Trajectory Generation for Unmanned Aerial Manipulators through Quadratic Programming

Roberto Rossi[1*], Angel Santamaria-Navarro[2*], Juan Andrade-Cetto[2] and Paolo Rocco[1]

***Abstract*—In this paper a trajectory generation approach using quadratic programming is described for aerial manipulation, *i.e.* for the control of an aerial vehicle equipped with a robot arm. The proposed approach applies the online active set strategy to generate a feasible trajectory of the joints, in order to accomplish a set of tasks with defined bounds and constraint inequalities. The definition of the problem in the acceleration domain allows to integrate and perform a large set of tasks and, as a result, to obtain smooth motion of the joints. A weighting strategy, associated with a normalization procedure, allows to easily define the relative importance of the tasks. This approach is useful to accomplish different phases of a mission with different redundancy resolution strategies. The performance of the proposed technique is demonstrated through real experiments with all the algorithms running onboard in real time. In particular, the aerial manipulator can successfully perform navigation and interaction phases, while keeping motion within prescribed bounds and avoiding collisions with external obstacles.**

***Index Terms*—Aerial Manipulation, Trajectory Generation, Aerial Robotics, Mobile Manipulation.**

I. Introduction

In the past few years, unmanned aerial vehicles (UAVs), and in particular multirotor systems, have received special attention by the research community thanks to the improvement of payload capacity and maneuverability together with a decrease in cost.Moreover, advances in the design of lightweight arms opened new application domains for UAVs such as to perform aerial manipulation [1], [2]. The UAVs incorporating one or more serial arms are known as unmanned aerial manipulators (UAM).

The control of these platform-arm systems is a very challenging problem, considering the under-actuation of the platform, the coupled dynamics between the two systems, and the interaction with the environment during manipulation. In this regard [3] presented an adaptive controller to deal with changes in the platform center of gravity (COG) produced by a suspended load. A Cartesian impedance control was designed and illustrated with a simulation case study in [4], which provides a desired relationship between external wrench and the system motion. A different approach is [5], where an adaptive controller is set based on output feedback linearization to compensate the unknown displacement of the center of mass during aggressive maneuvers. However, only the case of a quadrotor is studied. In [6] and [7] hierarchical control laws are presented, considering the vehicle kinematics to achieve several tasks during UAM missions. In this paper we follow a similar idea of task definitions but focus instead on a numerical optimization solution for the trajectory generation.

Even when the state of the art in control algorithms for UAMs is extensive, solutions for trajectory generation using optimal control in real time are rare. These methods usually require powerful computational units due to their iterative nature. [8] and [9] describe methods for 3D optimal trajectory generation and control, however their works are focused only on UAVs, thus considering few degrees of freedom (DOFs). In [10], this trajectory generation is optimized for a vehicle with a cable-suspended load computing nominal trajectories with various constraints regarding the load swing. The application of such optimal control for UAMs can be seen in [11], where a nonlinear model predictive control scheme is proposed to achieve pick-and-place operations. Another example is [12] where a linear model predictive control is described using a direct multiple shooting method. However, results for UAMs are only shown in a simulated environment.

Drawing inspiration from other robotic fields (*e.g.*, [13], [14]), in this paper we take advantage of a quadratic programming technique to solve several UAM tasks in real time, on board and subject to constraints. Specifically we use the online active set strategy ([15], [16]) which analyses the constraints that are active at the current evaluation point, and gives us a subset of inequalities to watch while searching for the solution, which reduces the complexity of the search and thus the computation time.

In contrast to global offline trajectory generation algorithms (*e.g.*, RRT* [17]) we obtain the trajectory by iteratively considering local second order approximations of the system. Although our approach can lose global optimality, the main advantages are twofold: First, the convex quadratic problem can be solved online, thanks to the lower computational burden, enabling a fast reactive behavior with environment changes. Second, by representing the system in the accelerations domain we obtain smooth trajectories in velocity and position spaces.

Manuscript received: September, 10, 2016; Accepted November, 7, 2016.
This paper was recommended for publication by Editor Jonathan Roberts upon evaluation of the Associate Editor and Reviewers' comments.
*Corresponding authors.
[1]Roberto Rossi and Paolo Rocco are with Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza L. Da Vinci 32, 20133, Milano, Italy, {roberto.rossi, paolo.rocco}@polimi.it
[2]A. Santamaria-Navarro and J. Andrade-Cetto are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, Barcelona 08028, Spain, {asantamaria, cetto}@iri.upc.edu. Their work has been partially funded by the EU project AEROARMS H2020-ICT-2014-1-644271 and by the Spanish Ministry of Economy and Competitiveness project ROBINSTRUCT TIN2014-58178-R.
The paper has a supplementary downloadable video, available at http://ieeexplore.ieee.org, showing experimental results of the presented approach.
Digital Object Identifier (DOI): see top of this page.

A similar approach is [18], where the common idea consists in dividing the problem in two parts, firstly accounting for the trajectory generation and then implementing a reactive controller guaranteeing bounds on velocities and accelerations and enforcing a hierarchical structure of constraints. In contrast to [18], in which the experimental case study is based on a 7 DOFs serial arm, we present a similar technique with specific tasks, constraints and bounds designed for UAMs. In this case, trajectory generation and redundancy resolution are integrated in the same framework. To the knowledge of the authors this is the first work applied to such robots with all computations done in real time, and on board a limited computational unit.

The remainder of this article is structured as follows. In the next Sections we develop the optimization methodology. It includes the formulation of the proposed approach and the specific design of tasks, constraints and bounds for UAMs. The feasibility of the proposed method is shown through real robot experiments in Section V. Finally, conclusions are given in Section VI.

II. Optimization-based trajectory generation

The goal of this Section is to provide an optimization method to generate feasible trajectories for all the joints of a quadrotor-arm system. As quadrotor vehicles are under-actuated systems (*i.e.*, 4 actuations and 6 DOFs), roll and pitch variables are used to control the translational velocities. Moreover, UAMs are not meant for acrobatic maneuvers. Hence, we have not included the platform tilt in the trajectory generation algorithm (roll and pitch angles will be assumed negligible in our analysis). Then, the robot is assumed to have n DOFs, $\boldsymbol{\xi} \in \mathbb{R}^n$, $\boldsymbol{\xi} = \begin{bmatrix} {}^w\boldsymbol{p}_b^\top & {}^w\psi_b & \boldsymbol{q}^\top \end{bmatrix}^\top$, namely, the platform position and its yaw orientation in an inertial frame (w) ${}^w\boldsymbol{p}_b \in \mathbb{R}^3$ and ${}^w\psi_b \in \mathbb{R}$, and the angles of the r arm joints $\boldsymbol{q} \in \mathbb{R}^r$. In the first part of this Section, we will assume that the inner control loop of the system can perfectly track the computed references. However, this hypothesis will be removed in Sec. II-C and its implications discussed.

A. Approach

The goal of a trajectory generation algorithm is to command the robot DOFs to accomplish some given tasks while satisfying the system constraints. These tasks and constraints can be generically expressed by

$$\min f_i(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, t) \qquad \text{and} \qquad f_j(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, t) \leq 0\,, \tag{1}$$

where $\min f_i(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, t)$ represents the ith generic task and $f_j(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, t) \leq 0$ stands for the jth constraint. For example, a trajectory tracking task with the arm end effector can be expressed as the minimization of the tracking error norm, $f(\boldsymbol{\xi}, t) = ||\boldsymbol{x}_e^d(t) - \boldsymbol{x}_e(t)||$, where $\boldsymbol{x}_e^d(t)$ and $\boldsymbol{x}_e(t)$ are the desired and actual end effector positions, respectively.

The key idea of this approach is to assign desired dynamics to $f_i(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, t)$ and $f_j(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}, t)$. In fact, by using the Gronwall inequality as in [19], a constraint expressed as $f \leq 0$ is satisfied if

$$\dot{f} \leq -\lambda_1 f, \tag{2}$$

where λ_1 is a positive scalar gain. Notice that it is just a sufficient condition, since the inequality of (2) is more restrictive than the original one.

By applying iteratively the Gronwall inequality on $\dot{f} + \lambda_1 f \leq 0$ from (2), we have

$$\ddot{f} \leq -\lambda_2 \left(\dot{f} + \lambda_1 f\right) - \lambda_1 \dot{f}, \tag{3}$$

where λ_2 is a positive scalar gain. Parameters λ_1 and λ_2 assign the maximum convergence dynamics towards the constraint. Similarly to the constraints, a task expressed by $\min f$, where $f \geq 0$, can be formulated as

$$\min ||\ddot{f} + (\lambda_2 + \lambda_1)\,\dot{f} + \lambda_2\lambda_1 f||^2. \tag{4}$$

Notice that if the cost function in (4) is always kept at its lower bound, the function f converges to 0 with a dynamics assigned by eigenvalues λ_1 and λ_2.

This approach is useful to obtain constraints and cost functions (*i.e.*, tasks) in the acceleration domain, when the original ones are expressed in the position domain. In fact, considering the constraint $f_j(\boldsymbol{\xi}) \leq 0$ depending only on robot joint values, and applying this approach, we end up with

$$A_j \ddot{\boldsymbol{\xi}} \leq u_{Aj}, \tag{5}$$

where

$$A_j = \frac{\partial f_j}{\partial \boldsymbol{\xi}}, \tag{6a}$$

$$u_{Aj} = -\lambda_2\lambda_1 f_j - \left(\dot{\boldsymbol{\xi}}^\top \frac{\partial^2 f_j}{\partial \boldsymbol{\xi}^2} + (\lambda_1 + \lambda_2)\,\frac{\partial f_j}{\partial \boldsymbol{\xi}}\right)\dot{\boldsymbol{\xi}}\,. \tag{6b}$$

On the other hand, when a constraint also depends on joint velocities ($f_j(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \leq 0$) the Gronwall inequality should be applyied only once. While the constraint expression is the same as in (5), in this case the terms A_j and u_{Aj} are computed with

$$A_j = \frac{\partial f_j}{\partial \dot{\boldsymbol{\xi}}}, \quad u_{Aj} = -\lambda_2 f_j - \frac{\partial f_j}{\partial \boldsymbol{\xi}}\dot{\boldsymbol{\xi}}\,. \tag{7}$$

Similarly to the linear formulation obtained for the constraints (see (5)), the cost functions can be defined as

$$\min ||F_i\ddot{\boldsymbol{\xi}} - b_i||^2, \tag{8}$$

where, similarly to the constraints, the computations of F_i and b_i are straightforward.

Notice that, so far the analysis has been performed for scalar functions f_i, such that F_i results in a row vector, however it can be easily extended to multidimensional tasks $\boldsymbol{f}_i$ and Jacobian matrices F_i.

B. Quadratic problem formulation

The formulation of the optimization problem is quite straightforward. The cost function in (8) results in the quadratic form

$$\min_{\boldsymbol{x}} \|F_i\boldsymbol{x} - b_i\|^2 = \min_{\boldsymbol{x}} \left(\boldsymbol{x}^\top F_i^\top F_i\boldsymbol{x} - 2b_i^\top F_i\boldsymbol{x}\right)\,, \tag{9}$$

where the regressor variable $\ddot{\boldsymbol{\xi}}$ has been replaced by $\boldsymbol{x}$ for simplicity.

As we want to minimize different objective functions, two different scenarios are possible. In the case that a strict hierarchy between tasks is required, a hierarchical solver has to be used (*e.g.*, [13]). Alternatively, as in our case, if there is no need for a hierarchy, a weighted sum of the N_T objective functions can be considered with

$$\begin{aligned}\min_{\boldsymbol{x}} \sum_{i=1}^{N_T} G_i &= \min_{\boldsymbol{x}} \sum_{i=1}^{N_T} \frac{w_i}{h_i} \left(\boldsymbol{x}^\top F_i^\top F_i \boldsymbol{x} - 2 b_i^\top F_i \boldsymbol{x} \right) \\ &= \min_{\boldsymbol{x}} \sum_{i=1}^{N_T} \frac{w_i}{h_i} \left(\boldsymbol{x}^\top H_i \boldsymbol{x} + \boldsymbol{m}_i^\top \boldsymbol{x} \right) ,\end{aligned} \tag{10}$$

where w_i and h_i are weights and a normalization factor, respectively. When a weighted sum is employed, it is very important to normalize the objective functions, in order to effectively set the desired weights w_i. Thus, we chose the weights h_i equal to the spectral norm of H_i, which is equal to the square root of the largest eigenvalue of the product matrix $H_i^\top H_i$. Notice that this spectral norm of H_i equals the square of the spectral norm of F_i when the objective function has the form as in (9).

Remark I In order to effectively use the normalization factor h_i, it is beneficial to split the tasks that are not dimensionally coherent. For instance, the end effector error is composed of translational and rotational parts. Then, computing two different factors h_i improves the effectiveness of the normalization.

Remark II The norms of joint velocities and accelerations define two useful cost functions. They assure the convexity of the problem and allow the distribution of the motion on the different joints a priori, by assigning different weights to each joint. When these two cost functions are used together, the weights on joint velocities should be larger by a factor 5 to 10 than the corresponding weights on joint accelerations, in order to obtain a coherent behavior.

Finally, the complete optimization system combines the cost functions and constraints, obtaining a quadratic problem with linear constraints defined as

$$\begin{aligned}\min_{\boldsymbol{x}} G &= \min_{\boldsymbol{x}} \left(\frac{1}{2} \boldsymbol{x}^\top H \boldsymbol{x} + \boldsymbol{m}^\top \boldsymbol{x} \right) \\ &\text{s.t.} \quad \boldsymbol{l}_b \leq \boldsymbol{x} \leq \boldsymbol{u}_b \quad \text{and} \quad \boldsymbol{l}_A \leq A \boldsymbol{x} \leq \boldsymbol{u}_A ,\end{aligned} \tag{11}$$

where $G = \sum G_i$, $H = \sum \frac{w_i}{h_i} H_i$, $\boldsymbol{m} = \sum \frac{w_i}{h_i} \boldsymbol{m}_i$ and $A = \begin{bmatrix} A_1^\top & A_2^\top & \dots & A_{N_C}^\top \end{bmatrix}^\top$ with N_C the number of constraints. $\boldsymbol{l}_b$, $\boldsymbol{u}_b$, $\boldsymbol{l}_A$ and $\boldsymbol{u}_A$ assign lower ($\boldsymbol{l}_x$) and upper ($\boldsymbol{u}_x$) bounds on acceleration and constraints respectively.

Then, we obtain a solution $\boldsymbol{x}_0$ of the system in (11) for every time step k, thus the position and velocity references can be updated, for example, with an Euler integration as

$$\boldsymbol{\xi}(k) = \boldsymbol{\xi}(k-1) + \dot{\boldsymbol{\xi}}(k-1)\,\delta t + \boldsymbol{x}_0 \frac{\delta t^2}{2} \tag{12a}$$

$$\dot{\boldsymbol{\xi}}(k) = \dot{\boldsymbol{\xi}}(k-1) + \boldsymbol{x}_0\,\delta t. \tag{12b}$$

where δt is the time step differential.

It is important to remark that the trajectory generation algorithm can either be computed offline or online. As the quadratic optimization method can be efficiently solved by state of the art algorithms, it is particularly suitable to be computed online even in limited computational units. Online iteration allows to dynamically change the optimization parameters during specific phases of a mission. It can be particularly useful to implement different redundancy resolution strategies, by changing cost function weights, depending on the mission phase or external triggers. This behavior is shown in the experiments Section for an UAM mission.

C. Interface with the control algorithm

So far in this Section we have assumed a perfect tracking of the generated reference trajectory by the inner controller. With a real system, the performance of this closed control loop is not ideal and dynamics between desired ($\boldsymbol{\xi}^d$) and actual ($\boldsymbol{\xi}$) values of the robot DOFs are introduced. As a consequence, the parameters λ_1 and λ_2 should be set low enough with respect the control bandwidth such that the time scale separation principle holds, and the reference can be tracked. Alternatively, one can include a simplified model of the closed loop system in the dynamics considered by the optimization. Such dynamics analysis is out of the paper scope. In addition, if the actual measures are fed into the optimization algorithm (feedback scheme), feasibility and stability issues can arise. In fact, using this feedback requires techniques of robust optimization, as in [20], to prevent unfeasible points. Moreover, the closed loop control can affect the stability of the system, and the search for stability bounds and conditions can be complicated by the nonlinearity of the optimization method. For the previous reasons, in the rest of the paper the trajectory generation algorithm, as it is common in all robotic trajectory generation systems, is computed without any feedback of joints measures, and parameters λ_1 and λ_2 are chosen according to the inner control bandwidth and time step δt.

III. UAM TASKS

We can consider several objective functions for UAMs, in order to accomplish a given task or to preserve stability. In this Section we present a number of useful cost functions with the corresponding methods to compute the H matrix and $\boldsymbol{m}$ vector. The weighting factors will not be mentioned because they depend on the particular navigation phase and mission objectives, as explained in Section V.

A. End effector tracking

The main interaction task with UAMs, in general, will be executed by the arm end effector, thus it is important to be able to track a desired end effector trajectory, $\boldsymbol{x}_e^d \in \mathbb{R}^6$. The following cost function can be defined

$$G_1 = \|\boldsymbol{e}_e\|^2 , \tag{13}$$

where $\boldsymbol{e}_e = \boldsymbol{x}_e(t) - \boldsymbol{x}_e^d(t)$ is the tracking error, which can be written in acceleration form, following the procedure described in Sec. II-A, as

$$F_1 = 2 \boldsymbol{e}_e^\top J_e , \tag{14}$$

with $J_e \in \mathbb{R}^{6\times n}$ the end effector Jacobian matrix. The expression of b_1 can be easily obtained and is here omitted for the sake of conciseness. Notice that, if the end effector velocity reference is available, it is included in the b_1 term, providing an equivalent feed-forward action.

As pointed out in Sec. II-B, normalization factors for the end effector task should be computed separately for the translational and rotational parts of the Jacobian.

B. Robot center of gravity alignment

When the arm center of gravity (COG) is not coincident with the gravitational vector of the platform, undesired torques appear in the vehicle's base, which must be compensated with the action of the propellers. In order to minimize this actuation effort and avoid instability, it is beneficial to design a task to favor this alignment. Thus, we can easily compute the cost function G_2 and the related Jacobian F_2 as

$$G_2 = {}^b\boldsymbol{x}_c^\top {}^b\boldsymbol{x}_c\,, \qquad \text{and} \qquad F_2 = 2\,{}^b\boldsymbol{x}_c^\top {}^bJ_c\,, \tag{15}$$

where ${}^b\boldsymbol{x}_c \in \mathbb{R}^2$ is the displacement of the center of gravity in the horizontal plane, expressed in the quadrotor body frame, and ${}^bJ_c \in \mathbb{R}^{2\times n}$ is its corresponding Jacobian. This expressions can be obtained as in [6], [7].

Notice that we are assuming that the quadrotor is internally balanced. Otherwise, a different equilibrium point should be assigned for the arm COG.

C. Forces on quadrotor horizontal plane

Limiting the forces exerted by the robotic arm on the quadrotor horizontal plane is beneficial because the vehicle cannot oppose them due to its under-actuation. Thus, we can consider the following objective function

$$G_3 = \frac{1}{2}\ddot{\boldsymbol{\xi}}^\top {}^bJ_c^\top {}^bJ_c\,\ddot{\boldsymbol{\xi}} + \frac{1}{2}\dot{\boldsymbol{\xi}}^\top {}^b\dot{J}_c^\top {}^b\dot{J}_c\,\dot{\boldsymbol{\xi}} + \ddot{\boldsymbol{\xi}}^\top {}^bJ_c^\top {}^b\dot{J}_c\,\dot{\boldsymbol{\xi}}. \tag{16}$$

In this case, we can compute directly the terms H_3 and $\boldsymbol{m}_3$ from the expression in (16). This cost function penalizes the inertial forces exerted by the arm, by considering the acceleration of its center of mass.

D. Limiting quadrotor accelerations

When rapid end effector motions are required, it is better to distribute the motion in the arm joints instead of the platform. This goal is achieved by penalizing quadrotor accelerations with

$$G_4 = {}^w\ddot{\boldsymbol{p}}_b^\top {}^w\ddot{\boldsymbol{p}}_b\,. \tag{17}$$

In fact, quadrotor accelerations in the horizontal plane are obtained through platform tilting, which can potentially affect other tasks if it is not compensated by the inner control loop. In this case, the matrix H_4 turns out to have an identity matrix in the 3×3 upper left block, while $\boldsymbol{m}_4$ is null.

E. Manipulability

During a manipulation task, a useful objective function is represented by the arm manipulability index. A first direct way of expressing this cost function is

$$G_5 = \frac{1}{\det\left(J_{e,q}J_{e,q}^\top\right)}\,, \tag{18}$$

where $J_{e,q} \in \mathbb{R}^{6\times r}$ is the submatrix of the Jacobian J_e composed by the columns corresponding to the arm joints.

Notice that it is almost impossible to analytically compute the required matrices to apply this objective function to a 6 DOFs robotic arm. Instead, we could compute it for a reduced arm (*e.g.*, a 4 DOFs robot) applied to the translational part of the end effector, which effectively allows to obtain a smooth behavior of the system, keeping the robot configurations far from Jacobian singularities. An alternative formulation of the cost function can be obtained by applying the gradient based method of [21].

F. Desired joint position

We can choose a set of desired arm joint positions $\boldsymbol{q}^d \in \mathbb{R}^r$ to achieve some favorable conditions such as distance from joint limits, a high manipulability or a limited center of gravity displacement. This configuration can be set as a reference during the navigation phase, or just used as an attractor to guide the optimization solution when the main tasks present a nonempty null space. This goal can be achieved by applying the cost function defined by

$$G_6 = \sum_{i=1}^{r}\left(\frac{q_i - q_i^d}{q_{M,i} - q_{m,i}}\right)^{2p},$$

where $q_{M,i}$ and $q_{m,i}$ are upper and lower limits of the ith joint, while factor $p \in \mathbb{N}$ is used to modulate the behavior of the cost function between the limits. Notice that, this cost function would be just preserving the joint bounds if infinite values are assigned to the p factor and the weight w_6.

G. Velocity minimization

We can apply different weights to the joint velocities in order to arbitrarily distribute the motion on the UAM joints with

$$G_{7,i} = \dot{\xi}_i^{\,2}\,. \tag{19}$$

Here, the i subscript is used to remark that distinct weights are assigned to the n velocity cost functions, in order to achieve the desired behavior. This cost function is useful to assure the convexity of the problem.

IV. UAM CONSTRAINTS AND BOUNDS

Important constraints, specific for aerial manipulators, are described in the following.

A. Self-collision avoidance

With the arm mounted underneath the aerial platform, the end effector is potentially subject to collisions with the vehicle undercarriage, thus the first obvious constraint is that self-collisions have to be avoided. Modeling the leg l of the quadrotor as a line described by two points $\boldsymbol{x}_{l1}$ and $\boldsymbol{x}_{l2}$, the distance d_l between the leg and a point of the arm $\boldsymbol{x}$ can be computed with

$$d_l = ||D_l \left(\boldsymbol{x}_{l1} - \boldsymbol{x}\right)||\,, \tag{20}$$

where $D_l = \left(I - \boldsymbol{n}_l \boldsymbol{n}_l^\top\right)$ with $\boldsymbol{n}_l = \frac{(\boldsymbol{x}_{l2} - \boldsymbol{x}_{l1})}{||\boldsymbol{x}_{l2} - \boldsymbol{x}_{l1}||}$. Notice how the matrix D_l only depends on the position of the quadrotor legs. Then, the following constraint can be enforced

$$d_l^2 \geq d_{\text{min}}^2\,, \tag{21}$$

where d_{min} is the minimum acceptable distance between legs and the arm point.

Considering $\boldsymbol{x}$ the arm end effector position with respect to the quadrotor base, we have that $\boldsymbol{x} = {}^b J_{e,q} \dot{\boldsymbol{q}}$, where ${}^b J_{e,q}$ is the submatrix of the translational Jacobian composed only by the columns corresponding to the arm joints. This constraint can be expressed in velocity form as

$$2(\boldsymbol{x}_{l1} - \boldsymbol{x})^\top D_l \, {}^b J_{e,q} \, \dot{\boldsymbol{q}} \leq -\lambda_1 \left(d_l^2 - d_{min}^2\right) \tag{22}$$

from which $A_1 = 2(\boldsymbol{x}_{l1} - \boldsymbol{x})^\top D_l \, {}^b J_{e,q}$, and it is straightforward to compute the constraint in acceleration form.

Note that this constraint can easily be adapted to avoid any external obstacle with cylindrical shape.

B. Obstacle avoidance

A simpler condition is to avoid an obstacle described as a sphere. Given the obstacle center point $\boldsymbol{x}_o \in \mathbb{R}^3$ and a point of the UAM (*e.g.*, end effector or quadrotor positions) $\boldsymbol{x} \in \mathbb{R}^3$, we can define the inequality

$$(\boldsymbol{x}_o - \boldsymbol{x})^\top (\boldsymbol{x}_o - \boldsymbol{x}) \geq d_{min}^2\,. \tag{23}$$

It results that $A_2 = 2(\boldsymbol{x}_o - \boldsymbol{x})^\top J$, where $J \in \mathbb{R}^{3\times n}$ is the Jacobian of the considered point $\boldsymbol{x}$.

C. Position, velocity and acceleration bounds

The position, velocity and acceleration limits of the UAM joints have to be included in the optimization problem. For instance, the quadrotor height DOF has an obvious lower bound, while arm joint bounds are given by the physical arm structure. The bounds have to be reported in acceleration form with the strategy presented in Section II. Given an upper bound $\xi_{M,i}$ on the ith joint position, the Gronwall inequality can be applied twice, obtaining the following position bound in acceleration form

$$\ddot{\xi}_i \leq -\left(\lambda_1 + \lambda_2\right)\dot{\xi}_i - \lambda_1\lambda_2\left(\xi_i - \xi_{M,i}\right)\,. \tag{24}$$

On the other hand, when an upper bound $\dot{\xi}_{M,i}$ is given on the ith joint velocity, the Gronwall inequality has to be applied just once, resulting in the following velocity bound

$$\ddot{\xi}_i \leq -\lambda_2\left(\dot{\xi}_i - \dot{\xi}_{M,i}\right)\,. \tag{25}$$

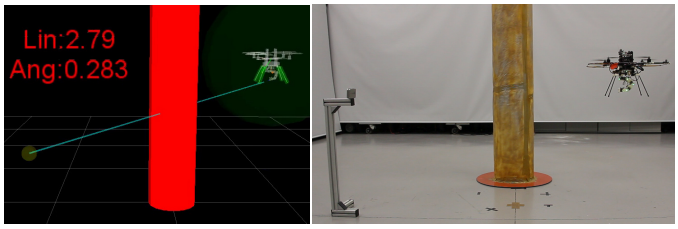


Fig. 1: Experiment setup with the flying arena and a cylinder pilar used as obstacle (the left frame corresponds to the telemetry visualization).

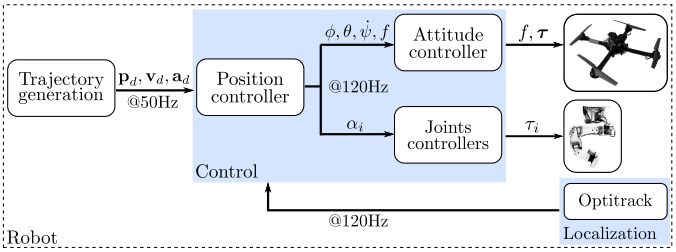


Fig. 2: Overview of the architecture pipeline for trajectory generation and UAM control with all algorithms running on board.

The vectors of lower and upper bounds ($\boldsymbol{l}_b$ and $\boldsymbol{u}_b$) are formed considering the most stringent conditions between position, velocity and acceleration bounds, for each joint.

V. Experiments

The proposed trajectory generation algorithm is implemented on a real UAM, and the effectiveness of the approach is demonstrated by performing an autonomous mission, by accomplishing different tasks while enforcing system constraints. Notice that we do not compare our performance against other methods because, to the authors knowledge, this is the first work using an optimization based approach to achieve such trajectory generation for UAMs with all algorithms running on board in real time. Although simulations using Matlab, Gazebo and ROS have been performed, their results are here avoided for the sake of conciseness. These simulations together with some of the lab experiments presented hereafter are also shown in the accompanying video.

The quadrotor used in the experiments is the ASCTEC Pelican research platform shown in Fig. 1 and its control architecture is shown in Fig. 2. This platform has a position controller in cascade with an off-the-shelf built-in autopilot for attitude estimation and control, which are not the focus of this paper. As for the robotic arm, we take advantage of a 6 DOFs really light structure with a kinematic design suitable to compensate the possible noise existing in the quadrotor positioning while hovering, and to avoid self collisions during take off and landing maneuvers. The arm design is a result of a trade-off between accuracy and payload, leading to a weight of $200g$ including batteries . The complete UAM shows an accuracy of about $0.2rad$ and $0.05m$. Each joint has its own proportional-integral-derivative controller. The Denavit-Hartenberg parameters of the arm are reported in Tab. I. The arm base is $10mm$ below the body frame along the vertical axis. All algorithms are running on board in real-time using an Intel Atom CPU (@1.6GHz) with Ubuntu 14.04LTS and ROS Indigo. The optimized high-rate C++ implementation takes

Joint	θ (rad)	d (m)	α (rad)	a (m)
1	q_1	0	0	0
2	$q_2 - \pi/2$	0	-$\pi/2$	0
3	$q_3 - \pi/2$	0	-$\pi/2$	0
4	q_4	0	0	0.065
5	$q_5 + \pi/2$	0.065	0	0.065
6	q_6	0	$\pi/2$	0

TABLE I: Denavit-Hartenberg parameters of our 6 DOF arm.

	Tasks				
	EE	COG	DesJPos	MinVel	
				Quad.	Arm
Navigation	1	10^1	10^2	10^{-5}	10^{-5}
Interaction	1	10^{-2}	10^{-2}	10^3	10^{-5}

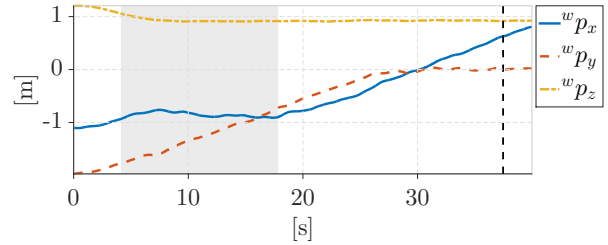
TABLE II: Tasks weightings depending on mission phases.

advantage of QPoases as quadratic programming solver [22] and is available upon request. All experiments have been performed at Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, equipped with an Optitrack motion capture system running at $120Hz$ and used in the low-level quadrotor position control. Notice how the trajectory is generated without positioning feedback, hence with a less accurate localization system (*e.g.*, visual odometry) only the robot tracking performance would be affected. In all the experiments the quadrotor is autonomously taken off and landed, and both maneuvers are considered out of the paper scope. Fig. 1 and the accompanying video show the experiment setup with the flying arena and the cylindrical pilar used as obstacle.

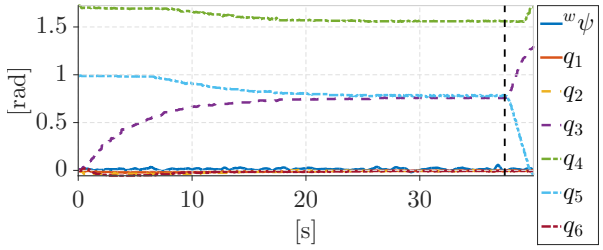
In the real platform, a subset of cost functions and constraints of Sections III and IV has been implemented. In particular, the main task of the mission consists in the end effector trajectory tracking, while complementary tasks used to solve the system redundancy are the alignment of arm COG, the positioning of arm joints to a favorable configuration, and the minimization of joints velocity. The constraint avoiding self-collision between robot end effector and quadrotor legs is active for all the duration of the experiment. In addition, the obstacle avoidance constraint is also tested. In particular, an experiment without any obstacle in the path and another one with a cylindrical shape obstacle will be compared. Finally, position, velocity and acceleration of all DOFs are subject to user selected bounds.

Therefore, we define a desired waypoint for the end effector positioning task (3D position plus 3D orientation), which will drive the whole robot. The waypoint presents a displacement of $2m$ in both x and y directions, and $0.2m$ in z direction. The mission is considered achieved when the linear and angular positioning errors of the end effector are below certain thresholds. These thresholds are selected considering the hardware characteristics previously mentioned, which are $0.05m$ and $0.2rad$ of linear and angular Euclidean errors.

In order to show the effects of the different tasks and constraints, we split the mission in two phases, navigation and interaction. In the first phase, the navigation towards the waypoint has to be preferably performed with quadrotor degrees of freedom, while arm joints should assume a configuration



(a) 3D position of the quadrotor.



(b) Quadrotor yaw angle and arm joint values.

Fig. 3: Values of the robot DOFs during a real experiment with an obstacle inline between the initial and desired end effector positions. The gray region in figure (a) corresponds to the activation of the obstacle avoidance constraint. Notice how the x axis (continuous blue line) is blocked. The vertical dashed line indicates the point where the weights of the arm joints positioning and arm COG alignment tasks are reduced, and quadrotor motion is penalized.

in order to maximize stability and minimize disturbances, *e.g.*, torques produced by displacement of the arm COG. In the interaction phase, when the robot is close to the desired waypoint (*i.e.*, almost hovering for close manipulation), it is preferable to perform the motion with arm joints, because more accurate movements are needed and a minimum safety distance with the interaction object can be kept.

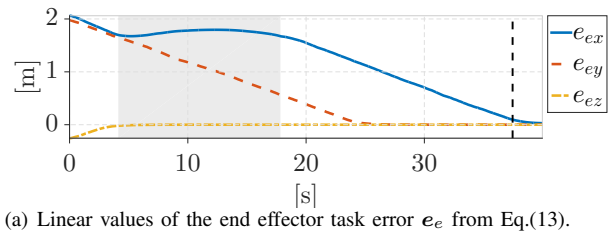
The easiness of the proposed method allows to distinguish the different phases by dynamically changing the weights of the different tasks. The proposed normalization procedure allows to effectively assign a relative priority to the tasks by means of the weights, even with tasks of nonhomogeneous dimensions. All these weights are summarized in Tab. II. In the subsequent figures, the transition between navigation and close interaction phases is shown with a black vertical dashed line.

As presented in Section II, the dynamics of the joints, tasks and the approach to constraints is governed by parameters λ_1 and λ_2. For the described system, they have been chosen equal to $0.8\frac{1}{s}$ and $4\frac{1}{s}$, respectively, in accordance with lower-level control loop bandwidth.

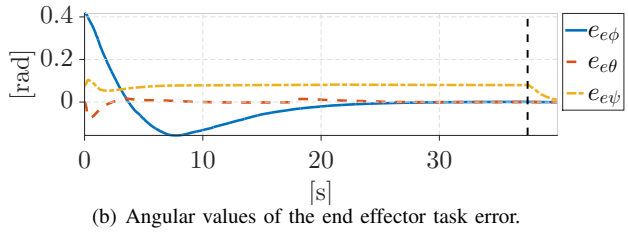
In Fig. 3 the behavior of all UAM joints during the complete experiment with the obstacle is reported. Fig. 4 shows the task errors during the mission, whereas in Fig 5 the quadrotor trajectories (x and y plot) are shown with and without an obstacle lying in the middle of the shortest path. The different motion profiles are discussed in the following.

A. Navigation phase

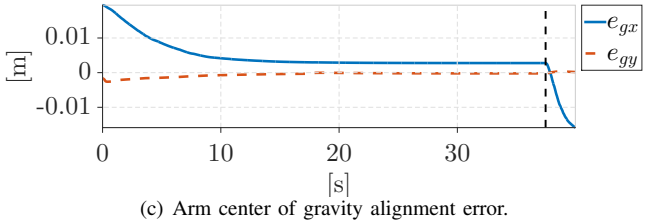
During the navigation, large weights are assigned to the cost functions of COG alignment and desired joint positions. Thus, long displacements are performed by the quadrotor and



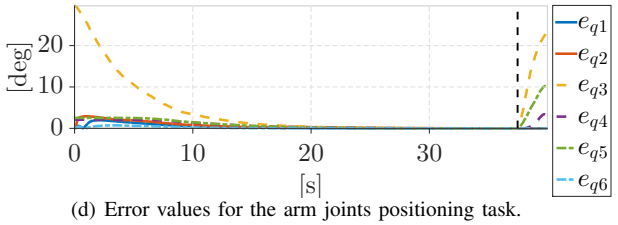
(a) Linear values of the end effector task error $\boldsymbol{e}_e$ from Eq.(13).



(b) Angular values of the end effector task error.



(c) Arm center of gravity alignment error.



(d) Error values for the arm joints positioning task.

Fig. 4: Task errors corresponding to the real experiment of Fig. 3 with an obstacle between the initial and desired end-efector positions. The gray region in frame (a) corresponds to the activation of the obstacle avoidance constraint. Notice how the error in the x axis (continuous blue line) is increased to avoid the obstacle. The vertical dashed line indicates the point where the weights of the arm joints positioning and arm COG alignment tasks are reduced in favor of the end effector positioning task.

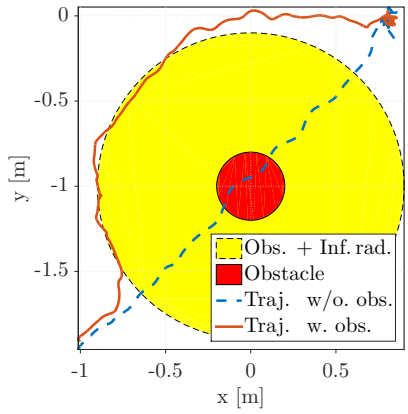


Fig. 5: Comparison between two real trajectories with the same initial and final positions, but with (continuous brown line) and without (dashed blue line) any obstacle lying between the initial and desired end effector positions. The small red circle corresponds to the actual obstacle and the yellow area (yellow circle with a dashed edge) includes the inflation radius applied to the obstacle.

the arm is driven to a desired position while minimizing COG misalignment. These profiles are reported in Fig. 3. The behavior of the end effector task error is reported in Fig. 4(a) and Fig. 4(b), for the translational and rotational parts, respectively. In Fig. 4(c) and 4(d), the profiles of the COG alignment and the arm joints positioning can be analyzed. Notice that the COG task is not completely accomplished, *i.e.*, the task error is not reduced to zero, because the latter has larger weight, and the equilibrium solution is a weighted average between the two goals.

The arm configuration is shown in Fig. 6, where frame (a) is just after the take off, and frame (b) represents the robot configuration during navigation where the COG alignment and arm joints positioning tasks prevail. Frame (c) represents the arm configuration during the interaction phase, that will be described in the following.

To show the need for the collision avoidance constraint we added an obstacle in the middle of the trajectory (*i.e.*, shortest path). The quadratic programming solver generates a feasible trajectory, and the UAM avoids the obstacle with the minimum assigned distance of $0.6m$ (inflation radius around the platform). To clearly see how the obstacle avoidance works, we show in Fig. 5 a comparison between two trajectories with the same parameters. A first trajectory is executed without any obstacle (blue dashed line) and the computed path corresponds to the shortest path as expected. When an obstacle appears (defined by the red circle) the trajectory is modified to avoid it (brown continuous line), satisfying the inflation radius constraint (yellow area plotted, in this case, around the obstacle).

This behavior is evident in the gray area of Fig. 3(a), where the motion of the platform is prevented for the x axis (continuous blue line). Once the obstacle has been avoided, the trajectory is resumed.

As it can be seen in Fig. 3, the quadrotor motion is clearly performed with constant velocity. In fact, for large end effector errors, the maximum velocity bound is saturated and the quadrotor moves with constant velocity. Notice that the two curves of x and y quadrotor positions have the same slope, because the same velocity bound has been assigned. On the other hand, the z coordinate presents a smaller initial error, thus the velocity is not saturated and, as a result, the behavior is exponential because its dynamics is governed by parameter λ_1. Notice that the z position reference is reached after $6s$ and $7s$, while the theoretical settling time $\frac{5}{\lambda_1}$ is equal to $6.25s$.

B. Interaction phase

The second phase starts when the end effector is $15cm$ far from the desired position. At this point, weights are changed to the values of Tab. II. Notice how the arm joints move towards the goal, see Fig. 6(c) that represents the robot configuration during the interaction phase, and Fig. 3, which reports the behavior of the joint variables. As a consequence, COG and joint positioning task errors are increasing, as reported in Fig 4(c) and Fig 4(d). The end effector task is performed to a greater extent by robot arm joints, because the weights of the cost functions corresponding to the joint velocity minimization

(a) (b) (c)

Fig. 6: Samples of arm configurations depending on tasks weightings during a real experiment: (a) After taking off, (b) during navigation and (c) in the close interaction zone.

are assigned such that quadrotor motions are more heavily penalized than arm ones. Notice that the trajectory is computed using the acceleration as a regressor and applying bounds to it. For this reason, the transition between the two phases is smooth, without discontinuities.

As a conclusive remark, our UAM can effectively accomplish the mission in the two distinct phases, avoiding an obstacle and self-collisions and respecting variable bounds.

VI. Conclusions

In this work, we presented a trajectory generation method for aerial manipulators using a quadratic programming approach and showed its execution on a real platform. The method uses an online active set strategy to compute feasible joints acceleration references, in order to accomplish given tasks, while enforcing constraints and variables bounds. A number of tasks and constraints specific to unmanned aerial manipulators have been integrated in the algorithm. A weighting factor, associated with a normalization procedure, allows to define the relative importance of tasks, determining the redundancy resolution. This is particularly important to effectively perform distinct phases of an articulated mission. The viability of the proposed approach is demonstrated through simulations and real robot experiments. In particular, the objective functions implemented in the real setup include the end effector positioning task, the alignment of the arm COG with the platform gravitational vector and the positioning of arm joints to a favorable configuration. In addition, we demonstrated that the robot can avoid collisions with known obstacles in the scene and self-collision between end effector and quadrotor legs. To the authors knowledge, this is the first work using an optimization based approach to compute trajectories for aerial robots with high number of DOFs, working onboard and in realtime.

References

[1] C. M. Korpela, T. W. Danko, and P. Y. Oh, "MM-UAV: mobile manipulating unmanned aerial vehicle," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1, pp. 93–101, 2012.

[2] M. Fumagalli, R. Naldi, A. Macchelli, F. Forte, A. Q. L. Keemink, S. Stramigioli, R. Carloni, and L. Marconi, "Developing an aerial manipulator prototype: physical interaction with the environment," *IEEE Robotics Automation Magazine*, vol. 21, no. 3, pp. 41–50, Sept 2014.

[3] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: safe and efficient load manipulation with aerial robots," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 69–79, Sept 2012.

[4] V. Lippiello and F. Ruggiero, "Exploiting redundancy in Cartesian impedance control of UAVs equipped with a robotic arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 3768–3773.

[5] I. Palunko and R. Fierro, "Adaptive control of a quadrotor with dynamic changes in the center of gravity," *18th IFAC World Congress*, vol. 18, no. 1, pp. 2626–2631, 2011.

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

[7] V. Lipiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Esteves, and A. Viguria, "Hybrid visual servoing with hierarchical task composition for aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 259–266, Jan 2016.

[8] M. Hehn and R. D'Andrea, "Quadrocopter trajectory generation and control," *18th IFAC World Congress*, vol. 44, no. 1, pp. 1485–1491, 2011.

[9] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[10] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load-A differentially-flat hybrid system," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 4888–4895.

[11] G. Garimella and M. Kobilarov, "Towards model-predictive control for aerial pick-and-place," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 4692–4697.

[12] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," *arXiv preprint arXiv:1602.01949*, 2016.

[13] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *International Journal of Robotics Research*, 2014.

[14] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *IEEE International Conference on Robotics and Automation*, May 2014, pp. 2589–2594.

[15] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[16] A. Potschka, C. Kirches, H. G. Bock, and J. P. Schlöder, "Reliable solution of convex quadratic programs with parametric active set methods," *Interdisciplinary Center for Scientific Computing, Heidelberg University, Im Neuenheimer Feld*, vol. 368, p. 69120, 2010.

[17] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 7681–7687.

[18] A. M. Zanchettin and P. Rocco, "Reactive motion planning and control for compliant and constraint-based task execution," in *IEEE International Conference on Robotics and Automation*, May 2015, pp. 2748–2753.

[19] R. Bellman and K. L. Cooke, "Differential-difference equations," *Academic Press, New York*, 1963.

[20] A. M. Zanchettin and P. Rocco, "Robust constraint-based control of robot manipulators: an application to a visual aided grasping task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2016.

[21] Y. Zhang, D. Guo, K. Li, and J. Li, "Manipulability-maximizing self-motion planning and control of redundant manipulators with experimental validation," in *IEEE International Conference on Mechatronics and Automation*, Aug 2012, pp. 1829–1834.

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