_W$ is assumed to be directed in the opposite direction of the gravity vector.

A body-fixed frame $\mathscr{F}_R = \{O_R, \boldsymbol{x}_R, \boldsymbol{y}_R, \boldsymbol{z}_R\}$ is attached to the aerial vehicle, in which the origin O_R coincides with the Center of Mass (CoM) of the vehicle (without arm). The position and orientation of the vehicle w.r.t. $\mathscr{F}_W$ are given by the vector $\boldsymbol{p}_R \in \mathbb{R}^3$ and the rotation matrix $\boldsymbol{R}_R \in SO(3)$, respectively. We denote by $\boldsymbol{q}_R = (\boldsymbol{p}_R, \boldsymbol{R}_R) \in \mathscr{C}_R = \mathbb{R}^3 \times SO(3)$ the full configuration of the sole aerial vehicle. Its velocity can be defined as $\boldsymbol{v}_R = [\dot{\boldsymbol{p}}_R^\top \ \boldsymbol{\omega}_R^\top]^\top \in \mathscr{V}_R \subset \mathbb{R}^6$, where $\dot{\boldsymbol{p}}_R = d\boldsymbol{p}_R/dt$ and $\boldsymbol{\omega}_R \in \mathbb{R}^3$ is the angular velocity of $\mathscr{F}_R$ w.r.t. $\mathscr{F}_W$ and expressed in $\mathscr{F}_R$. The aerial vehicle is a multidirectional-thrust platform, with $n_R \in \mathbb{N}_{\geq 6}$ thrusters rigidly attached to the vehicle body. We define the vector $\boldsymbol{f}_R = [f_1 \dots f_{n_R}]^\top \in \mathbb{R}^{n_R}$ where $f_i \in \mathbb{R}_{\geq 0}$ for $i = 1, \dots, n_R$ is the force intensity produced by each rotor.

The vehicle is endowed with a robotic arm consisting of $n \in \mathbb{N}_{\geq 0}$ links. In order to describe its configuration, we rigidly attach to each link a frame $\mathscr{F}_{Ji} = \{O_{Ji}, \boldsymbol{x}_{Ji}, \boldsymbol{y}_{Ji}, \boldsymbol{z}_{Ji}\}$ with $i = 1, \dots, n$, using the standard Denavit-Hartenberg convention [14]. In particular, $\boldsymbol{z}_{Ji}$ is the axis of actuation of the i-th joint, either rotational or prismatic; $q_{Ji} \in \mathbb{R}$ denotes the rotation angle about $\boldsymbol{z}_{Ji}$ or the translation along $\boldsymbol{z}_{Ji}$. The configuration of the arm is then given by the vector $\boldsymbol{q}_A = [q_{J1} \dots q_{Jn}]^\top \in \mathscr{C}_A \subset \mathbb{R}^n$. The velocity of the arm is defined as $\boldsymbol{v}_A = \dot{\boldsymbol{q}}_A \in \mathscr{V}_A \subset \mathbb{R}^n$. We assume that each joint is driven by a motor applying a generalized torque $\tau_{Ji} \in \mathbb{R}$ along the joint axis $\boldsymbol{z}_{Ji}$, for $i = 1, \dots, n$. We define by $\boldsymbol{\tau}_A = [\tau_{J1} \dots \tau_{Jn}]^\top \in \mathbb{R}^n$ the vector containing all the motor torques.

We finally denote by $\boldsymbol{q} = (\boldsymbol{q}_R, \boldsymbol{q}_A) \in \mathscr{C} = \mathscr{C}_R \times \mathscr{C}_A$ the full configuration of the aerial robotic system, and by $\boldsymbol{v} = [\boldsymbol{v}_R^\top \ \boldsymbol{v}_A^\top]^\top \in \mathscr{V} = \mathscr{V}_R \times \mathscr{V}_A$ the corresponding velocity. Its acceleration is finally given by $\dot{\boldsymbol{v}} = [\dot{\boldsymbol{v}}_R^\top \ \dot{\boldsymbol{v}}_A^\top]^\top \in \mathscr{A} = \mathscr{A}_R \times \mathscr{A}_A$. Using Newton-Euler method, we can write the aerial manipulator dynamics in the following form:

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} = \boldsymbol{c}(\boldsymbol{q}, \boldsymbol{v}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{G}(\boldsymbol{q})\boldsymbol{u}, \tag{1}$$

where $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{(6+n)\times(6+n)}$ is the positive-definite inertia matrix, $\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{v}) \in \mathbb{R}^{(6+n)}$ is the vector collecting the centrifugal and Coriolis forces, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^{(6+n)}$ represents the gravitational term, $\boldsymbol{G}(\boldsymbol{q}) \in \mathbb{R}^{(6+n)\times(n_R+n)}$ is the input matrix, and $\boldsymbol{u} = [\boldsymbol{f}_R^\top \boldsymbol{\tau}_A^\top]^\top \in \mathbb{R}^{(n_R+n)}$ is the vector containing all the inputs of the aerial manipulator. The *robot state* is defined as $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{v}, \dot{\boldsymbol{v}}) \in \mathscr{X} = \mathscr{C} \times \mathscr{V} \times \mathscr{A}$, collecting the configuration, the velocity and the acceleration of the robotic system, whilst $\mathscr{X}$, is defined as the *robot extended*[1] *state space*, shortly called *state space* in the following.

[1] In fact, strictly speaking, the robot state space is only $\mathscr{C} \times \mathscr{V}$.

The *end-effector* of the arm is characterized by an attached frame denoted as $\mathscr{F}_E = \{O_E, \boldsymbol{x}_E, \boldsymbol{y}_E, \boldsymbol{z}_E\}$. Its configuration is given by $\boldsymbol{q}_E = (\boldsymbol{p}_E, \boldsymbol{R}_E) \in \mathscr{C}_E = \mathbb{R}^3 \times SO(3)$, with $\boldsymbol{p}_E \in \mathbb{R}^3$ and ${}^W\boldsymbol{R}_E \in SO(3)$ representing the corresponding position and orientation w.r.t. $\mathscr{F}_W$, respectively. The linear and angular velocities of the end-effector w.r.t. $\mathscr{F}_W$ are described by the vector $\boldsymbol{v}_E = [\dot{\boldsymbol{p}}_E^\top \ \boldsymbol{\omega}_E^\top]^\top \in \mathscr{V}_E \subset \mathbb{R}^6$. The end-effector pose, velocity and acceleration can be computed from the full state of the aerial manipulator $\boldsymbol{x}$, by using the forward kinematics $f_E(\cdot)$ and its time derivatives, :

$$\begin{aligned} \boldsymbol{q}_E = f_E(\boldsymbol{q}), \quad \boldsymbol{v}_E = \boldsymbol{J}_E(\boldsymbol{q})\boldsymbol{v}, \\ \dot{\boldsymbol{v}}_E = \boldsymbol{J}_E(\boldsymbol{q})\dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}_E(\boldsymbol{q},\boldsymbol{v})\boldsymbol{v} \end{aligned} \tag{2}$$

where $\boldsymbol{J}_E(\boldsymbol{q}) \in \mathbb{R}^{6\times(6+n)}$ is the Jacobian matrix. We call $\boldsymbol{x}_E = (\boldsymbol{q}_E, \boldsymbol{v}_E, \dot{\boldsymbol{v}}_E) \in \mathscr{X}_E = \mathscr{C}_E \times \mathscr{V}_E \times \mathscr{A}_E$, the *end-effector state*.

III. PROBLEM FORMULATION

In many applications related to manipulation, we want the end-effector of a robot to track a certain trajectory or, in the case of inspection by contact, to reach a series of points on a surface of interest while being in contact with the surface. In those cases, we are not interested on the full system motion, but rather in the correct execution of the sought *task*. The redundancy, if any, can be exploited to optimize other criteria.

Let us define a task characterized by a m-dimensional vector $\boldsymbol{y}$ belonging to the subset $\mathscr{C}_{\boldsymbol{y}} = \{\boldsymbol{y} \in \mathbb{R}^m \mid \boldsymbol{f}_{\boldsymbol{y}}^{\mathrm{c}}(\boldsymbol{y}) = \boldsymbol{0}\} \subseteq \mathbb{R}^m$, defined by the function $\boldsymbol{f}_{\boldsymbol{y}}^{\mathrm{c}} : \mathbb{R}^m \to \mathbb{R}^{m'}$ with $0 < m' \leq m$. The task $\boldsymbol{y}$ is linked to the robot configuration by the kinematic map $\boldsymbol{f}_{\boldsymbol{y}} : \mathscr{C} \to \mathbb{R}^m$, such that

$$\boldsymbol{y} = \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{q}). \tag{3}$$

Notice that a task is feasible if $\dim(\mathscr{C}_{\boldsymbol{y}}) \leq \dim(\mathscr{C})$. Then, let us call $\mathscr{C}_{\mathrm{obs}} \subset \mathscr{C}$ the set of configurations for which the robot is in collision with some obstacles. Consequently, $\mathscr{C}_{\mathrm{free}} = \mathscr{C} \setminus \mathscr{C}_{\mathrm{obs}}$ is the free space. In this manuscript we address the following task-constrained motion planning problem:

Problem 1. Consider the robot whose dynamic model is described by (1) with a certain initial state, $\boldsymbol{x}^0$. Consider also a certain task $\boldsymbol{y} \in \mathscr{C}_{\boldsymbol{y}}$, which is related to the robot configuration variables by (3). Given a desired final task value $\boldsymbol{y}^\star \in \mathscr{C}_{\boldsymbol{y}}$, the problem is to find a collision-free trajectory $\boldsymbol{q}(t) \in \mathscr{C}_{\mathrm{free}}$, which is feasible w.r.t. the robot dynamics, and such that task constraints are satisfied, i.e., $\boldsymbol{y}(T) = \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{q}(T)) = \boldsymbol{y}^\star$, and $\boldsymbol{y}(t) = \boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{q}(t)) \in \mathscr{C}_{\boldsymbol{y}}$ for all $t \in [0, T]$ and $T \in \mathbb{R}_{>0}$.

IV. ALGORITHM OVERVIEW

The proposed *control-aware motion planner* combines a control method, used as local planner, and a sampling-based algorithm to compute the global trajectory (global planner).

The approach is based on the following reformulation of the problem. Given a certain desired task trajectory, $\boldsymbol{y}^d(t)$, we define the vector $\boldsymbol{x}_{\boldsymbol{y}}^d = [\boldsymbol{y}^{d\top} \ \dot{\boldsymbol{y}}^{d\top} \ \ddot{\boldsymbol{y}}^{d\top}]^\top \in \mathscr{X}_{\boldsymbol{y}} = \mathscr{C}_{\boldsymbol{y}} \times \mathscr{V}_{\boldsymbol{y}} \times \mathscr{A}_{\boldsymbol{y}}$, called *task state*, where $\mathscr{V}_{\boldsymbol{y}}$ and $\mathscr{A}_{\boldsymbol{y}}$ are the space of the velocities and accelerations of the task, respectively. The space $\mathscr{X}_{\boldsymbol{y}}$ will be called *task state space* in the following. Then, we assume that a controller

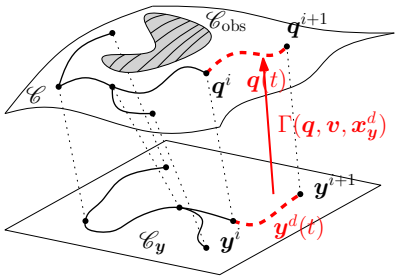


Fig. 2: Schematic representation of the control-aware planning paradigm. To each trajectory in the task space corresponds a motion of the robot, by-product of the used local controller.

$$\boldsymbol{u} = \Gamma(\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{x}_{\boldsymbol{y}}^d), \tag{4}$$

is applied to the system (1), such that the task error $\boldsymbol{e}_{\boldsymbol{y}} = \boldsymbol{y}^d(t) - \boldsymbol{y}(t)$ asymptotically converges to zero. If the system is redundant w.r.t. the task, the controller optimizes the remaining degrees of freedom according to given criteria (see Sec. V-B for the details). Inside the proposed motion planning algorithm, the controller $\Gamma(\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{x}_{\boldsymbol{y}}^d)$ will serve as local planner, also called *steering method*, to connect states. This will be further explained in Sec. VI. The task-constrained motion planning problem can be then reformulated as:

Problem 2. Find a desired task trajectory $\boldsymbol{y}^d(t)$ that, used as reference for the controller (4), will generate a robot motion that solves Problem 1.

The sampling-based algorithm can then search for a solution to Problem 2 directly in $\mathscr{X}_{\boldsymbol{y}}$ (also called *search space*), delegating to the controller the generation of the full robot configuration. For a planned (local) trajectory in the task space $\boldsymbol{y}^d(t)$ for $t \in [0 \ldots T]$, the corresponding motion of the full robot $\boldsymbol{q}(t)$ is obtained simulating the closed loop system. The validity (i.e., collision freeness, input feasibility, etc.) of the states generated by the controller is checked by the global planner. Indeed, the controller might not have an obstacle avoidance feature or the capability to check for other constraints of the system. Figure 2 shows a schematic representation of the approach.

Once the planner finds a solution, the planned trajectory in the task space $\boldsymbol{y}^d(t)$ can be given as reference to the controller for execution. Since the closed loop system has already been simulated inside the planner, even in the case of tracking errors, the real behavior of the system will be very close to the planned one. Thus, the execution of the trajectory will be in general more consistent and reliable.

Trade-off between delegation and exploration

In our approach, the management of redundant degrees of freedom (w.r.t. the task) is delegated to the controller. It is clear that a high level of delegation implies a small dimension of the search space, and thus a lower complexity. However, it also forces the global planner to look for a solution only

in the search space, i.e., the space of DoFs of the robot that are 'non-redundant' w.r.t. to the task at hand. In other words this approach reduces the 'exploration authority' of the global planner for the sake of a reduced complexity. This is an important issue for the (probabilistic) completeness of the planner, which may fail to find exiting solutions in some cases. To circumvent this problem, one can consider additional 'tasks' that increase the dimension of the search space, increasing the chances to find a solution. In particular consider that a task $\boldsymbol{y}' \in \mathscr{C}_{\boldsymbol{y}}{}'$ and the relative task space $\mathscr{X}_{\boldsymbol{y}}{}'$ has been defined. Let us assume that the planner, searching in $\mathscr{X}_{\boldsymbol{y}}{}'$ for a solution, after a certain time, it is still not able to find a trajectory $\boldsymbol{y}^{d'}(t)$ solution of Problem 2. To increase the chances of finding a solution we can then enlarge the search space defining an additional task $\boldsymbol{y}'' \in \mathscr{C}_{\boldsymbol{y}}{}''$ and the relative task space $\mathscr{X}_{\boldsymbol{y}}{}''$, such that $\dim(\mathscr{C}_{\boldsymbol{y}}{}') < \dim(\mathscr{C}_{\boldsymbol{y}}{}' \times \mathscr{C}_{\boldsymbol{y}}{}'') \leq \dim(\mathscr{C})$. We can then consider $\mathscr{X}_{\boldsymbol{y}} = \mathscr{X}_{\boldsymbol{y}}{}' \times \mathscr{X}_{\boldsymbol{y}}{}''$ as a new task space for the planner, in which it will search for a solution $\boldsymbol{y}^d(t) = [\boldsymbol{y}^{d'}(t)^\top \ \boldsymbol{y}^{d''}(t)^\top]^\top$. One has only to modify the second-order inverse kinematics controller to follow the new extended task $\boldsymbol{y}$. Doing so, we remove constraints to the planner, adding them to the controller.

The addition of new tasks to increase the dimension of the search space can be done incrementally. When the planner struggles on finding a solution inside $\mathscr{X}_{\boldsymbol{y}}$, a new task involving one or a small number of DoFs is added until a solution is found, if this exists. In the worst case, we will reach $\dim(\mathscr{X}_{\boldsymbol{y}}) = \dim(\mathscr{X})$, thus preserving the probabilistic completeness of the sampling-based planner, which comes at the cost of increasing complexity. Nevertheless, this worst scenario will rarely happen in practical applications.

We have to highlight that the need of increasing the dimension of $\mathscr{X}_{\boldsymbol{y}}$ strongly depends on the local controller. For example, if the controller already includes a local obstacle avoidance capability, the planner will probably find a solution more easily, without the need of extending $\mathscr{X}_{\boldsymbol{y}}$. Note also that, using a more sophisticated implementation of the planner, the extension of $\mathscr{X}_{\boldsymbol{y}}$ could be done only temporary. If the planner gets blocked in a region of the space, the dimension of $\mathscr{X}_{\boldsymbol{y}}$ can be locally increased only in this area. Afterwards, the search space can be reduced back to the original dimension.

V. CONTROL STRATEGY

The proposed control strategy is based on the separation between the kinematic and the dynamic loops, yielding the approach known in the literature as kinematic control. This approach is particularly suited to handle redundant systems and thus to assign multiple control objectives beyond the sole arm's end-effector, e.g., obstacle avoidance or mechanical joint limits, etc. Assigning a relative priority leads to what is known as task-priority inverse kinematics, which has been successfully implemented for aerial manipulation [6]. Such methods apply null-space-based behavioral control [7]. Figure 3 shows a schematic representation of the proposed control strategy.

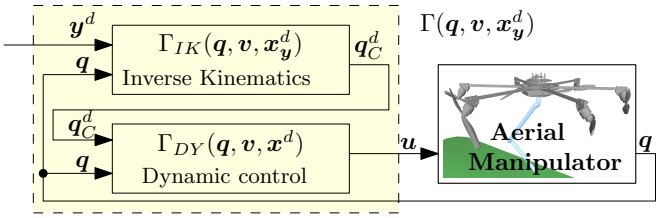


Fig. 3: Block diagram describing the control strategy.

We can decompose the controller (4) in the following way:

$$\boldsymbol{x}^d = \Gamma_{IK}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{x}^d_{\boldsymbol{y}}), \quad \boldsymbol{u} = \Gamma_{DY}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{x}^d), \tag{5}$$

where $\boldsymbol{x}^d = [\boldsymbol{q}^{d\top} \ \boldsymbol{v}^{d\top} \ \dot{\boldsymbol{v}}^{d\top}]^\top$, $\Gamma_{DY}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{x}^d)$ and $\Gamma_{IK}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{x}^d_{\boldsymbol{y}})$ are the dynamic and inverse kinematics controller, respectively. Due to the inner-outer loop nature of the proposed control strategy, the performance of the inverse kinematics controller to minimize the task error (outer loop) will improve when the dynamic controller (inner loop) is able to let the robot accurately follow the reference trajectory.

A. Dynamic controller

One of the positive features of using the control architecture presented above is that the inverse kinematics and the dynamic controllers are independent one of each other. A *simple* approach for the dynamic controller is obtained by considering the vehicle and the arm as two separate subsystems, i.e., without physical interaction [2], [3]. This control strategy obviously suffers for demanding trajectories, when the interaction terms become significant. More advanced non linear controllers consider the full system dynamics providing better performance [1], [4], [15]. However, since they strongly depend on the dynamic model, they are in general not very robust to model uncertainties. A recent technique is instead based on the flatness of the system to develop a controller that considers the full system dynamics but is still robust to modeling errors [5].

For the system considered in this work, we implemented a standard feedback linearization technique. This is possible because the considered vehicle is fully actuated, i.e., considering (1), $\boldsymbol{G}(\boldsymbol{q})$ is full rank. Therefore, we designed the controller as $\Gamma_{DY}(\boldsymbol{q},\boldsymbol{v},\boldsymbol{x}^d) = \boldsymbol{G}(\boldsymbol{q})^{-1}(\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}}^\star - \boldsymbol{c}(\boldsymbol{q},\boldsymbol{v}) - \boldsymbol{g}(\boldsymbol{q}))$ where $\dot{\boldsymbol{v}}^\star$ is the desired acceleration that brings the tracking error to zero, i.e., $\dot{\boldsymbol{v}}^\star = \dot{\boldsymbol{v}}^d + \boldsymbol{K}_D^{DY}(\boldsymbol{v}^d - \boldsymbol{v}) + \boldsymbol{K}_P^{DY}(\boldsymbol{q}^d - \boldsymbol{q})$.

B. Inverse kinematics controller

Inverse kinematics techniques can be implemented by resorting to the first or second order derivatives. In this case, in order to achieve smooth trajectories, the latter have been taken into account. Let us consider the task $\boldsymbol{y} \in \mathscr{C}_{\boldsymbol{y}}$ defined in (3) and differentiate it twice

$$\dot{\boldsymbol{y}} = \frac{\partial \boldsymbol{f}_{\boldsymbol{y}}}{\partial \boldsymbol{q}}\boldsymbol{v} = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{v}, \qquad \ddot{\boldsymbol{y}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{v}} + \dot{\boldsymbol{J}}(\boldsymbol{q},\boldsymbol{v})\boldsymbol{v}, \tag{6}$$

where $\boldsymbol{J}(\boldsymbol{q})$ is the Jacobian of the task w.r.t. $\boldsymbol{q}$. The trajectory for the robot configuration, $\boldsymbol{q}^d(t)$, that steers $\boldsymbol{y}(t)$ along to

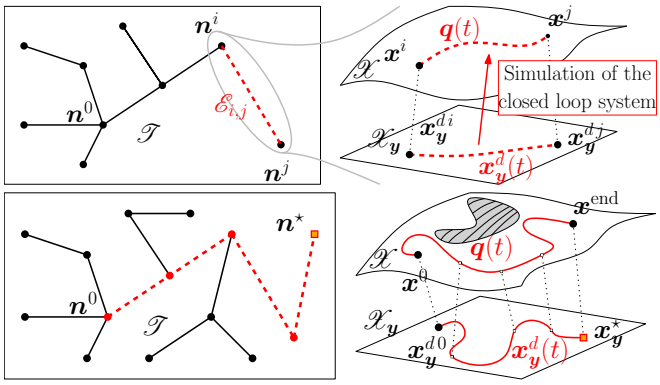


Fig. 4: Representation of the steering method (top) and of the extraction of solution trajectory (bottom). The images on the left represent the planning tree. On the right, we illustrate how one edge of the tree (top) and a solution trajectory (bottom) are mapped from the task space to the robot state space using the local controller.

Algorithm 1: Control-Aware Motion Planner

Input:
task $\boldsymbol{y} \in \mathscr{C}_{\boldsymbol{y}}$, in terms of functions $\boldsymbol{f}^{\mathrm{c}}_{\boldsymbol{y}}(\boldsymbol{y})$ and $\boldsymbol{f}_{\boldsymbol{y}}(\boldsymbol{q})$
initial robot state, $\boldsymbol{x}^0 \in \mathscr{X}$
desired final task value $\boldsymbol{y}^\star \in \mathscr{C}_{\boldsymbol{y}}$
Output: task trajectory $\boldsymbol{x}^d_{\boldsymbol{y}}(t)$ for $t \in [0, T]$
Main:
$\boldsymbol{x}^0_{\boldsymbol{y}} \leftarrow$ `forwardKinematics`$(\boldsymbol{x}^0)$
$\boldsymbol{n}^0 \leftarrow$ `initVertex`$(\boldsymbol{x}^0_{\boldsymbol{y}}, \boldsymbol{x}^0)$
$\mathscr{T} \leftarrow$ `initTree`$(\boldsymbol{n}^0)$
while not `stopCondition`$(\mathscr{T}, \boldsymbol{y}^\star)$ **do**
  $\boldsymbol{x}^{d\,rand}_{\boldsymbol{y}} \leftarrow$ `sampleRandomTask`$(\mathscr{X}_{\boldsymbol{y}})$
  $\boldsymbol{n}^{near} \leftarrow$ `nearestNeighbor`$(\mathscr{T}, \boldsymbol{x}^{d\,rand}_{\boldsymbol{y}})$
  $\boldsymbol{n}^{next} \leftarrow$ `extend`$(\boldsymbol{n}^{near}, \boldsymbol{x}^{d\,rand}_{\boldsymbol{y}})$
  if `notEmpty`$(\boldsymbol{n}^{next})$ **then**
    `addNode`$(\mathscr{T}, \boldsymbol{n}^{next})$
$\boldsymbol{y}^d(t) \leftarrow$ `getTaskTrajectory`$(\mathscr{T})$
return $\boldsymbol{x}^d_{\boldsymbol{y}}(t)$

$\boldsymbol{y}^d(t)$, can be computed integrating $\dot{\boldsymbol{v}}^d$, which is calculated by the following expression:

$$\dot{\boldsymbol{v}}^d = \boldsymbol{J}^\dagger \left(\ddot{\boldsymbol{y}}^r - \dot{\boldsymbol{J}}\boldsymbol{v} \right), \quad \ddot{\boldsymbol{y}}^r = \ddot{\boldsymbol{y}}^d + \boldsymbol{K}_D \dot{\boldsymbol{e}}_{\boldsymbol{y}} + \boldsymbol{K}_P \boldsymbol{e}_{\boldsymbol{y}}, \qquad (7)$$

where $\boldsymbol{K}_D \in \mathbb{R}^{m \times m}$ and $\boldsymbol{K}_P \in \mathbb{R}^{m \times m}$ are positive defined matrices, and $\boldsymbol{J}^\dagger$ is the Moore-Penrose pseudo-inverse of the Jacobian matrix.

For the robotic system under study, $\mathscr{C}_{\boldsymbol{y}}$ has a lower dimension w.r.t. $\mathscr{C}$, i.e., the system is redundant. Therefore, a multi priority approach can be used such as, e.g. [16]. As anticipated, the tasks are handled in a *flexible* way, i.e., by adding/removing tasks based on the need to find a solution to Problem 2.

VI. SAMPLING BASED GLOBAL PLANNER

For the sake of clarity, we will consider here only the case in which the search space is equal to the task space. Treating the general case would involve the addition of new variables to increase the dimension of the search space, and the corresponding modification in the controller to consider the associated 'tasks' as explained in Sec. V-B.

Algorithm 1 shows the pseudo-code of the proposed control-aware planner. In this work, we have applied the Rapidly-Exploring Random Trees (RRT) algorithm [11] for the exploration of the task space. However, as it will be explained in Section VIII, other algorithms could be used. The algorithm samples states in the task space with a uniform distribution and extends the exploration tree using the controller until a leaf reaches the desired goal.

Although the RRT algorithm operates in the task space, each vertex of the exploration tree, $\mathscr{T}$, denoted by the symbol $\boldsymbol{n}^i$, contains the desired task state $\boldsymbol{x}^{d\,i}_{\boldsymbol{y}} := \boldsymbol{x}^d_{\boldsymbol{y}}(t_i)$ and the corresponding robot state $\boldsymbol{x}^i := \boldsymbol{x}(t_i)$ at a certain time $t_i \in \mathbb{R}_{>0}$. The edge, $\mathscr{E}_{i,j}$, from vertex $\boldsymbol{n}^i$ to $\boldsymbol{n}^j$ contains the desired task trajectory $\boldsymbol{x}^d_{\boldsymbol{y}}(t)$ from value $\boldsymbol{x}^{d\,i}_{\boldsymbol{y}}$ to $\boldsymbol{x}^{d\,j}_{\boldsymbol{y}}$, and the outcome of the simulated closed loop system, namely $\boldsymbol{q}(t)$ for $t \in [t_i, t_j]$. The latter is needed to check the feasibility of the local motion. Fig. 4 illustrates the several stages of the planning process.

Starting from a given initial robot state $\boldsymbol{x}^0$, the tree is initialized computing the initial task state $\boldsymbol{x}^0_{\boldsymbol{y}}$ corresponding to $\boldsymbol{x}^0$. This is done using forward kinematics, i.e., equations (3) and (7). The initial vertex $\boldsymbol{n}^0 = (\boldsymbol{x}^0, \boldsymbol{x}^0_{\boldsymbol{y}})$ is then added to the tree. The methods used at each iteration of the RRT algorithm are discussed in the following sub-sections. Once the planner finds a solution, the desired task trajectory is returned. The latter will be then used as reference for the controller, for execution with the real robot.

A. Steering method

The function `extend`$(\boldsymbol{n}^i, \boldsymbol{x}^j_{\boldsymbol{y}})$ in Algorithm 1 is divided into three main phases:

1) First, a trajectory for the desired task $\boldsymbol{x}^d_{\boldsymbol{y}}(t)$ is computed such that $\boldsymbol{x}^d_{\boldsymbol{y}}(t_i) = \boldsymbol{x}^{d\,i}_{\boldsymbol{y}}$ and $\boldsymbol{x}^d_{\boldsymbol{y}}(t_j) = \boldsymbol{x}^{d\,j}_{\boldsymbol{y}}$ for a certain $t_j > t_i$. To this purpose, any trajectory generator can be applied. However, with the aim to produce short trajectories, we use the minimum-time trajectory generator presented in [17]. This method ensures the generation of a continuous and derivable trajectory (up to the 4^{th} order) respecting the constraint $\boldsymbol{x}^d_{\boldsymbol{y}}(t) \in \mathscr{X}_{\boldsymbol{y}}$ for all $t \in [t_i, t_j]$.
2) The closed loop system, i.e., (1) together with (5), is simulated starting from the robot state $\boldsymbol{x}^i$ and using $\boldsymbol{x}^d_{\boldsymbol{y}}(t)$, computed at the previous step, as reference for the controller. The numerical simulation provides the motion of the robot, i.e., $\boldsymbol{x}(t)$, while tracking $\boldsymbol{x}^d_{\boldsymbol{y}}(t)$ for $t \in [t_i, t_j]$.
3) If $\boldsymbol{x}(t) \in \mathscr{X}$ and $\boldsymbol{q}(t) \in \mathscr{C}_{\text{free}}$ for all $t \in [t_i, t_j]$, the motion can be considered as 'valid'. Afterwards the node $\boldsymbol{n}^j = (\boldsymbol{x}^{d\,j}_{\boldsymbol{y}}, \boldsymbol{x}(t_j))$ and the edge $\mathscr{E}_{i,j}$ are added to the tree.

B. Metric in the task state space

In kinodynamic planning, the metric has a crucial role to obtain good results. However, computing the distance between two states can be computationally very expensive [11]. In fact, one should try to extend every node in the tree toward the sampled state and add to the tree only the edge characterized by the feasible trajectory with minimum time.

This will drastically increase the computational cost. Instead, we use the *approximate quasi-metric* presented in [17]. This method allows to compute in closed-form an approximation of the time associated with the local path that would be obtained by the steering method trying to extend a vertex. It has been shown that it is a good approximation that allows to substantially reduce the computing time for the global planner.

VII. RESULTS

For the evaluation of the planner, we considered the aerial manipulator shown in Fig. 1: an hexarotor with tilted propellers (presented in [1]) endowed with a planar 2-link robotic arm (presented in [5]). A sensor for surface inspection is installed at the end-effector. The popularity of multidirectional-thrust vehicles w.r.t. unidirectional-thrust platform is increasing thanks to their proven superiority for aerial manipulation [1]. Indeed, they allow to independently control position and orientation, and to instantaneously react to interaction forces and external disturbances.

The vehicle is characterized by a mass, $m_R = 1.2$ [kg], and an inertia tensor w.r.t. $\mathcal{F}_R$, $\boldsymbol{J}_R = \text{diag}(0.02, 0.02, 0.025)$ [kg $\cdot$ m^2]. The lengths of the first and second links of the arm are 0.3 [m] and 0.25 [m], respectively. Their masses are 0.145 [kg] and 0.123 [kg], respectively. The arm is attached at 0.05 [m] below the CoM of the aerial vehicle, along the z-axis. We tested the method in simulation using two scenarios:

a) A challenging cluttered environment with several obstacles. One of them is a concave obstacle made of three walls in a U-shaped configuration. This type of obstacle can be a trap for classical local controllers. The surface being inspected is the flat ground.
b) An application-oriented scenario. The task consists in the inspection by contact of a cylindrical pipe in an industrial site. The surface of interest is the outer part of the pipe.

In both cases, the task consists in safely bringing the end-effector of the robot, endowed with the inspection sensor, to a desired point on the surface of interest, keeping the contact while moving. In Scenario b) the task includes the orientation of the end-effector as well. Indeed, the last link has to be perpendicular to the surface to properly gather the data.

Once defined the task, the robot is controlled as defined in Sec. V, i.e., using a second order inverse kinematics controller plus a dynamic controller based on dynamics inversion.

Figure 5 shows the planned trajectories for the two considered scenarios. One can notice how the search tree grows in the task space until the planner finds a task trajectory, $\boldsymbol{y}^d(t)$, solving Problem 2. Videos showing the robot following the trajectories are in the multimedia attachment.

We compared the proposed method against a purely-reactive (optimization-based) method based on a local controller using a simplified version of Scenario a) involving the U-shaped obstacle only. In particular, we used the NSB method including an obstacle avoidance feature based on virtual potential fields. As expected, using only the local controller, the robot gets trapped between obstacles and does not reach the goal. Additional details about this experiment are included in a technical report in the multimedia attachment.

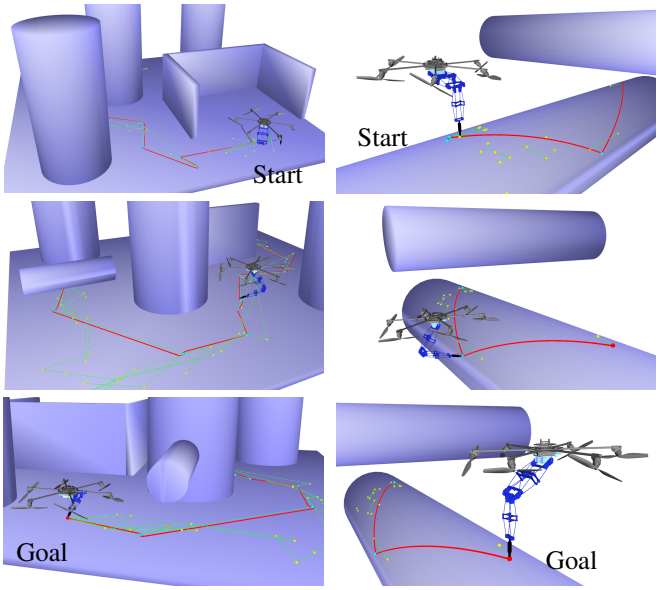


Fig. 5: Images of the motions provided by the control-aware planner for the Scenario a) (left) and Scenario b) (right). The desired task trajectory is represented by a red line, ending in the goal task position. The yellow dots are the vertexes of the tree, while the green lines are the edges. The images (from the top to the bottom) show intermediate snap-shoots along the solution trajectories.

		Planning time [s]	Rejected nodes #	Accepted nodes #	Trajectory time [s]	Path length [m]
a)	Average	47.26	58.75	62.15	97.34	9.6
	σ	6.61	18.5	2.25	8.14	0.71
b)	Average	21.92	28.5	22.5	35.6	3.7
	σ	5.3615	11	0.8	4.08	0.5

	Steering method	Collision checking	Neighbor search	Selection of the best path
% of total time	63.78	1.54	28.94	5.72

TABLE I: Performance of the planner out of 20 runs for the two scenarios. σ stands for the standard deviation.

The planner has been implemented in ROS using *Moveit!*[2] and the OMPL library[3]. To quantify the performance of the planner, we run the method 20 times for each scenario. Tab. I reports the average and the standard deviation of the most meaningful variables. It also shows the repartition of the total planning time between the four major operations. One can observe that the majority of the time is taken by the steering method, which has to simulate the system. Note however that we have used a preliminary implementation, which could be substantially improved to reduce computing time.

VIII. CONCLUSIONS

We have presented a task-constrained motion planner for aerial manipulation based on the paradigm of combining sampling based motion planning methods together with local controllers. The motivation comes from the fact that,

[2] http://moveit.ros.org/
[3] http://ompl.kavrakilab.org/

although trajectories generated by planning methods may have good theoretical properties, in practice, control methods applied for motion execution may have difficulties to track them. This is particularly true for complex robot systems such as aerial manipulators. Therefore, we propose to apply control methods already at the trajectory planning stage. More precisely, we considered a second-order inverse kinematics controller together with a dynamic controller. The use of this controller inside a RRT-based planner has a threefold advantage: i) generating motions that are feasible for the real closed loop system; ii) planning directly in the reduced and more relevant task space; iii) considering control singularities and tracking errors already at the planning level.

In this work, we have applied a basic RRT algorithm as global planner. However, the proposed paradigm can be extended to other planners. Note however that the use of the controller as a local planner imposes some restrictions at this level, since planning algorithms requiring the solution of a two-point boundary value problem (BVP) to connect sampled states involving the full system cannot be directly applied in this context. This is for instance the case of RRT* [18], which is an asymptotically-optimal variant of RRT. An interesting alternative for optimal trajectory planning would be the SST* algorithm [19], which does to rely on a BVP solver. Other technical improvements of the algorithm can be investigated, e.g., biased sampling to favor particular regions of the search space, post-processing to shorten the trajectory, or real-time implementation to take into account dynamic environments or incremental map building, etc.

As natural future work, we aim to test the proposed approach on a real aerial manipulator. Finally, we should mention that the general ideas proposed in this paper could be applied to other areas in robotics in addition to aerial manipulation.

REFERENCES

[1] M. Ryll, G. Muscio, F. Pierri, E. Cataldi, G. Antonelli, F. Caccavale, and A. Franchi, "6D physical interaction with a fully actuated aerial robot," in *2017 IEEE Int. Conf. on Robotics and Automation*, Singapore, May 2017, pp. 5190–5195.

[2] F. Ruggiero, M. A. Trujillo, R. Cano, H. Ascorbe, A. Viguria, C. Perz, V. Lippiello, A. Ollero, and B. Siciliano, "A multilayer control for multirotor uavs equipped with a servo robot arm," in *2015 IEEE Int. Conf. on Robotics and Automation*, Seattle, WA, May 2015, pp. 4014–4020.

[3] K. Kondak, K. Krieger, A. Albu-Schäffer, M. Schwarzbach, M. Laiacker, I. Maza, A. Rodriguez-Castano, and A. Ollero, "Closed-loop behavior of an autonomous helicopter equipped with a robotic arm for aerial manipulation tasks," *International Journal of Advanced Robotic Systems*, vol. 10, pp. 1–9, 2013.

[4] 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.

[5] 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.

[6] K. Baizid, G. Giglio, F. Pierri, M. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero, "Behavioral control of unmanned aerial vehicle manipulator systems," *Autonomous Robots*, vol. 35, no. 8, pp. 1–18, 2016.

[7] G. Antonelli, F. Arrichiello, and S. Chiaverini, "The entrapment/escorting mission: An experimental study using a multirobot system," *IEEE Robotics & Automation Magazine*, vol. 15, no. 1, pp. 22–29, 2008.

[8] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, May 2009, pp. 625–632.

[9] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[10] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Trans. on Robotics*, vol. 26, no. 3, pp. 576–584, 2010.

[11] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] M. Cefalo and G. Oriolo, "Task-constrained motion planning for underactuated robots," in *2015 IEEE Int. Conf. on Robotics and Automation*, Seattle, WA, May 2015, pp. 2965–2970.

[13] A. Shkolnik and R. Tedrake, "High-dimensional underactuated motion planning via task space control," in *2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Nice, France, Sep. 2008, pp. 3762–3768.

[14] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2009.

[15] B. Yüksel, G. Buondonno, and A. Franchi, "Differential flatness and control of protocentric aerial manipulators with any number of arms and mixed rigid-/elastic-joints," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Daejeon, South Korea, Oct. 2016, pp. 561–566.

[16] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Dynamic multi-priority control in redundant robotic systems," *Robotica*, vol. 31, no. 7, p. 11551167, 2013.

[17] A. Boeuf, J. Cortés, R. Alami, and T. Siméon, "Enhancing sampling-based kinodynamic motion planning for quadrotors," in *2015 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Hamburg, Germany, Sep. 2015, pp. 2447–2452.

[18] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[19] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

Additional Analysis and Simulations for the Control-Aware Planner

Technical report of:
"Control-Aware Motion Planning for Task-Constrained Aerial Manipulation"
IEEE Robotics and Automation Letters, Special Issue on Aerial Manipulation

M. Tognon[1], E. Cataldi[2], H. A. Tello Chavez[1], G. Antonelli[2], J. Cortés[1], A. Franchi[1]

***Abstract*— This document is a technical attachment to [1] as an extension of the validation part. Here we present additional plots, simulations, and analysis of the proposed method. The parameter of the simulated system are the one reported in [1].**

I. How to Cite this Work

This technical report is accompanying our IEEE Robotics and Automation Letters paper [1]. If you wish to reference this work, please cite this paper as follows:

```
@Article{Tognon18ral,
  author  = {M. Tognon and E. Cataldi and
     H. A. Tello Chavez and G. Antonelli and
     J. Cort\'es and A. Franchi},
  title   = {Control-Aware Motion Planning for
     Task-Constrained Aerial Manipulation},
  journal = {{IEEE} Robot. Autom. Lett.,
     Special Issue on Aerial Manipulation},
  year    = {2018},
  doi     = {10.1109/LRA.2018.2803206},
}
```

II. Comparison with Only Local Control Strategy

We compared the proposed method against a purely-reactive (optimization-based) method based on a local controller using a simplified version of Scenario a) involving the U-shaped obstacle.

Figure 1a shows the results obtained using only the local controller to achieve the goal task. For this, we have used the NSB method including an obstacle avoidance feature based on virtual potential fields. As expected, the robot gets trapped between the walls and does not reach the goal.

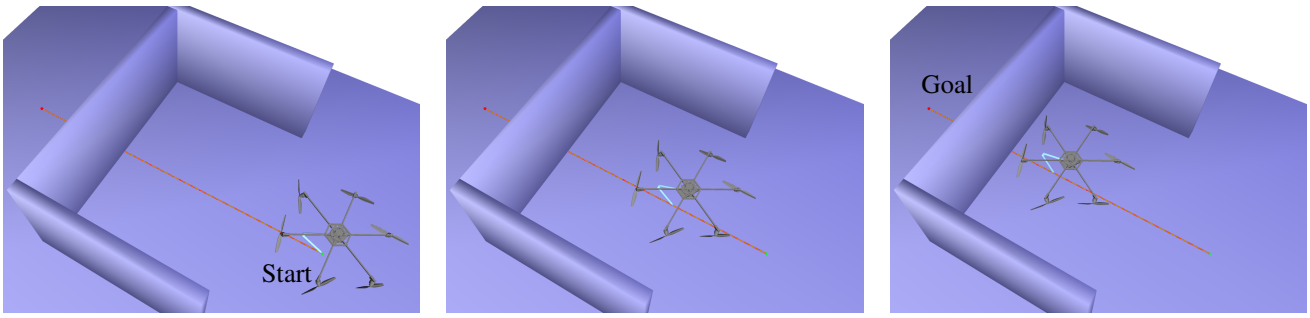
On the contrary, the proposed control-aware-planner is able to find a proper solution to Problem 2 of [1], providing a good task trajectory, $\boldsymbol{x}_{\boldsymbol{y}}^{d}(t)$, that brings the end-effector to the desired goal avoiding the obstacles. The growth of the tree, the final desired task trajectory, and the relative execution is shown in Fig. 1b.

III. Additional Simulation Results

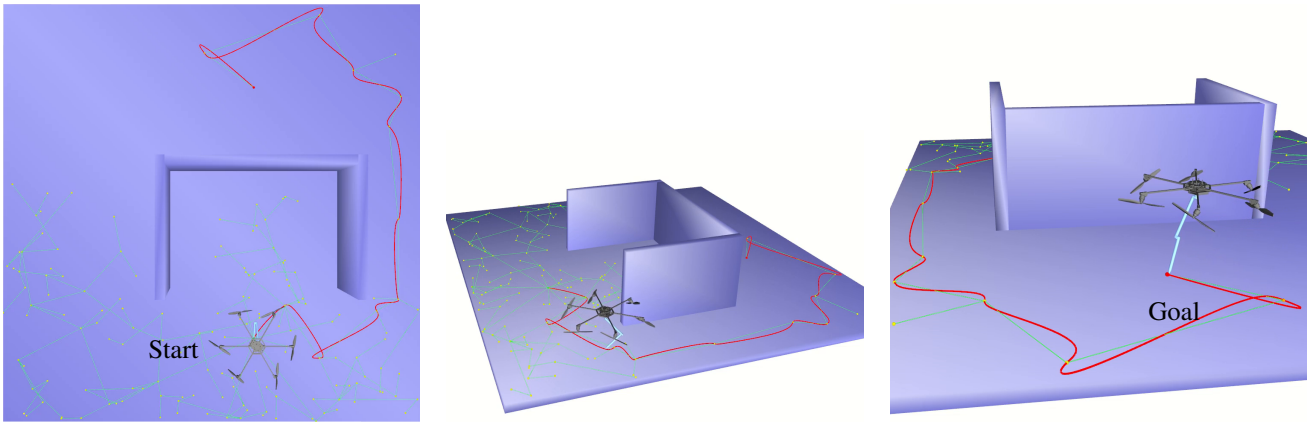
In the following we present more detailed plots of the trajectory planed and executed by the robot in the two scenarios considered in Sec. VII of the paper. In particular, Fig. 2a and Fig. 2b show the tracking of the end-effector trajectory and the evolution of the configuration of the robot for scenario a) and b), respectively.

References

[1] M. Tognon, E. Cataldi, H. Tello Chavez, G. Antonelli, J. Cortés, and A. Franchi, "Control-aware motion planning for task-constrained aerial manipulation," *IEEE Robotics and Automation Letters, Special Issue on Aerial Manipulation*, 2018.

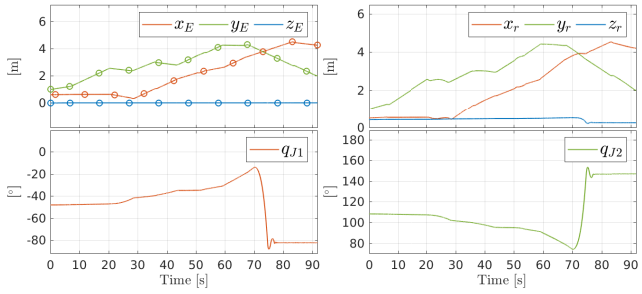


(a) Only local controller with an obstacle avoidance feature.

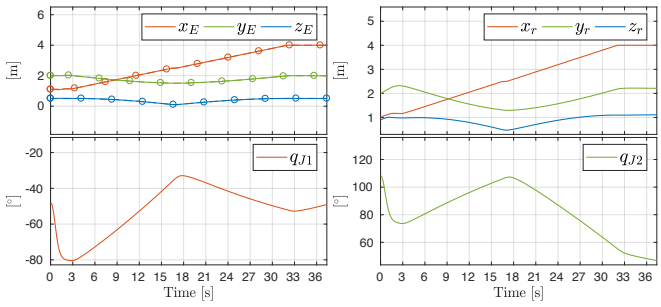


(b) Solution fond by the control-aware planner.

Fig. 1: Images of environment and of the motions provided by the local controller alone (a) and by the proposed control-aware planner (b). In both cases, the desired task trajectory is represented by a red line, ending in the goal task position. In Fig. (b), the yellow dots are the vertexes of the tree, while the green lines are the edges.



(a) Scenario a).



(b) Scenario b).

Fig. 2: End-effector tracking and configurations variables corresponding to the motion found by the planner for the two scenarios considered in Sec. VII of the paper. (x_E, y_E, z_E) and (x_r, y_r, z_r) are the coordinates of the position of the end-effector and of the aerial vehicle, respectively. (q_{J1}, q_{J1}) are the joint angles of the arm. The line with circles corresponds to the desired end-effector trajectory provided by the planner.