POLITECNICO
MILANO 1863

POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

CONTROL OF AN UNMANNED AERIAL VEHICLE EQUIPPED WITH A ROBOTIC ARM

Doctoral Dissertation of:
Roberto Rossi

Supervisor:
Prof. Paolo Rocco

Tutor:
Prof. Luca Bascetta

The Chair of the Doctoral Program:
Prof. Andrea Bonarini

Year 2016 – Cycle XXIX

A mio padre, mia madre, mia sorella,
e a Mireia.

The good life is one inspired by love
and guided by knowledge

B. Russell

Acknowledgments

MY sincere gratitude goes first and foremost to my supervisor Professor Paolo Rocco, for his precious scientific advice and human support. His professional scrupulousness, wisdom and helpfulness have been a fundamental example for me. Moreover, his encouraging and reassuring attitude, together with his confidence on me as a person and a researcher, represented a great support to me.

I would also to thank Professor Luca Bascetta and Professor Andrea Zanchettin for their fruitful collaboration and mentoring.

I am thankful to Professor Juan Andrade Cetto and Angel Santamaria Navarro, for giving me the opportunity of developing part of my research in the laboratory of the Institut de Robòtica i Informàtica Industrial in Barcelona, where I experienced a beneficial exchange of knowledges and skills.

I would like to acknowledge Professor Lorenzo Marconi and Professor Fabrizio Caccavale for reviewing my PhD thesis and for their constructive comments and suggestions. Their inputs have been essential to add value to the finalized version of this thesis.

I would like to thank the friends and PhD fellows of the robotic office, Nicola, Giovanni, Matteo Roger, Matteo Peris and Davide, and of the Environmental group, Andrea, Francesca, Simona and all the others, for the stimulating and interesting discussions and the *hygge* time together.

I wish to acknowledge the Master students I supervised, and in particular Luca, Alberto, Mattia, Stefano, Giulio, Dario and Mirko, for the valuable contribution provided to my research. In addition, I want to thank Vasco for his precious help, who solved a long list of problems I coped with.

My PhD is not entirely confined in the contents of this thesis and of my research papers. It is made by the challenges I tackled, by the experiences I lived, and most of all, by the people I met, who enriched me as a person.

My deepest gratitude goes to these persons, to my friends and, especially, to the ones who made this journey possible and delightful, my family and Mireia.

Abstract

ROBOTS ability to autonomously move in the terrain, underwater and aerial domains opened a large number of new applications in industry, service robotics and search and rescue activities. In particular, the impressive growth of aerial robotics during the last decade paved the way to the birth and subsequent development of the new research field of aerial manipulation. The improvement of payload capacity and dexterity of aerial vehicles, in conjunction with advances in the design of light-weight arms, enabled the conception of flying robots equipped with articulated manipulators. Although first demonstrations of aerial manipulation activities are promising, this research field is still characterized by low maturity, and several open research challenges in the areas of perception, control and planning require further investigation.

The present thesis aims at increasing the degree of autonomy and improving the control performance of an aerial robot composed of a quadrotor equipped with a robotic manipulator underneath. To this end, a stability control strategy which tackles the coupling dynamics of the underactuated flying base with the robotic arm is developed. Moreover, the intrinsic kinematic redundancy of the system is exploited with two different optimization-based trajectory generation methods, in order to accomplish prescribed tasks while satisfying given constraints. Finally, contributions in the fields of the state estimation of the aerial systems and the control of interaction with the external environment are provided.

Contents

CHAPTER 1

Introduction

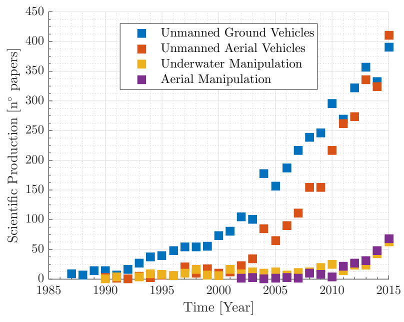
1.1 Context and Motivations

Nowadays, robotics technologies have a relevant and progressively growing impact in the manufacturing sector and they are widespread in several other fields, mentioning the medical, military, space and logistics applications as the most important examples. Nevertheless, the pervasiveness and the degree of impact of robotics in human activities is limited by the circumscribed physical workspace reachable by robots with the current technology. In the last two decades, efforts of researchers have been addressed to extend this reachable workspace, enabling robots autonomous navigation on ground surfaces, as well as in underwater and aerial environment.

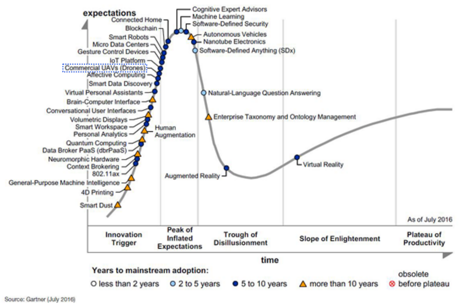
Autonomous ground and aerial vehicles have been leading trends in robotic research for the last decade. In Fig. 1.1(a), the scientific production[1] relative to mobile robots is represented. Additional impulse to ground, underwater, and aerial mobile robotics research has been motivated by the humanitarian and environmental disasters such as the World Trade Center collapse in the United States in 2001, after which the IEEE RAS[2] Technical

[1]Data represent the number of published paper, indexed on www.scopus.com, obtained applying suitable filters in the "Article Title, Abstract and Keywords" search field.

[2]IEEE RAS stands for the Robotic and Automation Society of the Institute of Electrical and Electronics



(a) Scientific Production



(b) Hype Cycle for Emerging technologies

Figure 1.1: *The left-hand side figure represents the scientific production, in terms of published number of papers, in the field of mobile robotics, and mobile manipulation. At the right-hand side the 2016 Hype cycle proposed by Gartner Inc. is reported.*

Committee on Safety, Security, and Rescue was established, the Deepwater Horizon Oil Spill in 2010, and the Great Eastern Japan Earthquake of 2011. The mentioned events, and the market demand for a number of industrial applications, lead to envision aerial and underwater systems that could perform tasks requiring physical interaction with the environment, originating the aerial and underwater manipulation research fields. However, the research activities on aerial manipulator systems should be considered under the wider perspective of the aerial robotics movement and its peculiar characteristics.

The growth rate of aerial robotics both in academic research and commercial products has been particularly impressive. The U.S. Federal Aviation Administration (FAA) reports that about 2.5 million of small Unmanned Aerial Vehicles (UAVs) have been sold in 2016 in the U.S., and this figure is expected to grow to 7.0 million in 2020. In fact, the 2016 Hype cycle for emerging technologies[3], reported in Fig. 1.1(b), shows that commercial UAVs have just passed the innovation trigger phase.

The main professional applications for aerial robots are in the field of agriculture, infrastructure inspection, photography and film production. A relevant aspect concerning the data reported by the U.S. FAA is the prevailing share of small UAVs sold as hobbyist models, the 76%, with respect to the commercial non-model aircrafts share. This phenomenon is strictly connected with the open-source movement (see, for instance, [64]), which is particularly active in the development and sharing of hardware and soft-

Engineers organization.

[3]Source Gartner, Inc. www.gartner.com.

ware designs for UAVs, and fosters the growth of the DIY[4] and makers community.

The mentioned characteristics of the overall aerial robotics movement have had a tremendous impact on academic research. The motivations provided by the public and private interests in the field, merged with the availability of low-cost hardware and shared algorithms, have enabled a prolific production of theoretical and technological advances in academia, which contributes and enhances the expansion of the field.

Today, UAVs platforms range from very big aerial vehicles with tens of meters wingspan to micro and even nano vehicles with grams of weight and millimeters wing-span. Aerial robots are now using new navigation sensors, actuators, embedded control and communication components, including miniature components with significant capabilities. Thus, in addition to electrical helicopters and fixed wing aircrafts with a higher payload, multi-rotor systems (quadcopters, hexacopters or octocopters) have been developed. In many applications rotary wing aerial platforms are preferred due to vertical takeoff and landing and hovering capabilities, and a greater dexterity compared with fixed wing vehicles.

Aerial robotics activities still require the development of new algorithms and technologies, for accurate positioning and trajectory tracking. The enhancement of perception and planning capabilities, control robustness, flight and payload limitations and the reliability of hardware and software subsystems are among the most relevant open research problems.

In this context, characterized by growing market demand and academic research interest, aerial robotic manipulation emerged as a new promising field. It is hard to define this novel research field as either technology pushed or market pull. The first trials of aerial manipulation experiments are less recent than expected. For instance, in 1939 the Stinson Reliant SR10 aircraft was employed to test a unique airmail service for communities that did not have landing fields. Later, in the 1950s the Fulton surface-to-air recovery system was used for retrieving persons on the ground without the need of landing. The technologies developed in the recent years, as the conceiving of Vertical Take-Off and Landing (VTOL) vehicles, more accurate positioning systems and miniaturized components, enable the possibility to provide physical interacting capabilities to aerial robots.

[4]DIY stands for Do It Yourself, which describes behaviors where "individuals engage raw and semi-raw materials and component parts to produce, transform, or reconstruct material possessions, including those drawn from the natural environment (landscaping)" [107].

The versatility and the degrees of freedom of the aerial robots, provided by the aerial platform and the manipulator device, open many new applications in different fields. The possibility to perform manipulation tasks in otherwise inaccessible, or very costly to access, sites empowers the accomplishment of activities such as maintenance, transportation, assembly, and construction of structures.

The first examples of interaction of autonomous aerial robots with the environment or other manned vehicles had the goal of increasing the endurance of the missions. For instance, in [72] and [31] the air-air refueling with an UAV, and an helicopter passive perching on cylindrical surfaces were proposed, respectively. Then, slung load transportation has been performed with unmanned helicopters in [16] and quadrotors in [83], while joint transportation with several helicopters was demonstrated in the AWARE project [4], see [17]. Load transportation has direct applicability to the industrial field and in search and rescue activities as performed in the SHERPA project [5], where a first-aid package was transported with a high payload unmanned aircraft. Unmanned Aerial Manipulators (UAMs) can substitute human workers by executing simple tasks in sites that are otherwise reachable by human workers with costly and dangerous methods. For example, in [8] a method to substitute human workers in the cleaning operations of windows or walls with a semi-autonomous aerial vehicle is proposed. More complex manipulation operations have been demonstrated in the framework of the ARCAS project [3]. Assembly and construction of structures are performed with the cooperation of multiple UAVs, employing articulated robotic arms mounted on VTOL aircrafts. Inspection and maintenance operations can represent a relevant industrial application of aerial manipulators, mostly for structures and plants which present sites hardly reachable by workers. Demonstrations of remote inspection by contact are shown in the AIRobots [2] and in the AEROARMS [1] projects.

1.2 Research Challenges

The fascinating idea of substituting humans in dangerous and costly activities with unmanned aerial vehicles, able to safely and effectively interact with the environment, is a realistic scenario that can be achieved in few years. To this extent, the efforts of researchers are addressed to reach the required autonomy of aerial manipulator systems, while granting a wide adaptability to different tasks and dynamic real-world environment. An effective collaboration with humans, in terms of tele-operation and interfaces, and the reliability and safety of the systems are other two main requirements to be satisfied in order to achieve the applicability of aerial manipulator systems to the real world.

Notice that a number of different configurations of UAMs have been proposed in the literature. In the following, the attention will be focused on multi-rotors UAVs equipped with, at least, an articulated arm underneath. The advantage of this configuration is the high flexibility and adaptability to different tasks, thanks to the capability of hovering of the vehicle and the high number of degrees of freedom available, which provide high dexterity to the complete platform. Different design approaches to perform aerial manipulation, for example based on passive compliant tools directly mounted on the aircraft as in [89] and in [12], provide clear advantages in terms of simplicity of design and control, and robustness to compliance, but they are hardly reconfigurable for several different tasks. The design of the aerial manipulators is a quite active open research line. In fact, different design concepts have been investigated, together with the selection and development of the most suitable hardware and sensors for such systems.

One of the most critical research issues aimed at providing autonomy to aerial manipulators is the stability control of the entire system. The stability control of unmanned aerial vehicles with a robotic arm is per se a challenging problem. The forces and torques exchanged at the physical junction between the two subsystems, the flying base and the arm, impose a dynamic coupling that, in most of the cases, cannot be neglected in the model and control strategy. The compensation of these mutual actions is difficult for the typical actuation structure of multi-rotors aerial vehicle. In fact, vehicles as helicopters, quadrotors and similar configuration aircrafts, are under-actuated, thus they present a number of actuators lower than the number of degrees of freedom of the body. As a consequence, the compensation of disturbances is performed with a variation of the vehicle attitude, in order to orient the propellers axes in the suitable direction to provide the required control actions. When a robotic arm is attached to the vehicle base, the 3D rotation of the aircraft is affecting the absolute position of the robot itself, and inertial torques and forces are exchanged between the two systems. As explained in [56], when the two subsystems are controlled independently, without taking into account this interaction, the described coupling dynamics can drive the full system to instability.

The control of an aerial manipulator can be designed applying three different approaches, as reported in [57]. According to a first simpler approach, two decoupled independent controls can be considered, one accounting for the stabilization of the aircraft and one for the robotic arm; nevertheless, the reasons provided above limit the applicability of this approach to systems where the ratio between the vehicle and robotic arm masses is high. In a second approach, a coupling on the kinematic level can be considered. It means that positions and velocities of both subsystems are coupled in the control approach. Thirdly, the dynamic model of the whole system is considered. This last approach can provide the best possible performance, but some practical issues can compromise the potential advantages of the approach. Quality of the sensor signals, communication delays and errors in model system parameters and disturbances, can undermine the stability of model-based control approaches. In Chapter 2, an extensive state of the art on aerial manipulator control approaches will be provided, and a control based on the coupling on the dynamical level will be proposed.

Additional control issues arise when the aerial manipulator comes into contact with the external environment. Aerial vehicles presents in general high sensitivity to contact force, and kinematic constraints added by contact with environment surface can lead to instability. For instance, the helicopters dynamic rollover instability in single-skid ground contact is well-known. Moreover, helicopters and quadrotors can only generate a limited range of control torques, which makes even harder the control of system behavior during physical interaction. Notice that these control problems have to be addressed considering a limited state knowledge, due to the characteristics of commonly employed proprioceptive sensors. Therefore, advances on perception and navigation algorithms and sensing systems are critical enabling factors for the accomplishment of tasks requiring physical interaction with the environment.

Notwithstanding the described control challenges, the configuration composed of a multi-rotor aerial vehicle and an articulated robotic arm is characterized by high dexterity, as mentioned above. Such systems commonly present kinematic redundancy, thus they are characterized by a number of degrees of freedom greater than the main task to be performed. Thus, the additional degrees of freedom can be used to perform secondary tasks, or to maximize some performance criteria as a function of the robot joint configuration. To this extent, a trajectory generation algorithm, capable of solving the kinematic redundancy in order to satisfy some given tasks, takes particular relevance in the field of aerial manipulation. Moreover, the mentioned stability issues and the complexity of the physical structure of the system impose hard constraints to the set of joint configurations which can be taken by the system. For instance, a complete extension of the arm can undermine the controllability of the system, when the resulting gravitational torque is higher than the maximum control torque that can be applied by the flying base. Therefore, an algorithm for generating a suitable joint trajectory is not only recommended in order to exploit the redundancy of the system to perform multiple tasks, but it is also a critical requirement to avoid dangerous system conditions. In the literature, the approaches for kinematic redundancy resolution can be classified in two main categories. A null-space based strategy, proposed in [9], is applied in a number of works in the field of aerial manipulation; the control strategy can manage a hierarchical series of tasks, by computing the joint velocities to accomplish given tasks, on the base of a first order local approximation of the system. A second category of trajectory generation approaches are based on optimal control strategies. In the literature, few examples have been proposed, as in [41], but the potentialities of the approach are promising. A detailed state of the art on trajectory generation algorithms will be presented in Chapter 3.

The work presented in this thesis is in the main stream if the research described so far. As pointed out, aerial manipulation is a novel research field, which presents challenging issues in the areas of control, perception, planning and the overall design of the system. Further open research lines on aerial manipulation include the collaboration with humans, effective teleoperation strategies, and cooperative control of multiple aerial manipulators. These themes have not been detailed because they are not addressed in the present work.

1.3 Thesis Contents and Research Contribution

The present dissertation deals with the control, trajectory generation and state estimation of an aerial manipulator system, composed of a quadrotor equipped with an articulated robotic arm underneath. As pointed out in the previous Section, the selected topics present several open research problems. In the present work, some contributions will be provided, with the ambition to help reducing the present gap that prevents a widespread adoption of flying robotic workers in real world activities.

The discussion presented in the previous Section pointed out the stability control issues for aerial manipulators, whose flying base and robotic arm present comparable masses. In Chapter 2, a control approach addressed to this specific problem will be presented. In particular, the main advantage of such platforms, the dexterity and the redundancy of the system, is exploited in order to compensate the motion of roll and pitch angles, needed for the control of the translational degrees of freedom of the quadrotor, with the redundant variables of the system. A suitable change of variables, entailing the mentioned compensation, is applied, and an inverse dynamics control is proposed. The stability of the control system is demonstrated by means of Lyapunov analysis. Notice that the effectiveness of model-based controls that consider the coupling on the dynamical level, as the present one, can be mitigated by inaccurate sensors information, model errors or communication delays. Nevertheless, the validity of the proposed approach is demonstrated with an extensive simulation campaign, and an improvement of the tracking performance is obtained with respect to a state of the art algorithm. The application of the proposed algorithm can guarantee higher impact on aerial manipulation performance for systems with accurate sensors information, compared to the size of the system itself.

A second contribution concerns the resolution of kinematic redundancy to perform prescribed tasks. Two trajectory generation algorithms based on constrained optimization are presented. To the author knowledge, they represents the first demonstrations of optimal trajectory generation performed on line and on board of an aerial manipulator. Both methods allow to perform a number of prescribed tasks, while satisfying equality and inequality constraints. The first approach is based on a second order local approximation of the system, and the generation of an optimal trajectory is converted into a quadratic programming problem, whose regressor variables are joint accelerations. The second approach applies a non linear model predictive control strategy, where the control action is represented by joint velocities. The differences and advantages between the two methods will be compared and discussed.

In Chapter 4, the suitable sensing system and algorithms needed to obtain an estimate of the state of a small size aerial manipulators are analyzed and described. In this part, the attention is drawn also on technological issues. A specific theoretical contribution consists in a method for the calibration and performance evaluation of a camera, mounted on the quadrotor, employing an industrial robot. It represents a method particularly useful for academic laboratory and industry, where motion capture systems or similar technology are not available.

Finally, a contribution to the control of interaction of aerial manipulators with the external environment is provided. In particular, a method to limit the damages in a potential impact of the aerial manipulator with an external object is proposed. The approach has been conceived in the context of safe human-robot interactions, in order to bound the potential injuries to human workers caused by accidental collision with collaborative robots. The application of the method to the aerial manipulation field is useful when particularly fragile equipments or surfaces are manipulated, whose integrity can be undermined by the aerial robots itself. An index to define the energy dissipated in potential inelastic impacts for a generic manipulator is proposed, which describes the severity of damages occurring as a consequence of a potential impact. Moreover, a constraint-based hierarchical control strategy is applied to perform prescribed tasks while bounding the proposed index under a certain threshold.

1.4 Method and Experimental Setup

Chapters are structured to be quite independent from each other. For each topic, a state of the art is provided and the approach and main theoretical contributions are presented. Contextually, experiments validating the specific topics are shown. It is worth pointing out that the large majority of the performed experimental campaigns have been performed with the facilities of the MEchatronics and Robotics Laboratory for Innovation (MERLIN) of the Politecnico di Milano [7]. A complete aerial manipulator has been designed and developed in this lab to perform the experimental activities. Components and architectures of the system, together with the characterization of relevant parameters are described in Appendix A. Part of the experimental validation of the methods presented in Chapter 3 has been performed employing the facilities of the Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, in Barcelona, Spain [6].

1.5 Publications

The contents of this thesis are based upon the following publications:

1. R. Rossi, P. Rocco. Inverse Dynamics Control for Aerial Manipulation. *Automatica* (submitted).
2. R. Rossi, A. Santamaria Navarro, J. Andrade Cetto, P. Rocco. Trajectory Generation for Unmanned Aerial Manipulators through Quadratic Programming. *IEEE Robotics and Automation Letters*, April 2017.
3. D. Lunni, A. Santamaria Navarro, R. Rossi, P. Rocco, L. Bascetta, J. Andrade Cetto. Non Linear Model Predictive Control for Aerial Manipulation. *IEEE International Conference on Robotics and Automation (ICRA 2017)*, May 2017 (submitted).
4. R. Rossi, G. Melacarne, P. Rocco. Perfomance Evaluation of Visual Odometry using an Industrial Robot as Ground Truth. *42nd Annual*

Conference of the IEEE Industrial Electronics Society (IECON 2016), Oct 2016.

5. R. Rossi, M. Parigi Polverini, A. Zanchettin, P. Rocco. A Pre-Collision Control Strategy for Human-Robot Interaction Based on Dissipated Energy in Potential Inelastic Impacts. *IEEE International Conference on Intelligent Robots and Systems 2015 (IROS 2015)*, Sept 2015. Finalist for the IEEE RAS Best Student Paper Award.

6. R. Rossi, M. Parigi Polverini, A. Zanchettin, P. Rocco. An Energy Based Injury Index for Pre-Collision Safety Control in Human-Robot Interaction. *The 8th International Workshop on Human-Friendly Robotics (HFR 2015)*, Oct 2015.

7. R. Rossi, P. Rocco. Trajectory Generation and Control for Aerial Manipulation. *Convegno SIDRA 2016*, Sept 2016.

Other works published during the author's PhD activity, though not strictly related to this thesis, are listed here:

1. M. Parigi Polverini, R. Rossi, G. Morandi, L. Bascetta, A. M. Zanchettin, P. Rocco. Performance Improvement of Implicit Integral Robot Force Control through Constraint-Based Optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, Oct 2016.

2. R. Rossi, L. Bascetta, P. Rocco. Implicit Force Control for an Industrial Manipulator based on Stiffness Estimation and Compensation during Motion. *IEEE International Conference on Robotics and Automation 2016 (ICRA 2016)*. May 2016.

3. R. Rossi, L.Bascetta, P. Rocco. Implicit Force Control for an Industrial Robot with Flexible Joints and Flexible Links. *IEEE International Conference on Intelligent Robots and Systems 2014 (IROS 2014)*. Sept 2014.

1.6 Thesis Organization

The remainder of the thesis is organized as follows:

In **Chapter 2**, a model-based control strategy, based on a novel variables representation of aerial manipulator systems, will be proposed. The stability of the proposed control strategy is demonstrated with Lyapunov analysis. A simulation campaign is performed, and the effectiveness of the approach along with the performance improvement with respect to a state of the art approach are shown.

Chapter 3 presents two different strategies to generate an optimal trajectory for the joints of aerial manipulators. The first strategy is based on a quadratic programming approach, while the second applies a non linear model predictive control strategy. Experiments validating the two strategies are shown.

In **Chapter 4**, the design of a suitable sensing system for small size aerial manipulators is presented. The selection of sensors and the estimation algorithms are analyzed and discussed. Finally, a method to calibrate and evaluate the performance of a camera, mounted on a quadrotor, employing an industrial robot, is proposed.

In **Chapter 5**, a contribution to the field of the physical interaction of aerial manipulators with external environment is provided. A novel index to predict the severity of the potential damages, based on the dissipated energy in potential inelastic impacts, is presented. A hierarchical quadratic programming approach is employed to bound the proposed damage index under a certain threshold during the motion, preserving the integrity of the manipulated objects.

Finally, **Chapter 6** summarizes the thesis contributions.

Three appendices are reported at the end of the manuscript.

Appendix A presents the design of a complete aerial manipulator and the characterization of relevant parameter.

In **Appendix B**, proofs of propositions of Chapter 2 are reported.

In **Appendix C** the experimental validations of the danger assessment and control approach proposed in Chapter 5 are reported.

CHAPTER 2

Inverse Dynamics Control

This Chapter presents a novel control approach for a system composed of a multi-rotor unmanned aerial vehicle endowed with a robotic arm. Aerial manipulation systems give rise to several control challenges, due to the coupling dynamics of UAV and robotic arm and to the under-actuation of the flying base. A kinematic and dynamic decoupling between quadrotor variables used for control aims, e.g. roll and pitch rotations, and motion variables of the end effector and center of mass of the robotic arm is proposed. An inverse dynamics control that takes into account the under-actuation of the system is developed. The local stability of the control method is proven with Lyapunov analysis. An extensive simulation campaign is conducted in order to demonstrate the system stability in various operating points and show the improvement of control performance with respect to a state of the art algorithm.

2.1 Introduction

First studies on aerial manipulation deal with keeping contact with environment and controlling the interaction dynamics [13, 36, 79, 85, 89, 98], or performing a grasping operation [65, 87, 101, 104]. While these studies

can account for applied external forces, they do not consider the dynamic effects of a moving arm fixed on the UAV.
The main challenges to face in controlling an aerial vehicle equipped with a robotic arm are the displacement of the center of mass, the change of the total inertia of the system, and the coupling dynamics of UAV and robotic arm. Notice that quadrotors are considered as the most suitable UAVs for aerial robotics operations, because of their maneuverability and high payload-mass rate. However, the intrinsic under-actuation of these UAVs entails additional difficulties in the control problem. Studies on the effects of the displacement of the center of mass on the system during flight are reported in [20, 24, 52].

Decentralized control approaches, e.g. where distinct control architectures for UAV and manipulator are used, addressed change of inertial properties of the system during flight by means of adaptive parameters in simple PID control structures [78, 81, 82], backstepping control algorithms [50] or feedback linearization approaches [83, 84]. These control schemes succeed in stabilizing the system when quasi-static changes of the inertial parameters occur, however coupling dynamics between the two systems are not considered.

As a matter of fact, studies in [47, 55, 56] demonstrate that coupling dynamics between an under-actuated UAV and a robotic arm can lead to instability. A centralized control that considers the whole system dynamics is thus needed to address such coupling.
In [11, 47, 55, 56, 66], a kinematic solution is proposed, exploiting the redundancy of the robotic arm to limit the disturbances on the UAV. Proposed centralized controls are based on Cartesian impedance control [68], integral backstepping [45, 49, 74], and sliding mode control [53]. A centralized control with adaptive terms is proposed in [21, 54, 99]. Another interesting approach [108] is based on passive decomposition [61] applied in a backstepping control structure.

In the above works, the coupling dynamics are considered by compensating torques and forces related to the robotic arm motion. However, the quadrotor compensates torques and forces in different ways. Quadrotors are under-actuated vehicles, since the propellers can just generate forces oriented along the vertical direction of the quadrotor body frame. Thus, in order to compensate forces in other directions, the quadrotor has to change its orientation (also called attitude). On the other hand, a rotation of the quadrotor itself yields a displacement of the end effector and of the center of mass of the robotic arm. Eventually, this results in a disturbance force on the quadrotor itself. These coupling dynamics are not addressed by the existing control approaches, that exploit the quadrotor attitude as though it were a proper control input. The work in [108] applies a decomposition that describes the system as composed of a combination of rotational and translational motions. The authors mention the two goals that torques on the rotational motion have to perform: tracking of a trajectory for the center of mass and achievement of the desired quadrotor orientation. However, conditions assuring the accomplishments of both goals without mutual interaction are not provided.

In this Chapter a centralized control structure for a quadrotor equipped with a robotic arm it will be present.
The approach aims to completely decouple the variables used for the quadrotor control, e.g. roll and pitch motion, from the motion of the end effector and the center of mass of the robotic arm. To this end, the actuated variables of quadrotor and manipulator are exploited to compensate the effects of roll and pitch variation on the center of mass and end effector position. A change of variables is proposed, which implicitly entails the mentioned compensation. Thus, quadrotor roll and pitch angular velocities can be used as proper control variables, similar to the case of a simple quadrotor, without a robotic arm underneath.

Then, on the basis of the novel representation of the system, an inverse dynamics control is proposed, which takes into account the under-actuation of the system. Torques are computed so as to minimize disturbance forces lying on the plane of the quadrotor, which can not be counteracted since they are orthogonal to the UAV thrust vector. The implicit compensation of roll and pitch motion obtained with the new representation of the system, together with the minimization of disturbances provided by the proposed control, lead to a substantial improvement on control performance in terms of tracking error with respect to state of the art algorithms.

The stability of the linearized system around the equilibrium point is proven by means of Lyapunov analysis. The stability proof is not trivial, due the particular structure of the closed loop system matrix. In fact, the minimization of disturbances mentioned above is obtained with pseudo-inversion operations, which in closed loop result in semidefinite idempotent matrices.

The control approach has been validated on a detailed simulator that includes sensor errors, torque saturation and model errors. The method has been tested on various operating points, and it has been compared with a state of the art algorithm.

2.2 Modeling

2.2.1 Kinematic and Dynamic Model

Consider a system composed of a quadrotor equipped with a robotic arm. Let's define a fixed world reference frame Σ_w and a local reference frame Σ_b fixed with the body axes of the quadrotor as shown in Fig. 2.1. We

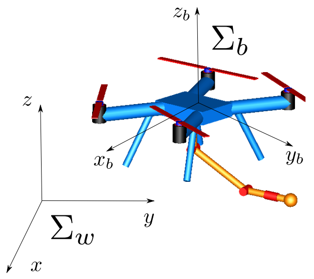


Figure 2.1: *Virtual model of the system. The global and local reference frames are represented.*

define $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{x}_q^T & \boldsymbol{\phi}^T & \boldsymbol{q}^T \end{bmatrix}^T$, $\boldsymbol{\xi} \in \mathbb{R}^n$, where $\boldsymbol{x}_q = \begin{bmatrix} x_q & y_q & z_q \end{bmatrix}^T$ is the absolute position of the center of mass of the quadrotor in the frame Σ_w, $\boldsymbol{\phi} = \begin{bmatrix} \varphi & \theta & \psi \end{bmatrix}^T$ are the roll, pitch and yaw angles of the quadrotor; $\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & \dots & q_{n_r} \end{bmatrix}^T$ are the joint positions of the robotic arm, and n_r is the number of joints. The goal is to track a given trajectory of the end effector position $\boldsymbol{x}_e \in \mathbb{R}^{n_e}$ of the robot, defined in the frame Σ_w. In the following, just the translational part of the end effector motion will be considered, i.e. $n_e = 3$, while the extension to the end effector orientation is straightforward.
The dynamic system can be described by the Euler-Lagrange equations and the kinematic relations as follows.

$$B_o(\boldsymbol{\xi})\ddot{\boldsymbol{\xi}} + C_o\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\dot{\boldsymbol{\xi}} + g_o(\boldsymbol{\xi}) = \boldsymbol{\tau}_o \tag{2.1a}$$

$$\dot{\boldsymbol{x}}_e = J_{eo}(\boldsymbol{\xi})\dot{\boldsymbol{\xi}}, \tag{2.1b}$$

where B_o is the inertia matrix, C_o the matrix of the Coriolis and centrifugal terms, g_o the gravity torques, $\boldsymbol{\tau}_o$ the generalized torques applied to the system and J_{eo} the positional Jacobian of the end effector. The detail derivation of the dynamic and kinematic model of an aerial manipulator is well described in [67]. Notice that $\boldsymbol{\tau}_o = \begin{bmatrix} (R\,\boldsymbol{e}_3\,F)^T & \left(W^{-T}\boldsymbol{\tau}_q\right)^T & \boldsymbol{\tau}_r^T \end{bmatrix}^T$ where $R(\boldsymbol{\phi}) \in SO(3)$ is the rotation matrix from the absolute frame to the body

frame Σ_b, $\boldsymbol{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is the unit vector along the direction of the quadrotor thrust $F \in \mathbb{R}$, the matrix $W(\boldsymbol{\phi})^{-1} \in \mathbb{R}^{3\times 3}$ relates Euler angle derivatives $\dot{\boldsymbol{\phi}}$ to angular velocities of the quadrotor $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ in the body frame Σ_b, and $\boldsymbol{\tau}_q \in \mathbb{R}^3$ and $\boldsymbol{\tau}_r \in \mathbb{R}^{n_r}$ are torques applied to the quadrotor by propellers action and to robot joints, respectively.
Two main characteristics of the system should be pointed out. First, the quadrotor is under-actuated, thus the complete system has less control variables $(n-2)$ than the number of the degrees of freedom (DoFs). As largely known in the literature on quadrotor control, two DoFs of the quadrotor, in particular roll and pitch angles, are used as control variables for the translational motion of the quadrotor in order to deal with the under-actuation of the system.
Secondly, quadrotor and robotic arm dynamics are strongly coupled. When the masses of the two elements are comparable, the coupling effects can make the whole system unstable.
These two main issues will be explicitly addressed by means of a suitable change of variables.

2.2.2 Change of Variables

The proposed change of variables is defined by a transformation linear with respect to the joint derivatives $\dot{\boldsymbol{\xi}}$.
The new set of variables will be composed of three groups: velocities that lie in directions that cannot be directly actuated; velocities of variables that are demanded to control actions (as roll and pitch for the classical quadrotor control scheme); and velocities on actuated directions. The scheme is derived and extended from the usual classification of variables in the classical quadrotor control architecture, which separates roll and pitch variables from the translational degrees of freedom and yaw rotation.
In addition, the set of new variables will be chosen such that the second group, the control variables, can be used neither producing dynamic torques on the first set of velocities (dynamical decoupling), nor giving any direct velocity contribution to the output of the system, the end effector velocity (kinematic decoupling). The change of variables will be performed in two different steps.

First Step of the Change of Variables

The first step aims to explicitly separate the three groups by means of the following transformation matrices:

$$\boldsymbol{\eta}_1 = T_b(\boldsymbol{\phi})\,\dot{\boldsymbol{\xi}} = S\,T_1(\boldsymbol{\phi})\,\dot{\boldsymbol{\xi}}$$

where $T_b(\boldsymbol{\phi}) \in \mathbb{R}^{n\times n}$, $T_1(\boldsymbol{\phi}) \in \mathbb{R}^{n\times n}$ is a block diagonal matrix composed as follows[1]:

$$T_1(\boldsymbol{\phi}) = \begin{bmatrix} R(\boldsymbol{\phi})^T & 0 & 0 \\ 0 & W(\boldsymbol{\phi})^{-1} & 0 \\ 0 & 0 & I \end{bmatrix},$$

and $S \in \mathbb{R}^{n\times n}$ is a selection matrix that rearranges terms in $\boldsymbol{\eta}_1$ such that: $\boldsymbol{\eta}_1 = \begin{bmatrix}\boldsymbol{t}^T & \boldsymbol{\omega}_{xy}^T & \boldsymbol{\alpha}^T\end{bmatrix}^T$. In this way, velocities are divided as desired in the first part not directly controllable $\boldsymbol{t} = \begin{bmatrix}v_{bq,x} & v_{bq,y}\end{bmatrix}^T$, the control variables $\boldsymbol{\omega}_{xy} = \begin{bmatrix}\omega_x & \omega_y\end{bmatrix}^T$, and the controllable variables $\boldsymbol{\alpha} = \begin{bmatrix}v_{bq,z} & \omega_z & \dot{\boldsymbol{q}}^T\end{bmatrix}^T$. Notice that $\boldsymbol{v}_{bq} = \begin{bmatrix}v_{bq,x} & v_{bq,y} & v_{bq,z}\end{bmatrix}^T$ are the quadrotor velocity in the body frame. The first set of variables $\boldsymbol{t}$ are not actuated because they always lie on the xy plane of the body frame Σ_b, thus orthogonal to rotors axes.

Second Step of the Change of Variables

The second step of the change of variables aims to perform the kinematic and dynamic decoupling of the variables $\boldsymbol{\omega}_{xy}$ as explained above, in order to use them as proper control inputs.
To this extent, it is necessary to compute the Jacobian matrices $J_{e1}(\boldsymbol{\xi}) \in \mathbb{R}^{n_e\times n}$ and $J_{c1}(\boldsymbol{\xi}) \in \mathbb{R}^{2\times n}$ that relate $\boldsymbol{\eta}_1$ to, respectively, the end effector velocity $\dot{\boldsymbol{x}}_e$ and the projection $\dot{\boldsymbol{x}}_{c,xy} \in \mathbb{R}^2$ of the velocity of the center of mass of the robotic arm onto the xy plane of the quadrotor body frame:

$$J_{e1}(\boldsymbol{\xi}) = J_{eo}(\boldsymbol{\xi})\,T_b(\boldsymbol{\phi})^{-1}, \quad J_{c1}(\boldsymbol{\xi}) = J_{co}(\boldsymbol{\xi})\,T_b(\boldsymbol{\phi})^{-1} \tag{2.2a}$$

$$J_{ec1}(\boldsymbol{\xi}) = \begin{bmatrix} J_{e1}(\boldsymbol{\xi}) \\ J_{c1}(\boldsymbol{\xi}) \end{bmatrix}, \tag{2.2b}$$

where J_{co} is the Jacobian that relates the velocities $\dot{\boldsymbol{\xi}}$ of the original coordinates to the arm center of mass velocity in the xy quadrotor body plane, and the matrix $J_{ec1}(\boldsymbol{\xi}) \in \mathbb{R}^{(n_e+2)\times n}$ is just defined by gathering matrices J_{e1} and J_{c1} into a single one.

[1] I denotes an identity matrix with suitable dimensions.

Coordinates velocities $\dot{\boldsymbol{x}}_e$ and $\dot{\boldsymbol{x}}_{c,xy}$ have been selected because they are associated to the tasks that must not be affected by $\boldsymbol{\omega}_{xy}$ variables. While the choice of the end effector velocity is quite obvious, the velocity of the center of mass in the xy plane of the body frame is important because its variation is related to the dynamical forces applied on the non actuated directions of the quadrotor (on the xy plane of the body frame, precisely). Our approach aims at canceling the effect of the control variables $\boldsymbol{\omega}_{xy}$ on the non actuated directions of the quadrotor. The achievement of this goal will be demonstrated by the zeros in the structure of the new inertia matrix B computed in the following.

It is useful to organize the matrix J_{ec1} in three submatrices $J_{ec1,t}(\boldsymbol{\xi}) \in \mathbb{R}^{(n_e+2)\times 2}$, $J_{ec1,\omega}(\boldsymbol{\xi}) \in \mathbb{R}^{(n_e+2)\times 2}$ and $J_{ec1,\alpha}(\boldsymbol{\xi}) \in \mathbb{R}^{(n_e+2)\times(n-4)}$, corresponding to the three groups of variables $\boldsymbol{t}$, $\boldsymbol{\omega}_{xy}$ and $\boldsymbol{\alpha}$:

$$J_{ec1}(\boldsymbol{\xi}) = \begin{bmatrix} J_{ec1,t}(\boldsymbol{\xi}) & J_{ec1,\omega}(\boldsymbol{\xi}) & J_{ec1,\alpha}(\boldsymbol{\xi}) \end{bmatrix}.$$

The last definition is used to introduce a new set of velocities $\boldsymbol{\beta} \in \mathbb{R}^{n-4}$ as follows:

$$\boldsymbol{\beta} = \boldsymbol{\alpha} + T_{\beta\omega}(\boldsymbol{\xi})\,\boldsymbol{\omega}_{xy} \tag{2.3}$$

$$T_{\beta\omega}(\boldsymbol{\xi}) = W_T^{-1}\left(J_{ec1,\alpha}(\boldsymbol{\xi})\,W_T^{-1}\right)^{\dagger} J_{ec1,\omega}(\boldsymbol{\xi}), \tag{2.4}$$

where W_T is a weight matrix, the matrix $T_{\beta\omega}(\boldsymbol{\xi}) \in \mathbb{R}^{(n-4)\times 2}$ has been introduced, and the Moore-Penrose right pseudo-inverse operation has been applied[2]. In the following, it is assumed that the matrix $J_{ec1,\alpha}$ is full rank. For this reason, it is required that $n_r \geq n_e$. However, a larger number of robotic arm DoFs is beneficial. A proper trajectory planning algorithm can assure the condition on the rank of $J_{ec1,\alpha}$. To this end, the method presented in Section 3.2.2 can be employed.
Matrix $T_{\beta\omega}$ has the role of projecting the roll and pitch angular velocities $\boldsymbol{\omega}_{xy}$ in an equivalent motion of $\boldsymbol{\alpha}$ variables. The two motions are equivalent for their effect on end effector and center of mass motion. The weight matrix W_T is useful to decide a priori how to distribute the motion on the controllable degrees of freedom. In principle, arm degrees of freedom should have lower weights on account of their dexterity, which is generally higher than yaw rotation and z translation of the quadrotor.
Finally, the set of variables $\boldsymbol{\eta}$ can be expressed as follows:

$$\boldsymbol{\eta} = T(\boldsymbol{\xi})\,\dot{\boldsymbol{\xi}}, \tag{2.5}$$

[2]When $A \in \mathbb{R}^{m\times n}$ the operation $A^{\dagger}$ corresponds to the right pseudo-inverse $A^{\dagger} = A^T\left(AA^T\right)^{-1}$ if rows are linearly independent (so that $m \leq n$), while it corresponds to the left pseudo-inverse $A^{\dagger} = \left(A^TA\right)^{-1}A^T$ if columns are linearly independent ($m \geq n$).

where matrix $T\left(\boldsymbol{\xi}\right) \in \mathbb{R}^{n \times n}$ depends on matrix $T_\beta\left(\boldsymbol{\xi}\right) \in \mathbb{R}^{n \times n}$:

$$T\left(\boldsymbol{\xi}\right) = T_\beta\left(\boldsymbol{\xi}\right) S\, T_1\left(\boldsymbol{\phi}\right)$$

$$T_\beta\left(\boldsymbol{\xi}\right) = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & T_{\beta\omega}\left(\boldsymbol{\xi}\right) & I \end{bmatrix}.$$

New Representation of the System

The new variables $\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{t}^T & \boldsymbol{\omega}_{xy}^T & \boldsymbol{\beta}^T \end{bmatrix}^T$ can be substituted in (2.1):

$$B\left(\boldsymbol{\xi}\right)\dot{\boldsymbol{\eta}} + C\left(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}\right)\boldsymbol{\eta} + g\left(\boldsymbol{\xi}\right) = \boldsymbol{\tau} \tag{2.6a}$$

$$\dot{\boldsymbol{x}}_e = J_{e\eta}\left(\boldsymbol{\xi}\right)\boldsymbol{\eta}, \tag{2.6b}$$

where matrices B, C, g, $\boldsymbol{\tau}$ and $J_{e\eta}$ are computed as follows:

$$B = T^{-T} B_o T^{-1}, \quad C = T^{-T} C_o T^{-1} + T^{-T} B_o \dot{\overline{T^{-1}}}, \tag{2.7}$$

$$g = T^{-T} g_o, \quad \boldsymbol{\tau} = T^{-T}\boldsymbol{\tau}_o, \quad J_{e\eta} = J_{eo} T^{-1}. \tag{2.8}$$

The new matrices exhibit the desired properties, useful for the control approach that will be presented in the next Section. In particular, the inertia matrix and the end effector Jacobian take the following forms, respectively:

$$B = \begin{bmatrix} B_{tt} & 0 & B_{t\beta} \\ 0 & B_{\omega\omega} & B_{\omega\beta} \\ B_{\beta t} & B_{\beta\omega} & B_{\beta\beta} \end{bmatrix}, \quad J_{e\eta} = \begin{bmatrix} J_{e\eta,t} & 0 & J_{e\eta,\beta} \end{bmatrix}. \tag{2.9}$$

Notice that the control variables $\boldsymbol{\omega}_{xy}$ have no effect on the end effector velocity (they are kinematically decoupled, see the structure of $J_{e\eta}$), while their derivatives $\dot{\boldsymbol{\omega}}_{xy}$ do not produce any torque on the non controllable variables $\boldsymbol{t}$ (they are dynamically decoupled, see the structure of B). Moreover, Appendix B.1 demonstrates that, adding a concentrated mass on the robot end effector (i.e. a load due to the task), matrix T does not change, then matrices B and $J_{e\eta}$ keep the same structure.
Finally, equations of (2.6) can be written highlighting the properties of the new set of variables:

$$\begin{bmatrix} B_{tt} & 0 & B_{t\beta} \\ 0 & B_{\omega\omega} & B_{\omega\beta} \\ B_{\beta t} & B_{\beta\omega} & B_{\beta\beta} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{t}} \\ \dot{\boldsymbol{\omega}}_{xy} \\ \dot{\boldsymbol{\beta}} \end{bmatrix} + \begin{bmatrix} C_t^v \\ C_\omega^v \\ C_\beta^v \end{bmatrix} + \begin{bmatrix} g_t \\ g_\omega \\ g_\beta \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{\tau}_\omega \\ \boldsymbol{\tau}_\beta \end{bmatrix} \tag{2.10a}$$

$$\dot{\boldsymbol{x}}_e = \begin{bmatrix} J_{e\eta,t} & 0 & J_{e\eta,\beta} \end{bmatrix}\boldsymbol{\eta}, \tag{2.10b}$$

where $C^v = \begin{bmatrix} C_t^{vT} & C_\omega^{vT} & C_\beta^{vT} \end{bmatrix}^T = C\boldsymbol{\eta}, \quad g = \begin{bmatrix} g_t{}^T & g_\omega{}^T & g_\beta{}^T \end{bmatrix}^T$ and $\boldsymbol{\tau} = \begin{bmatrix} 0^T & \boldsymbol{\tau}_\omega{}^T & \boldsymbol{\tau}_\beta{}^T \end{bmatrix}^T$. Notice that control torques on the first group of variables $\boldsymbol{t}$ are identically null, because velocities on the non actuated directions of the quadrotor body frame have been chosen.
In the following, matrices B, C and g will be considered lower and upper bounded. In particular, the interesting works in [38, 97] show how vehicle-manipulator systems can lose global boundedness property for inertia matrix and skew-simmetric property for the Coriolis matrix depending on the chosen representation. This is due to Euler angles singularities. As the aerial manipulator system are not meant to perform aerobatic maneuvers, it is assumed that roll and pitch angles will take values far from singularity points, such that usual properties hold.

2.3 Control Approach

In this Section the tracking control will be presented. As extensively discussed, the system presents two main critical issues for the control, which are the under-actuation of the quadrotor and the coupled dynamics of the two systems. The proposed inverse dynamics control approach deals with these issues.

2.3.1 Discussion on the Relative Degree of the System

It is useful to represent the system in (2.1) in the form affine in control:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + b(\boldsymbol{x})\,\boldsymbol{v} \tag{2.11a}$$
$$\boldsymbol{y} = h(\boldsymbol{x}) \tag{2.11b}$$

where $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\xi}^T & \dot{\boldsymbol{\xi}}^T \end{bmatrix}^T$, $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{\tau}_\omega^T & \boldsymbol{\tau}_\beta^T \end{bmatrix}^T$, $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{t}^T & \boldsymbol{\beta}^T \end{bmatrix}^T$, and:

$$\begin{aligned} f(\boldsymbol{x}) &= \begin{bmatrix} \dot{\boldsymbol{\xi}} \\ -B_o^{-1}\left(C_o\dot{\boldsymbol{\xi}} + g_o\right) \end{bmatrix}, \\ b(\boldsymbol{x}) &= \begin{bmatrix} 0 \\ B_o^{-1}G \end{bmatrix}, \qquad\qquad h(\boldsymbol{x}) = H(\boldsymbol{\xi})\,\dot{\boldsymbol{\xi}}, \end{aligned} \tag{2.12}$$

where:

$$\begin{aligned} G &= T^T S_b, & H &= S_h T, \\ S_b &= \begin{pmatrix} 0_{2\times n-2} \\ I_{n-2} \end{pmatrix}, & S_h &= \begin{pmatrix} I_2 & 0 & 0 \\ 0 & 0 & I_{n-4} \end{pmatrix}. \end{aligned} \tag{2.13}$$

The state has been represented with the original generalized coordinates because the new set of variables are pseudo-velocities and not physical variables. Nevertheless, inputs $\boldsymbol{v}$ has been chosen equal to the control variables in the new reference system $\boldsymbol{\tau}_\omega$ and $\boldsymbol{\tau}_\beta$, and the pseudo-velocities $\boldsymbol{t}$ and $\boldsymbol{\beta}$ have been selected as outputs $\boldsymbol{y}$.

Proposition 1. *The relative degree of the system in* (2.11) *is not well defined. The Lie derivative $L_b h$, reported in the following equation, has rank equal to $n-4$, while input and output have dimensions $n-2$:*

$$L_b h = \frac{dh}{d\boldsymbol{x}} b = S_h T B_o^{-1} T^T S_b = S_h B^{-1} S_b. \tag{2.14}$$

The proof of Proposition 1 is reported in Appendix B.2. Since the matrix $L_b h$ is neither invertible, nor identically null, the relative degree of the system is not well defined.

Remark Notice that Proposition 1 is due to the particular change of variables applied. In fact, imposing that the robot center of mass velocity is not influenced by roll and pitch velocities, implies that torques on roll and pitch cannot directly impose an acceleration of the quadrotor in x and y directions. This is the physical explanation of the obtained result. With the present approach, the interaction between roll and pitch motion and quadrotor translational velocities is the same as in a typical quadrotor, without any disturbances due to the coupled dynamics with the arm.

This result has an important implication. A not well defined relative degree implies that feedback linearization approach cannot be applied without performing a dynamic extension [48]. The classical approach of feedback linearization requires the calculation of the Lie derivatives $L_g L_f^k h$ of the system after the dynamic extension: the physical meaning of the considered variables will be lost and the complexity of the derivation would considerably increase. Therefore, a different approach will be presented in the following.

2.3.2 Control Structure

A sketch of the overall control scheme is reported in Fig. 2.2.

In the first place, a kinematic feedback control action is computed, as described in Sec. 2.3.3. Given as input a reference trajectory $\boldsymbol{\xi}_r$ and $\dot{\boldsymbol{\xi}}_r$, and actual values of variables $\boldsymbol{\xi}$ and $\dot{\boldsymbol{\xi}}$, a reference signal $\boldsymbol{u}_{t\beta}$ is computed. This reference signal represents the desired derivative of variables $\boldsymbol{\eta}_{t\beta} = \begin{bmatrix}\boldsymbol{t}^T & \boldsymbol{\beta}^T\end{bmatrix}^T$.

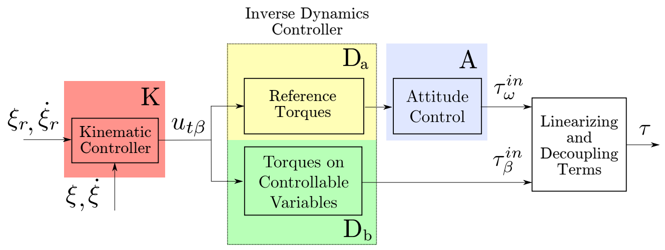


Figure 2.2: *Overall control scheme. Three main sections are identified, the kinematic control, the inverse dynamics control and the attitude control.*

Then, an inverse dynamics controller is applied to track the reference signal $\boldsymbol{u}_{t\beta}$ (the detailed development will be reported in Sec. 2.3.4). It is divided in two parts.

The first part, corresponding to box D_a in Fig. 2.2, has the goal of computing the reference quadrotor attitude that allows to produce the torques required to track the reference signal $\boldsymbol{u}_{t\beta}$. In fact, in general, a set of prescribed torques can be applied with a suitable orientation of the quadrotor, such that propellers thrust applies forces in the desired direction. Thus, reference torques are computed, by substituting the control action $\boldsymbol{u}_{t\beta}$ to the derivatives $\dot{\boldsymbol{t}}$ and $\dot{\boldsymbol{\beta}}$ in the dynamic model of (2.10a), where $\dot{\boldsymbol{\omega}}_{xy}$ is imposed to be null. Then, computed reference torques are the input to an attitude controller, which drives the quadrotor tilting, i.e. roll and pitch, to the desired orientation. It corresponds to box A in Fig. 2.2. Torques on quadrotor angular velocity $\boldsymbol{\tau}_\omega$, output of the attitude controller, will be composed of three terms:

$$\boldsymbol{\tau}_\omega = \boldsymbol{\tau}_\omega^{in} + \boldsymbol{\tau}_\omega^{t\beta} + \boldsymbol{\tau}_\omega^{Cg} \tag{2.15}$$

where torques $\boldsymbol{\tau}_\omega^{in}$ depend on the desired acceleration $\dot{\boldsymbol{\omega}}_{xy}$ to be tracked, $\boldsymbol{\tau}_\omega^{Cg}$ is the linearizing term for Coriolis, centrifugal and gravity torques, and $\boldsymbol{\tau}_\omega^{t\beta}$ is a decoupling term that will be defined in the following.

In the second part of the inverse dynamics controller, corresponding to box D_b in Fig. 2.2, actual torques $\boldsymbol{\tau}_\beta$ are computed, such that the difference between variables $\dot{\boldsymbol{\eta}}_{t\beta}$ and reference signal $\boldsymbol{u}_{t\beta}$ is minimized. Similar to the attitude controller, torques $\boldsymbol{\tau}_\beta$ will be computed as follows:

$$\boldsymbol{\tau}_\beta = \boldsymbol{\tau}_\beta^{in} + \boldsymbol{\tau}_\beta^{\omega} + \boldsymbol{\tau}_\beta^{Cg} \tag{2.16}$$

where $\boldsymbol{\tau}_\beta^{in}$ is a control action that depends on the reference signal $\boldsymbol{u}_{t\beta}$, and $\boldsymbol{\tau}_\beta^{Cg}$ and $\boldsymbol{\tau}_\beta^{\omega}$ are the linearizing and decoupling terms.
Finally, torques $\boldsymbol{\tau}_\omega$ and $\boldsymbol{\tau}_\beta$ will be expressed in the original variables representation to obtain the physical torques to be applied on the system.

2.3.3 Feedback Kinematic Controller

In the following, it is assumed that a specific algorithm generates suitable references $\boldsymbol{\xi}_r$ and $\dot{\boldsymbol{\xi}}_r$ for the quadrotor and robot positions, and they are passed as input to the proposed controller. Thus, a simple kinematic controller can be set, in order to track the given reference:

$$\boldsymbol{\eta}_{t\beta,r} = T_{t\beta}\left(K_{\xi_{ff}}\dot{\boldsymbol{\xi}}_r + K_\xi\left(\boldsymbol{\xi}_r - \boldsymbol{\xi}\right)\right), \tag{2.17}$$

$$\boldsymbol{\eta}_{t\beta} = T_{t\beta}\,\dot{\boldsymbol{\xi}} \tag{2.18}$$

$$\boldsymbol{u}_{t\beta} = K_{t\beta}\left(\boldsymbol{\eta}_{t\beta,r} - \boldsymbol{\eta}_{t\beta}\right) \tag{2.19}$$

where $K_{\xi_{ff}} \leq 1$, K_ξ and $K_{t\beta}$ are positive scalar gains, and $K_{t\beta} \gg K_\xi$ such that time scale separation principle can be assumed. Matrix $T_{t\beta} \in \mathbb{R}^{(n-2)\times n}$ is obtained by matrix T, as follows: $T_{t\beta} = S_h T$, where S_h has been defined in (2.13). Variables $\boldsymbol{\eta}_{t\beta}$, defined in Sec. 2.3.2, are a subset of pseudo-velocities $\boldsymbol{\eta}$, such that $\boldsymbol{\eta}_{t\beta} = S_h\boldsymbol{\eta} = T_{t\beta}\dot{\boldsymbol{\xi}}$, and $\boldsymbol{\eta}_{t\beta,r}$ are the corresponding reference.

Remark Notice that matrix $T_{t\beta}$ is rectangular, then it presents a non empty null space. The motion in the null space of $T_{t\beta}$ corresponds to the motion of system variables that compensates the effects of quadrotor tilting on end effector and center of mass positions.

In a fully actuated system, the required torques could be computed by multiplying the obtained desired acceleration $\boldsymbol{u}_{t\beta}$ by the inertia matrix and summing the non linear Coriolis, centrifugal and gravity torques. As the system is underactuated, a different approach is required.

2.3.4 Design of the Inverse Dynamics Controller

Let's consider the first and third set of equations in (2.10a):

$$\begin{bmatrix} B_{tt} & B_{t\beta} \\ B_{\beta t} & B_{\beta\beta} \end{bmatrix}\begin{bmatrix} \dot{\boldsymbol{t}} \\ \dot{\boldsymbol{\beta}} \end{bmatrix} + \begin{bmatrix} 0 \\ B_{\beta\omega} \end{bmatrix}\dot{\boldsymbol{\omega}}_{xy} + \begin{bmatrix} C_t^v \\ C_\beta^v \end{bmatrix} + \begin{bmatrix} g_t \\ g_\beta \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{\tau}_\beta \end{bmatrix}. \tag{2.20}$$

Notice the zero of the equations in place of the actuation $\boldsymbol{\tau}_t$, consistent with the fact that variables $\boldsymbol{t}$ cannot be directly actuated. The proposed control aims to find the torques $\boldsymbol{\tau}_\beta$ and the reference values for $\boldsymbol{\omega}_{xy}$ to track the desired accelerations $\boldsymbol{u}_t$ and $\boldsymbol{u}_\beta$ computed in Sec. 2.3.3. The approach is divided in two main parts. In the first part, corresponding to block D_a in Fig. 2.2, it is assumed that the system is completely actuated, thus a reference torque $\boldsymbol{\tau}_{t,r}$ acting on the variables $\boldsymbol{t}$ is computed in order

to generate a reference value for the attitude control. In the second part, corresponding to block D_b in Fig. 2.2, the under actuation of the system is considered and the torques $\boldsymbol{\tau}_\beta^{in}$ that minimize disturbances on the non controllable variables are computed.
Thus, assume that the actuation can generate a torque $\boldsymbol{\tau}_t$. The torques $\boldsymbol{\tau}_{t,r}$ and $\tau_{\beta z,r}$ (the first component of the torques $\boldsymbol{\tau}_{\beta,r}$), which we would ideally apply in the case the system were fully actuated, can be computed by means of the well-known inverse dynamics approach:

$$\begin{bmatrix}\boldsymbol{\tau}_{t,r}\\ \tau_{\beta z,r}\end{bmatrix} = \begin{bmatrix}B_{tt} & B_{t\beta}\\ B_{\beta zt} & B_{\beta z\beta}\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_t\\ \boldsymbol{u}_\beta\end{bmatrix} + \begin{bmatrix}C_t^v + g_t\\ C_{\beta z}^v + g_{\beta z}\end{bmatrix}. \tag{2.21}$$

where the system is obtained by the first three equation lines of (2.20), and a steady condition with respect to the quadrotor roll and pitch rotation has been considered, i.e. $\dot{\boldsymbol{\omega}}_{xy} = 0$. The obtained torques $\begin{bmatrix}\boldsymbol{\tau}_{t,r}^T & \tau_{\beta z,r}\end{bmatrix}^T$, which correspond to the forces to be applied to the quadrotor basis, are the input of the attitude control, discussed in the next Section. The attitude control will drive the quadrotor to orient its thrust in the desired direction.
Let's now account for the under-actuation of the system, as reported in (2.20), in order to compute the actual torques to be assigned to the system. Substituting in (2.20) the definition of $\boldsymbol{\tau}_\beta$ given in (2.16), where the linearizing torques $\boldsymbol{\tau}_\beta^{Cg} = C_\beta^v + g_\beta$ cancel with the corresponding terms, and $\boldsymbol{\tau}_\beta^\omega$ is equal to the coupling term $B_{\beta\omega}\dot{\boldsymbol{\omega}}_{xy}$, it results as follows:

$$\begin{bmatrix}B_{tt} & B_{t\beta}\\ B_{\beta t} & B_{\beta\beta}\end{bmatrix}\begin{bmatrix}\dot{\boldsymbol{t}}\\ \dot{\boldsymbol{\beta}}\end{bmatrix} = \begin{bmatrix}-\left(C_t^v + g_t\right)\\ \boldsymbol{\tau}_\beta^{in}\end{bmatrix}, \tag{2.22}$$

Substituting the desired acceleration $\boldsymbol{u}_{t\beta}$ in (2.22), in general the system has no solution in the variables $\boldsymbol{\tau}_\beta^{in}$. Thus, the torques $\boldsymbol{\tau}_\beta^{in}$ that minimize the following quantity should be found:

$$\boldsymbol{\tau}_\beta^{in} = \arg\min_{\boldsymbol{\tau}_\beta^{in}} \left|\left|\boldsymbol{u}_{t\beta} - B^{t\beta^{-1}}\begin{bmatrix}-\left(C_t^v + g_t\right)\\ \boldsymbol{\tau}_\beta^{in}\end{bmatrix}\right|\right|^2, \tag{2.23}$$

where the matrix $B^{t\beta}$ is structured as follows:

$$B^{t\beta} = \begin{bmatrix}B_{tt} & B_{t\beta}\\ B_{\beta t} & B_{\beta\beta}\end{bmatrix}.$$

Notice that matrix $B^{t\beta}$, which is a submatrix of matrix B defined in (2.9), is invertible and symmetric positive definite, as the following property holds.

Proposition 2. *Defining the symmetric matrix B and its submatrix $\hat{B}$ structured as follows:*

$$B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12}^T & B_{22} & B_{23} \\ B_{13}^T & B_{23}^T & B_{33} \end{bmatrix} \quad \hat{B} = \begin{bmatrix} B_{11} & B_{13} \\ B_{13}^T & B_{33} \end{bmatrix} .$$

If matrix B is symmetric positive definite, then $\hat{B}$ is symmetric positive definite.

The proof of Proposition 2 is reported in Appendix B.3. In order to solve the minimization in (2.23), matrix ${B^{t\beta}}^{-1}$ is divided in the sets of columns corresponding to the non actuated coordinates B_t^i and to the actuated ones B_β^i, and the optimal torques for tracking the desired accelerations are computed from (2.23) as follows:

$$\begin{aligned} {B^{t\beta}}^{-1} &= \begin{bmatrix} B_t^i & B_\beta^i \end{bmatrix} , \\ \boldsymbol{\tau}_\beta^{in} &= {B_\beta^i}^\dagger \left(\boldsymbol{u}_{t\beta} + B_t^i \left(C_t^v + g_t \right) \right) . \end{aligned} \tag{2.24}$$

The computation of torques $\boldsymbol{\tau}_\beta^\omega$ can be performed after the completion of the attitude control.
The accelerations $\dot{\boldsymbol{t}}^*$ and $\dot{\boldsymbol{\beta}}^*$ that will be obtained applying the control torques $\boldsymbol{\tau}_\beta$ can be computed as:

$$\begin{bmatrix} \dot{\boldsymbol{t}}^* \\ \dot{\boldsymbol{\beta}}^* \end{bmatrix} = B^{t\beta^{-1}} \begin{bmatrix} -\left(C_t^v + g_t \right) \\ \boldsymbol{\tau}_\beta^{in} \end{bmatrix} . \tag{2.25}$$

Then, the decoupling term $\boldsymbol{\tau}_\omega^{t\beta}$ results as follows:

$$\boldsymbol{\tau}_\omega^{t\beta} = B_{\omega\beta} \dot{\boldsymbol{\beta}}^* \tag{2.26}$$

Control Variables

We will finally address the control of the attitude of the quadrotor in order to provide the required torques on the non actuated directions. In particular, the quadrotor thrust will be oriented in direction of the forces required by the upper level control loop.
Once the required reference forces in the non actuated directions have been computed as in (2.21), the desired rotations of the quadrotor with respect to the current body axes x_b and y_b, denoted as $\delta\boldsymbol{\phi}_{xy,r}$, can be computed as follows:

$$\delta\boldsymbol{\phi}_{xy,r} = D \frac{1}{|F_{zb}|} \begin{bmatrix} F_{xb} \\ F_{yb} \end{bmatrix} , \tag{2.27}$$

where:

$$\begin{bmatrix} F_{xb} & F_{yb} & F_{zb} \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{\tau}_{t,r}^T & \tau_{\beta z,r} \end{bmatrix}^T, \quad D = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{2.28}$$

Quantity $\delta\boldsymbol{\phi}_{xy,r}$ represents roll and pitch angles describing the rotation between the current quadrotor attitude and the desired one.
Reference body angular velocities $\boldsymbol{\omega}_{xy,r}$ are computed by means of a proportional scalar gain K_ϕ:

$$\boldsymbol{\omega}_{xy,r} = K_\phi \delta\boldsymbol{\phi}_{xy,r} - \boldsymbol{\gamma}_\omega, \tag{2.29}$$

where $\boldsymbol{\gamma}_\omega$ is a decoupling term:

$$\boldsymbol{\gamma}_\omega = D^T \begin{bmatrix} \varphi \\ \theta \end{bmatrix} \beta_\psi, \tag{2.30}$$

β_ψ is the second component of $\boldsymbol{\beta}$, and φ and θ are roll and pitch angles of the quadrotor orientation. Notice that the present attitude controller has been obtained considering the small angles approximation for roll and pitch. See Appendix B.4 for its derivation.

Finally, another loop is closed on velocity reference, and the torques $\boldsymbol{\tau}_\omega^{in}$ can be computed.

$$\dot{\boldsymbol{\omega}}_{xy}^* = K_\omega \left(\boldsymbol{\omega}_{xy,r} - \boldsymbol{\omega}_{xy}\right) - \boldsymbol{\gamma}_{\dot{\omega}}, \tag{2.31}$$

$$\boldsymbol{\tau}_\omega^{in} = B_{\omega\omega} \dot{\boldsymbol{\omega}}_{xy}^*, \tag{2.32}$$

where K_ω is a positive scalar gain, $\boldsymbol{\gamma}_{\dot{\omega}}$ is another decoupling term, and the torques $\boldsymbol{\tau}_\omega^{in}$, similar to torques $\boldsymbol{\tau}_\beta^{in}$, are the inertial component of the final torques $\boldsymbol{\tau}_\omega$. The term $\boldsymbol{\gamma}_{\dot{\omega}}$ is analogous to the term in (2.30):

$$\boldsymbol{\gamma}_{\dot{\omega}} = D^T \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \end{bmatrix} \beta_\psi + D^T \begin{bmatrix} \varphi \\ \theta \end{bmatrix} \dot{\beta}_\psi^* \tag{2.33}$$

where $\dot{\beta}_\psi^*$ is the second component of $\dot{\boldsymbol{\beta}}^*$, computed in (2.25). Finally, the three terms composing torques $\boldsymbol{\tau}_\omega$ can be computed as follows:

$$\boldsymbol{\tau}_\omega^{in} = B_{\omega\omega} \dot{\boldsymbol{\omega}}_{xy}^*, \quad \boldsymbol{\tau}_\omega^{Cg} = C_\omega^v + g_\omega, \quad \boldsymbol{\tau}_\omega^{t\beta} = B_{\omega\beta} \dot{\boldsymbol{\beta}}^*$$

The desired acceleration $\dot{\boldsymbol{\omega}}_{xy}^*$ is used to compute the decoupling torques $\boldsymbol{\tau}_\beta^\omega$:

$$\boldsymbol{\tau}_\beta^\omega = B_{\beta\omega} \dot{\boldsymbol{\omega}}_{xy}^*$$

Overall Control Torques

The complete torques can be computed by summing the control torques $\boldsymbol{\tau}_\beta^{in}$ and $\boldsymbol{\tau}_\omega^{in}$ to the linearizing and decoupling terms, as in (2.16) and (2.15). The control torques in the new variables set can be obtained as follows:

$$\boldsymbol{\tau} = \begin{bmatrix} 0 \\ \boldsymbol{\tau}_\omega \\ \boldsymbol{\tau}_\beta \end{bmatrix} = \begin{bmatrix} 0 \\ \boldsymbol{\tau}_\omega^{in} \\ \boldsymbol{\tau}_\beta^{in} \end{bmatrix} + \begin{bmatrix} 0 \\ B_{\omega\beta}\dot{\boldsymbol{\beta}}^* \\ B_{\beta\omega}\dot{\boldsymbol{\omega}}_{xy}^* \end{bmatrix} + \begin{bmatrix} 0 \\ C_\omega^v + g_\omega \\ C_\beta^v + g_\beta \end{bmatrix} \tag{2.34}$$

Obviously the first rows are null because they correspond to the non actuated directions.
Finally, torques in the physical variables can be computed:

$$\boldsymbol{\tau}_o = T^T \boldsymbol{\tau},$$

where the thrust and the torques on the quadrotor have to be reported to rotor velocities with the well-known matrix composed of geometrical and aerodynamic terms [71]. The entire inverse dynamics control scheme is represented in Fig. 2.3.

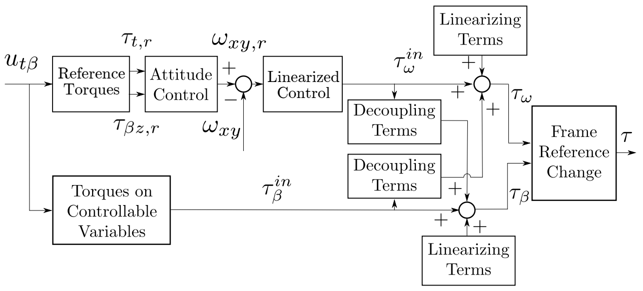


Figure 2.3: *Inverse dynamic control scheme. The upper branch of the scheme represents the attitude control, while the lower branch is the control of the completely actuated variables.*

2.4 Stability

In this Section, the stability of the system around the equilibrium point will be discussed. In particular, the linearization considers small rotations of the quadrotor body. This is a quite reasonable hypothesis because a quadrotor with an arm is not expected to perform aerobatic or aggressive maneuvers. We here assume perfect knowledge of the model and the states of the system.

2.4.1 Preliminary: Tracking of the Velocity Reference

Let's prove that, given the torques $\boldsymbol{\tau}$ computed as in (2.34), the system accelerations $\begin{bmatrix}\dot{\boldsymbol{t}} & \dot{\boldsymbol{\omega}}_{xy} & \dot{\boldsymbol{\beta}}\end{bmatrix}^T$ are identically equal to the reference signals $\begin{bmatrix}\dot{\boldsymbol{t}}^* & \dot{\boldsymbol{\omega}}^*_{xy} & \dot{\boldsymbol{\beta}}^*\end{bmatrix}^T$. By substituting (2.34) in (2.10a) and canceling the corresponding terms, it results:

$$\begin{bmatrix} B_{tt} & 0 & B_{t\beta} \\ 0 & B_{\omega\omega} & B_{\omega\beta} \\ B_{\beta t} & B_{\beta\omega} & B_{\beta\beta} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{t}} \\ \dot{\boldsymbol{\omega}}_{xy} \\ \dot{\boldsymbol{\beta}} \end{bmatrix} = \begin{bmatrix} -\left(g_t + C_t^v\right) \\ B_{\omega\omega}\dot{\boldsymbol{\omega}}^*_{xy} + B_{\omega\beta}\dot{\boldsymbol{\beta}}^* \\ \boldsymbol{\tau}^{in}_{\beta} + B_{\beta\omega}\dot{\boldsymbol{\omega}}^*_{xy} \end{bmatrix} \tag{2.35}$$

where the definition of $\boldsymbol{\tau}^{in}_{\omega}$ given in (2.32) has been used. It can easily be shown by substitution that $\begin{bmatrix}\dot{\boldsymbol{t}}^* & \dot{\boldsymbol{\omega}}^*_{xy} & \dot{\boldsymbol{\beta}}^*\end{bmatrix}^T$ is a solution of the system of equation in (2.35). To this end, consider that, from the definition of $\dot{\boldsymbol{t}}^*$ and $\dot{\boldsymbol{\beta}}^*$ given in (2.25), it follows:

$$\begin{bmatrix} B_{tt} & B_{t\beta} \\ B_{\beta t} & B_{\beta\beta} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{t}}^* \\ \dot{\boldsymbol{\beta}}^* \end{bmatrix} = \begin{bmatrix} -\left(g_t + C_t^v\right) \\ \boldsymbol{\tau}^{in}_{\beta} \end{bmatrix}. \tag{2.36}$$

Thus, the vector $\begin{bmatrix}\dot{\boldsymbol{t}}^* & \dot{\boldsymbol{\omega}}^*_{xy} & \dot{\boldsymbol{\beta}}^*\end{bmatrix}^T$ is a solution of the system of equations and it is also unique, because the matrix B is invertible. Thanks to this result, in the next part of this Section those reference signals will be assumed perfectly tracked: $\dot{\boldsymbol{\eta}} = \dot{\boldsymbol{\eta}}^*$. Notice that it does not mean that $\dot{\boldsymbol{\eta}}_{t\beta}$ is equal to the reference acceleration $\boldsymbol{u}_{t\beta}$.

2.4.2 Linearized System

The system will be analyzed for small values of roll and pitch angles, thus the equations will be linearized around the equilibrium point: $\begin{bmatrix}\varphi_0 & \theta_0\end{bmatrix}^T = \begin{bmatrix}0, 0\end{bmatrix}^T$.
In addition, the analysis will be performed linearizing the robot coordinates around the equilibrium point $\boldsymbol{\xi}_o = \begin{bmatrix}\boldsymbol{x}^T_{q,o} & \boldsymbol{\phi}^T_o & \boldsymbol{q}^T_o\end{bmatrix}^T$, $\dot{\boldsymbol{\xi}}_o = 0$ where $\boldsymbol{\phi}_o = \begin{bmatrix}0, 0, \psi_o\end{bmatrix}^T$. In fact, as the stability analysis is pretty involved, we neglect the evolution along time of inertia and Coriolis matrices, which depend on robot configuration.
The feedback control law on $\boldsymbol{\eta}_{t\beta}$, whose general form is presented in (2.19), will just consist of a proportional term: $\boldsymbol{u}_{t\beta} = -K_{t\beta}\,\boldsymbol{\eta}_{t\beta}$.
We here report the complete equations describing the dynamics of the sys-

tem, where no assumptions have been enforced:

$$\boldsymbol{u}_{t\beta} = -K_{t\beta}\,\boldsymbol{\eta}_{t\beta} \tag{2.37}$$

$$\dot{\boldsymbol{\eta}}_{t\beta} = B^{t\beta^{-1}} \begin{bmatrix} -\left(C_t^v + g_t\right) \\ B_\beta^{i\,\dagger}\left(\boldsymbol{u}_{t\beta} + B_t^i\left(C_t^v + g_t\right)\right) \end{bmatrix} \tag{2.38}$$

$$\boldsymbol{\omega}_{xy,r} = \frac{K_\phi}{\left|\tau_{\beta z,r}\right|} D\boldsymbol{\tau}_{t,r} - \boldsymbol{\gamma}_\omega \tag{2.39}$$

$$\dot{\boldsymbol{\omega}}_{xy}^* = K_\omega\left(\boldsymbol{\omega}_{xy,r} - \boldsymbol{\omega}_{xy}\right) - \boldsymbol{\gamma}_{\dot{\omega}} \tag{2.40}$$

where (2.24) in (2.25); equations (2.27) and (2.28) have been substituted in (2.29) obtaining (2.39); in (2.40) equation (2.31) has been reported.
In (2.38) the term $C_t^v + g_t$ is linearized as follows:

$$C_t^v + g_t = -\bar{g}_z\boldsymbol{\sigma}, \tag{2.41}$$

where $\boldsymbol{\sigma} = D^T\begin{bmatrix}\varphi & \theta\end{bmatrix}^T$ and $\bar{g}_z \in \mathbb{R}^+$ is the thrust the quadrotor keeps at the equilibrium to sustain the mass of the system. Notice that $DD^T = D^TD = I$.
In Appendix B.5 the linearization of the term in (2.41) is presented.
The terms $\boldsymbol{\tau}_{t,r}$ and $\tau_{\beta z,r}$ in (2.39) are linearized as follows:

$$\boldsymbol{\tau}_{t,r} = B_t\boldsymbol{u}_{t\beta} - \bar{g}_z\boldsymbol{\sigma}, \tag{2.42}$$

$$\tau_{\beta z,r} = \bar{g}_z, \tag{2.43}$$

where $B_t = \begin{bmatrix}B_{tt} & B_{t\beta}\end{bmatrix}$. The matrices of the system are evaluated at the equilibrium point.
Thus (2.38) can be written as follows:

$$\dot{\boldsymbol{\eta}}_{t\beta} = B_\beta^i B_\beta^{i\,\dagger}\boldsymbol{u}_{t\beta} + \left(I - B_\beta^i B_\beta^{i\,\dagger}\right) B_t^i\bar{g}_z\boldsymbol{\sigma}. \tag{2.44}$$

Again considering the approximation for small roll and pitch angles, see Appendix B.4, the following equations are enforced:

$$\boldsymbol{\omega}_{xy} = D\dot{\boldsymbol{\sigma}} - \boldsymbol{\sigma}\beta_\psi, \tag{2.45}$$

$$\dot{\boldsymbol{\omega}}_{xy}^* = \dot{\boldsymbol{\omega}}_{xy} = D\ddot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\sigma}}\beta_\psi - \boldsymbol{\sigma}\dot{\beta}_\psi. \tag{2.46}$$

Remind that $\boldsymbol{\gamma}_\omega = \boldsymbol{\sigma}\beta_\psi$ and $\boldsymbol{\gamma}_{\dot{\omega}} = \dot{\boldsymbol{\sigma}}\beta_\psi + \boldsymbol{\sigma}\dot{\beta}_\psi^* = \dot{\boldsymbol{\sigma}}\beta_\psi + \boldsymbol{\sigma}\dot{\beta}_\psi$. Then, we define:

$$\dot{\boldsymbol{\sigma}}_r = D^T\left(\boldsymbol{\omega}_{xy,r} + \boldsymbol{\sigma}\beta_\psi\right). \tag{2.47}$$

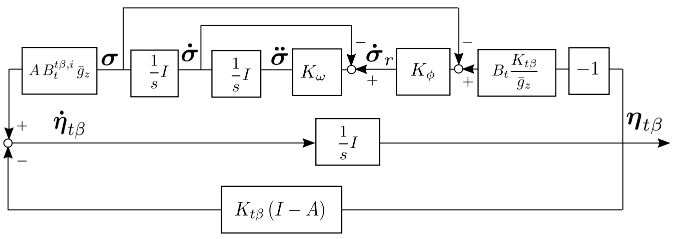

$AB_t^{t\beta,i}\bar{g}_z$
$\boldsymbol{\sigma}$
$\frac{1}{s}I$
$\dot{\boldsymbol{\sigma}}$
$\frac{1}{s}I$
$\ddot{\boldsymbol{\sigma}}$
K_ω
$-$
$\dot{\boldsymbol{\sigma}}_r$
$+$
K_ϕ
$-$
$+$
$B_t\frac{K_{t\beta}}{\bar{g}_z}$
-1
$+$
$\dot{\boldsymbol{\eta}}_{t\beta}$
$-$
$\frac{1}{s}I$
$\boldsymbol{\eta}_{t\beta}$
$K_{t\beta}(I-A)$

Figure 2.4: *Dynamics of the closed loop system linearized around the equilibrium.*

Considering the performed linearization, substituting (2.37) in (2.44) and (2.45),(2.46),(2.47) in (2.39) and (2.40), we obtain the following system of equations:

$$\dot{\boldsymbol{\eta}}_{t\beta} = -K_{t\beta}\boldsymbol{\eta}_{t\beta} + A\left(K_{t\beta}\boldsymbol{\eta}_{t\beta} + B_t^i\bar{g}_z\boldsymbol{\sigma}\right) \tag{2.48a}$$

$$\dot{\boldsymbol{\sigma}}_r = -K_\phi\left(B_t\frac{K_{t\beta}}{\bar{g}_z}\boldsymbol{\eta}_{t\beta} + \boldsymbol{\sigma}\right) \tag{2.48b}$$

$$\ddot{\boldsymbol{\sigma}} = K_\omega\left(\dot{\boldsymbol{\sigma}}_r - \dot{\boldsymbol{\sigma}}\right). \tag{2.48c}$$

where $A = \left(I - B_\beta^i B_\beta^{i\dagger}\right)$. In Fig. 2.4 a graphical representation of the linearized system has been reported in order to highlight the structure of the problem.

2.4.3 Stability Proof

Given the structure of the control algorithm, the stability proof will be similar to the formal derivation of backstepping control technique.
To this aim, it is useful to introduce the following error quantity:

$$\boldsymbol{e}_1 = B_t\frac{K_{t\beta}}{\bar{g}_z}\boldsymbol{\eta}_{t\beta} + \boldsymbol{\sigma} \tag{2.49}$$

Notice that $\dot{\boldsymbol{\sigma}}_r = -K_\phi\boldsymbol{e}_1$. Thus, the system in (2.48) results in:

$$\dot{\boldsymbol{\eta}}_{t\beta} = -K_{t\beta}\boldsymbol{\eta}_{t\beta} + AB_t^i\,\bar{g}_z\boldsymbol{e}_1 \tag{2.50a}$$

$$\dot{\boldsymbol{e}}_1 = -B_t\frac{K_{t\beta}^2}{\bar{g}_z}\boldsymbol{\eta}_{t\beta} + K_{t\beta}\boldsymbol{e}_1 + \dot{\boldsymbol{\sigma}} \tag{2.50b}$$

$$\ddot{\boldsymbol{\sigma}} = K_\omega\left(-K_\phi\boldsymbol{e}_1 - \dot{\boldsymbol{\sigma}}\right). \tag{2.50c}$$

where the following property has been used.

Proposition 3. *Given the symmetric positive definite matrix* $B_{t\beta} = \begin{bmatrix} B_t \\ B_\beta \end{bmatrix}$ *and its inverse* ${B^{t\beta}}^{-1} = \begin{bmatrix} B_t^i & B_\beta^i \end{bmatrix}$*, such that* $B_t B_t^i = I$ *and* $B_\beta B_\beta^i = I$*, and defining* $A = I - B_\beta^i {B_\beta^i}^\dagger$*, it results:*

$$A\, B_t^i B_t = A \tag{2.51}$$

The proof of Proposition 3 is reported in Appendix B.6. Notice that both matrices A and $B_t^i B_t$ are idempotent matrices. We are now in position to prove the main stability result:

Theorem 2.4.1. *Given the system in* (2.50)*, with matrices* A*,* B_t^i *and* B_t *as defined in Proposition 3,* $\bar{g}_z \in \mathbb{R}$*, and* $K_{t\beta}, K_\phi, K_\omega$ *positive scalar gains, such that* $K_\omega > K_\phi$ *and* $K_\phi > 2K_{t\beta}$*, the system matrix* H_s *defined as follows:*

$$H_s = \begin{bmatrix} -K_{t\beta} I & A B_t^i\, \bar{g}_z & 0 \\ -B_t \frac{K_{t\beta}^2}{\bar{g}_z} & K_{t\beta} I & I \\ 0 & -K_\omega K_\phi I & -K_\omega I \end{bmatrix} \tag{2.52}$$

is Hurwitz, and the system in (2.50) *is asymptotically stable.*

Proof. System in (2.50) can be formulated in a way to make the backstepping-like inner structure explicit. Let's introduce the variables $\boldsymbol{w}_1 = \begin{bmatrix} \boldsymbol{\eta}_{t\beta}^T & \boldsymbol{e}_1^T \end{bmatrix}^T$ and consider the variables $\dot{\boldsymbol{\sigma}}$ as input to their subsystem:

$$\dot{\boldsymbol{w}}_1 = f_w(\boldsymbol{w}_1) + g_w\, \dot{\boldsymbol{\sigma}}_r + g_w (\dot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\sigma}}_r) \tag{2.53a}$$

$$\ddot{\boldsymbol{\sigma}} = K_\omega (\dot{\boldsymbol{\sigma}}_r - \dot{\boldsymbol{\sigma}}). \tag{2.53b}$$

where:

$$f_w(\boldsymbol{w}_1) = H_w \boldsymbol{w}_1, \quad H_w = \begin{bmatrix} -K_{t\beta} I & A B_t^i \bar{g}_z \\ -\frac{K_{t\beta}^2}{\bar{g}_z} B_t & K_{t\beta} I \end{bmatrix}$$

$$g_w = \begin{bmatrix} 0 & I \end{bmatrix}^T$$

Remind that $\dot{\boldsymbol{\sigma}}_r = -K_\phi \boldsymbol{e}_1$, which represents the reference signal for the roll and pitch velocities $\dot{\boldsymbol{\sigma}}$. The quantity $g_w\, \dot{\boldsymbol{\sigma}}_r$ has been added and subtracted to (2.53a) because we are able to find a Lyapunov function for the following subsystem:

$$\begin{aligned} \dot{\boldsymbol{w}}_1 &= f_w(\boldsymbol{w}_1) + g_w\, \dot{\boldsymbol{\sigma}}_r \\ &= \bar{H}_w \boldsymbol{w}_1 \end{aligned} \tag{2.54}$$

where:

$$\bar{H}_w = \begin{bmatrix} -K_{t\beta} I & AB_t^i \bar{g}_z \\ -\frac{K_{t\beta}^2}{\bar{g}_z} B_t & \left(K_{t\beta} - K_\phi\right) I \end{bmatrix}$$

A candidate Lyapunov function for the subsystem in (2.54) is in fact the following one:

$$\begin{aligned} V_1 &= \frac{1}{2} \boldsymbol{w}_1^T P_1 \boldsymbol{w}_1 \\ P_1 &= \begin{bmatrix} \frac{K_{t\beta}^2}{\bar{g}_z^2} \left(2K_\phi - K_{t\beta}\right) I & \frac{K_{t\beta}^2}{\bar{g}_z} AB_t^i \\ \frac{K_{t\beta}^2}{\bar{g}_z} B_t^{iT} A & B_t^{iT} AB_t^i \left(K_\phi - K_{t\beta}\right) \end{bmatrix} \end{aligned} \tag{2.55}$$

The following property is needed to prove the positive definitiveness of matrix P_1.

Proposition 4. *Given the matrices defined in Proposition* 3*, the product matrix* $B_t^{iT} AB_t^i$ *is symmetric positive definite.*

The proof of Proposition 4 is reported in Appendix B.7.

Proposition 5. *Given matrix* P_1 *as defined in* (2.55)*,* P_1 *is positive definite if* $K_\phi > \frac{3}{2} K_{t\beta}$.

The proof of Proposition 5 is reported in Appendix B.8.
The derivative of the Lyapunov function V_1, if we consider the reduced subsystem of (2.54), results in:

$$\dot{V}_1 = \frac{1}{2} \boldsymbol{w}_1^T \left(\bar{H}_w^T P_1 + P_1 \bar{H}_w\right) \boldsymbol{w}_1 = -\boldsymbol{w}_1^T Q_1 \boldsymbol{w}_1$$

$$Q_1 = \begin{bmatrix} \frac{K_{t\beta}^3}{\bar{g}_z^2} \left(2K_\phi - K_{t\beta}\right) I + A \frac{K_{t\beta}^4}{\bar{g}_z^2} & 0 \\ 0 & B_t^{iT} AB_t^i K_\phi \left(K_\phi - 2K_{t\beta}\right) \end{bmatrix}$$

Notice that matrix Q_1 is positive definite for $K_\phi > 2K_{t\beta}$. Remember that matrix A, being idempotent, has eigenvalues equal to either 1 or 0.
Then, a candidate Lyapunov function for the complete system in (2.53) is the following:

$$\begin{aligned} V &= V_1 + V_2, \\ V_1 &= \frac{1}{2} \boldsymbol{w}_1^T P_1 \boldsymbol{w}_1, \quad V_2 = \frac{1}{2} \boldsymbol{e}_2^T P_2 \boldsymbol{e}_2 \end{aligned}$$

where:

$$\boldsymbol{e}_2 = \dot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\sigma}}_r = \dot{\boldsymbol{\sigma}} + K_\phi \boldsymbol{e}_1, \tag{2.56}$$

and P_2 is a symmetric positive definite matrix to be defined. The derivative of function V_1, considering the system in (2.53), is:

$$\begin{aligned}\dot{V}_1 &= \boldsymbol{w}_1^T P_1 \dot{\boldsymbol{w}}_1 = \boldsymbol{w}_1^T P_1 \Big(f_w(\boldsymbol{w}_1) + g_w \, \dot{\boldsymbol{\sigma}}_r + g_w (\dot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\sigma}}_r) \Big) \\ &= \boldsymbol{w}_1^T P_1 \left(\bar{H}_w \boldsymbol{w}_1 + g_w \boldsymbol{e}_2 \right) \\ &= -\boldsymbol{w}_1^T Q_1 \boldsymbol{w}_1 + \boldsymbol{w}_1^T P_1 g_w \boldsymbol{e}_2\end{aligned}$$

where:

$$P_1 g_w = \Big[B_t^{iT} A \tfrac{K_{t\beta}^2}{\bar{g}_z} \quad B_t^{iT} A B_t^i \left(K_\phi - K_{t\beta} \right) \Big]^T$$

The derivative of the other component V_2 of the Lyapunov function V is:

$$\dot{V}_2 = \boldsymbol{e}_2^T P_2 \dot{\boldsymbol{e}}_2 = \boldsymbol{e}_2^T P_2 \Big(- K_\omega \boldsymbol{e}_2 + K_\phi \dot{\boldsymbol{e}}_1 \Big)$$

where the derivative of $\boldsymbol{e}_2$ is computed reminding the definition in (2.56) and the relation in (2.48c). Then, the derivative of $\boldsymbol{e}_1$ can be obtained by applying the relation in (2.50b) and substituting $\dot{\boldsymbol{\sigma}} = \boldsymbol{e}_2 - K_\phi \boldsymbol{e}_1$. It results:

$$\dot{V}_2 = -\boldsymbol{e}_2^T Q_2 \boldsymbol{e}_2 - \boldsymbol{e}_2^T P_2 L \, \boldsymbol{w}_1$$

where:

$$\begin{aligned} Q_2 &= P_2 \left(K_\omega - K_\phi \right) \\ L &= K_\phi \Big[B_t \tfrac{K_{t\beta}^2}{\bar{g}_z} \quad \left(K_\phi - K_{t\beta} \right) I \Big] . \end{aligned}$$

Notice that Q_2 is positive definite for $K_\omega > K_\phi$. Finally, the derivative of the Lyapunov function V results:

$$\begin{aligned}\dot{V} &= \dot{V}_1 + \dot{V}_2 \\ &= -\boldsymbol{w}_1^T Q_1 \boldsymbol{w}_1 - \boldsymbol{e}_2^T Q_2 \boldsymbol{e}_2 + \boldsymbol{e}_2^T \Big((P_1 g_w)^T - P_2 L \Big) \boldsymbol{w}_1\end{aligned}$$

If the matrix P_2 is chosen as follows:

$$P_2 = \frac{1}{K_\phi} B_t^{iT} A B_t^i$$

it results $(P_1 g_w)^T - P_2 L = 0$. Again, the property in Proposition 3 has been used.
Finally, the derivative of the Lyapunov function V results in:

$$\dot{V} = -\boldsymbol{w}_1^T Q_1 \boldsymbol{w}_1 - \boldsymbol{e}_2^T Q_2 \boldsymbol{e}_2$$

and, since Q_1 and Q_2 are symmetric positive definite matrices for $K_\omega > K_\phi$ and $K_\phi > 2K_{t\beta}$, it is proved that the linearized system is asymptotically stable. This completes the proof of the Theorem. $\square$

The stability analysis has been performed for a steady state equilibrium point with assumptions on the perfect knowledge of the model and of the state of the system. The robustness of the control system in a general flight condition will be analyzed in the following Section and tested via a simulation campaign.

2.4.4 Discussion on the Robustness to Inertia Parameters Uncertainty

The adopted control strategy, similar to the common inverse dynamics techniques, can show lack of robustness with respect to parameters uncertainty. The goal of the present Section is to analyze the stability property of the closed-loop system under uncertainty on inertia parameters.

A first strategy that can be adopted is to test via simulation the system, modeling the effects of an unbalanced mass on the aerial manipulator. Obviously, due to the non linearity of the system, a number of experiments in different configurations have to be performed to have a reasonable confidence on the stability property. A second strategy consists in the analysis of the eigenvalues of the closed-loop system linearized around a chosen equilibrium point, when inertia matrix is subject to perturbations. Although the result has just local applicability, uncertainty bounds can be computed with more accuracy. Finally, in order to obtain a conservative value of the mass that can be added while preserving stability, a Lyapunov approach can be employed as it will be further detailed in the following.

Simulation Results

The simulation platform employed to the test the proposed control on the aerial manipulator is extensively described in Section 2.5. The system has been tested with a concentrated mass added on the quadrotor (in its center of gravity) and with a concentrated mass added at the end effector. A number of simulations have been performed in different configurations, and the approximated maximum values of added mass to preserve stability turn out to be 2.5 kg when it is added to the quadrotor, and 0.04 kg when added at the end effector. Notice that a mass of 2.5 kg represents the 125% of the mass of the quadrotor, while a mass of 0.04 kg corresponds to the 60% of the nominal mass of the robotic arm last link.

Linear System subject to Perturbation

The dynamic system in (2.50), whose system matrix H_s is reported in (2.52), is the linear approximation of the closed-loop system describing the aerial manipulator controlled with the inverse dynamics method described in the present Chapter. Organizing state variables as $x = \begin{bmatrix} \boldsymbol{e}_1^T & \boldsymbol{\eta}_{t\beta}^T & \dot{\boldsymbol{\sigma}}^T \end{bmatrix}^T$, and computing the corresponding system matrix $\bar{H}_s = \begin{bmatrix} \bar{H}_{s,e_1} \\ \bar{H}_{s,2} \end{bmatrix}$, the following equation can describe the effect of the uncertainty on inertia matrix:

$$\dot{x} = \left(\bar{H}_s + E\right) x \tag{2.57}$$

Matrix E is defined as follows:

$$E = \begin{bmatrix} 0 \\ \left[\left(I + B^{-1}\Delta B\right)^{-1} - I \right] \bar{H}_{s,2} \end{bmatrix},$$

where matrix $\Delta B \in \mathbb{R}^{n\times n}$ represents the difference between the real inertia matrix $B_r \in \mathbb{R}^{n\times n}$ and the estimated inertia matrix used in the control algorithm $B_r = \Delta B + B$. A simple heuristic method to obtain a reasonable value of the critical additional mass is to identify perturbation matrices with the structure $\Delta B = m\bar{\Delta B}$, and compute the eigenvalues of matrix $\bar{H}_s + E$ with increasing values of mass m. It is implicitly assumed that the perturbation is caused by a concentrated mass m, and the matrix $\bar{\Delta B}$ depends just on the position of the added mass.

The system has been tested around an equilibrium configuration. The obtained values for the maximum added mass on the quadrotor and on the robotic arm end effector to preserve stability are equal to 3.4 kg and 0.085 kg respectively.

Lyapunov based Approach

A different approach can be applied in order to obtain a conservative value of the maximum additional mass which preserves stability, without the need of performing a series of numerical simulation. The method is reported in [28] for a general linear system as in (2.57), where a Lyapunov function

P exists, such that:

$$P\bar{H}_s + \bar{H}_s^T P = -2Q$$

for a symmetric positive definite matrix Q. In particular, when perturbation matrix E is structured as the weighted sum of r matrices E_i :

$$E = \sum_{i=1}^{r} p_i E_i,$$

the system in (2.57) is stable if the following relation holds:

$$\sum_{i=1}^{r} |p_i|^2 < \frac{4\sigma^2_{\min(Q)}}{\sum_{i=1}^{r} \mu_i^2}, \tag{2.58}$$

where $\sigma_{\min(Q)}$ is the minimum singular value of Q and μ_i is given by:

$$\mu_i = \left\| E_i^T P + P E_i \right\|.$$

where $\| \cdot \|$ is the 2-norm of the Frobenius norm. Notice that, when $r = 1$, then a single perturbation matrix is considered, condition in (2.58) reduces to the simple following relation (it can be assumed $p_1 = 1$):

$$\left\| E^T P + PE \right\| < 2\sigma_{\min(Q)}. \tag{2.59}$$

The goal of the present Section is to devise, starting form relation in (2.59), a simple condition on the maximum additional mass preserving stability for the dynamical system under study. Notice that matrices P and Q for the system in (2.50) can be easily computed by matrices P_1, P_2, Q_1 and Q_2 defined in the previous Section.

The left member of (2.59) can be simplified as follows:

$$\left\| E^T P + PE \right\| < 2 \left\| E \right\| \left\| P \right\|$$

Similarly, the following relation holds on the norm of matrix E:

$$\|E\| = \left\| \left[\left(I + B^{-1}\Delta B \right)^{-1} - I \right] \bar{H}_{s,2} \right\| \leq \left\| \left(I + B^{-1}\Delta B \right)^{-1} - I \right\| \left\| \bar{H}_{s,2} \right\|$$

Notice that, for a matrix $C \in \mathbb{R}^{n\times n}$, $(I + C)^{-1} - I = -C\,(I + C)^{-1}$. Then, $(I + B^{-1}\Delta B)^{-1} - I = -B^{-1}\Delta B\,(I + B^{-1}\Delta B)^{-1}$ and the norm of E can be decomposed as follows:

$$\|E\| \leq \left\| B^{-1}\Delta B \right\| \left\| \left(I + B^{-1}\Delta B \right)^{-1} \right\| \left\| \bar{H}_{s,2} \right\|$$

Additionally, the following property is used [93]:

Proposition 6. *Given an invertible matrix T, such that $\|T - I\| < 1$, then $\|T^{-1}\| \leq \frac{1}{1-\|T-I\|}$.*

Therefore, if $\|B^{-1}\Delta B\| < 1$, it results:

$$\|E\| \leq \frac{\|B^{-1}\Delta B\|}{1 - \|B^{-1}\Delta B\|} \; \|\bar{H}_{s,2}\|$$

In order to separate the geometrical information of ΔB from the magnitude of the additional mass, inertia matrix perturbation can be defined as described above: $\Delta B = \bar{\Delta B}\, m$. Finally, the relation in (2.59) can be employed to compute a conservative estimate of the maximum additional mass the drives the system to instability.

$$m < \frac{\sigma_{\text{min}}(Q)}{\left\|B^{-1}\bar{\Delta B}\right\| \left(\|P\| \left\|\bar{H}_{s,2}\right\| + \sigma_{\text{min}}(Q)\right)}$$

The relation can be written highlighting the minimum eigenvalue $\lambda_{\text{min}}(B)$ of inertia matrix B:

$$m < \frac{\sigma_{\text{min}}(Q)\, \lambda_{\text{min}}(B)}{\left\|\bar{\Delta B}\right\| \left(\|P\| \left\|\bar{H}_{s,2}\right\| + \sigma_{\text{min}}(Q)\right)} \tag{2.60}$$

The relation is conservative, but it allows to directly compute a maximum value of an additional concentrated mass, given the matrices of the system.

Unfortunately, in the case of the system in (2.50), the results are too conservative. As an example, the value $2\,\|E\|\,\|P\|$ that is used as a conservative estimate of $\left\|E^T P + PE\right\|$, results to be 24 times bigger than the norm $\left\|E^T P + PE\right\|$, for the chosen configuration and a reasonable perturbation. In fact, the computed limit mass is equal to $2\,10^{-3}$ kg and $4\,10^{-5}$ kg for the mass added to the quadrotor and at the robot end effector respectively. For different systems, the method can be useful to fast evaluate the effect of an additional mass on stability. Inequality (2.60) thus has conceptual value rather than a practical role. Further refinements on the robustness conditions should be performed in future works in order to obtain conditions of practical use.

2.5 Simulation Experiments

In this Section, we show the validation of the proposed control system on a simulation platform. A first set of simulations aims at demonstrating the stability of the system in operating points different from the hovering condition and in presence of non ideal actuation, model and state knowledge.

The performance obtained with the proposed method will be compared with a different state of the art control approach. In addition, a second set of experiments is shown to demonstrate the applicability of a simple impedance control scheme to the proposed inverse dynamics control, in order to perform a task requiring the interaction with the environment.

2.5.1 Simulation Platform and Model Description

Matlab was used for simulations and a recursive Newton-Euler dynamics model of the manipulators was implemented using the Robotics Toolbox [25]. The model analyzed is composed of a 6 DoFs quadrotor and a 4 DoFs robotic arm attached at its base. The Denavit-Hartenberg parameters of the arm are reported in Tab. 2.1. The mass of the quadrotor and the

i	$a_{i-1}\,[m]$	α_{i-1} [deg]	$d_i\,[m]$	θ_i
1	0	+90	0.1143	θ_1
2	0.1970	0	0	θ_2
3	0	-90	0	θ_3
4	0.0800	0	0.1225	θ_4

Table 2.1: *Denavit-Hartenberg Parameters of the Robotic Arm.*

robotic arm are $2.00\,\text{kg}$ and $0.80\,\text{kg}$, respectively. The control algorithm proposed in Sec. 2.3 has been implemented with minor differences. The Coriolis matrix in the new reference system C has been approximated to the value $\bar{C} = T^{-T}C_oT^{-1}$, neglecting the part depending on the derivative of the matrix T. In fact, in the real system this component will be neglected to speed up the computation. In addition, as those approximations are made just in the control part, they can simulate a non perfect knowledge of the system, or an equivalent disturbance.
The quadrotor actuation is performed by setting the four rotor velocities, while saturations on rotor velocities have been imposed. Bounds on rotor velocities obviously imply an equivalent saturation on thrust and torques that can be provided by the quadrotor.
In addition to torque saturation, the simulator can model the typical estimation errors of the sensing systems applied to common small UAVs.
Quadrotor propeller motion, and consequently quadrotor actuation, is affected by flapping effect, modeled accordingly to [22] and implemented in [90].

2.5.2 Experiment in Free Motion

Description of the Designed Experiment

The experiment consists in providing a robot joints reference trajectory $\boldsymbol{\xi}_r$ in order to track a prescribed end effector translational trajectory. The redundancy resolution is solved by a quadratic programming optimization algorithm, capable of generating on line the joint trajectory while minimizing different tasks (the method applied for the generation of joint trajectory is presented in Section 3.2). The tracking error is passed in feedback to the optimizator, in order to compensate possible errors in the kinematics. The relative tracking gain is sufficiently low with respect to the inner control loops such that time-scale separation principle holds.
The scheme of the complete simulation structure is reported in Fig. 2.5.

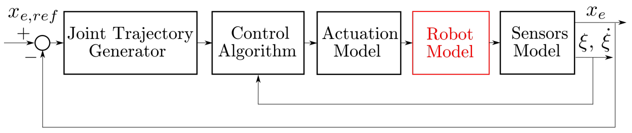


Figure 2.5: *Complete scheme of the simulation structure.*

The end effector trajectory is composed of three main stages:

First stage In the first stage, a straight line trajectory in forward direction (x) with velocity trapezoidal profile is passed as a reference. The system will perform a displacement of 8.7 m, with a pick velocity of $0.4\,{}^{\mathrm{m}}\!/\!_{\mathrm{s}}$. The optimization solution is forced such that the trajectory is performed just with quadrotor DoFs, while keeping constant the arm joint reference. The goal of this experiment stage is to show the effect of a step variation in translational acceleration of the quadrotor, generating a pitch motion, on the end effector error.

Second stage In the second stage, the end effector has to perform a small amplitude sinusoidal trajectory on y direction. The end effector performs a displacement of 0.10 m and the reference sinusoid angular frequency is $1\,{}^{\mathrm{rad}}\!/\!_{\mathrm{s}}$. Similarly to the previous stage, the optimization solution is forced to perform the motion with just arm joints motion. This stage is designed to show the effect of the arm motion on the quadrotor. While performing the motion, the arm center of mass position is changing, generating inertial forces on the quadrotor.

K_ξ	$K_{t\beta}$	K_ϕ	K_ω
2	10	50	250

Table 2.2: *Gains of the Control Loop*

Third stage The last stage is just a super imposition of the two trajectories, in order to show the performance of the system when all the joints are moving and with operating points far from hovering condition.

The dynamic response of the system is set by means of the gains K_ξ, $K_{t\beta}$, K_ϕ and K_ω. The values chosen for the performed simulation are reported in Tab. 2.2, corresponding to an equivalent end effector control bandwidth for quadrotor and joint positions of 2.00 rad/s, reasonable for the real system. The feed-forward coefficient $K_{\xi_{ff}}$ is equal to 1.
The proposed control system will be compared with a state of the art control as described in [21], without the uncertainties estimation component of such method. In particular, the reference method is an inverse dynamics control, where roll and pitch variables are used as control variables. Note that in the work the mentioned uncertainty is compensated with an adaptive approach which will not be considered in the implemented comparison method. By so doing, it is possible to analyze the phenomenons that are compensated by the proposed approach. For simplicity, we will call this reference method standard inverse dynamics control (sID). The difference between the two methods is represented exactly by the contribution claimed by this work: the dynamical and kinematic decoupling obtained by controlling the system in the new variables (the change of variables reported in (2.5)), and the minimization of disturbances by applying the optimal torques computed in (2.23). Obviously, the two methods will be compared by applying the same control gains.

Results and Discussion

The first experiment is performed with perfect knowledge of the state, in order to cleanly analyze the performance of the method.
The end effector trajectory performed in the three stages is reported in the upper box of Fig. 2.6. Reference trajectory is reported with dashed lines, while solid lines represent the actual displacement. The two groups of lines are almost indistinguishable. Notice that x displacement is shown with different scale.
In the lower box of Fig. 2.6 the tracking error is reported. In Fig. 2.7 the 3D plot of the robot configurations during four instants of the experiment

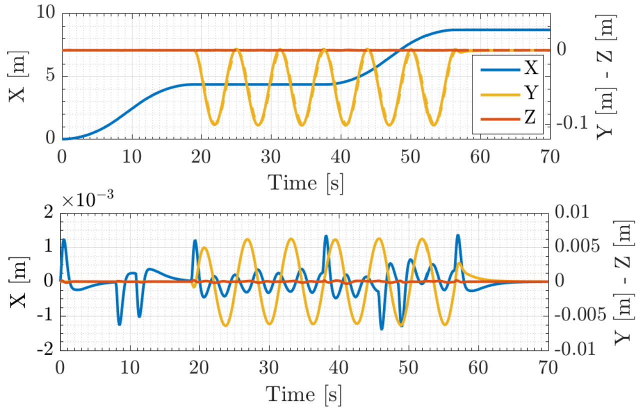


Figure 2.6: *End effector trajectory (upper figure) and tracking error (lower figure) during the experiment. In the upper figure solid lines show the actual displacement, while dashed lines represent the reference (they are almost indistinguishable). Notice that x displacement, in both figure, has different scale.*

have been reported. It is interesting to analyze the performance in the three stages, and compare the results obtained with the sID method.

First stage The trapezoidal velocity reference profile implies an initial step variation of required acceleration and consequently a step variation of pitch angle. This situation is useful to analyze the effect of fast pitch (and roll) variations on end effector position. In Fig. 2.8 the end effector behavior during the first 0.5 s of the experiment has been reported. In particular, solid lines represent the actual motion of the end effector, dashed line represents the x reference trajectory, dotted line represents the end effector motion obtained with the sID control method, and starred line is the theoretical response that should be obtained with the given control gains. The theoretical response is obtained by computing the output of the following linear transfer function, excited with the x reference position:

$$F_{Th} = \frac{s + K_\xi}{s^2/K_{t\beta} + s + K_\xi}. \tag{2.61}$$

This transfer function simply represents a cascade control with gains K_ξ and $K_{t\beta}$, unitary feed-forward, applied to a system represented by two integrators.
It is interesting that the sID method shows a behavior similar to a non minimum phase system, typical of aerial manipulator systems. In fact, a positive pitch rotation, in an East-North-Up (ENU) frame, is required to obtain

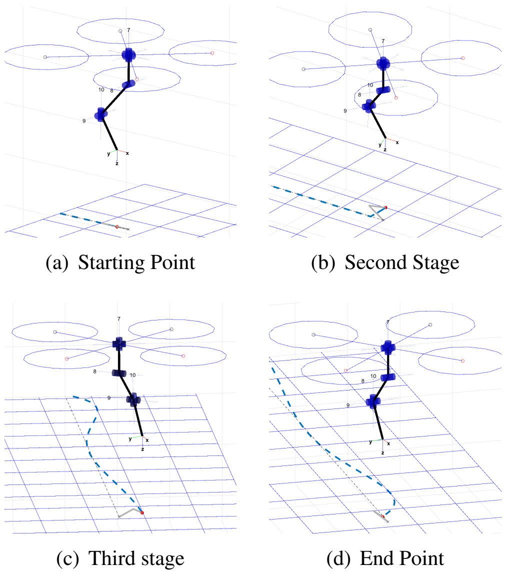


Figure 2.7: *Different frames of the robot configuration along the experiment. The dashed blue line on the floor reports the performed trajectory of the end effector on x and y axes. Grey line on the floor is the projection of arm links on x and y axes.*

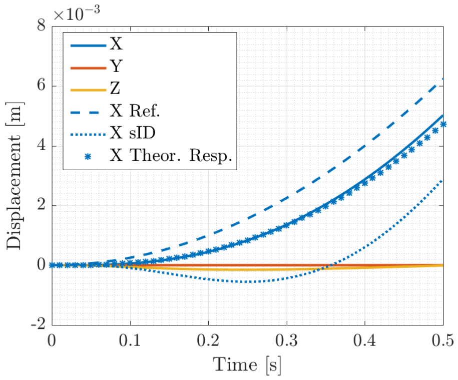


Figure 2.8: *Detail of the first instants of end effector trajectory. Solid lines represent the actual motion of the end effector, dashed line represents the x reference trajectory, dotted line represents the x end effector motion obtained with the sID control method, and starred line is the x theoretical response that should be obtained with the given control gains.*

a positive acceleration of the quadrotor in x direction, but when it is applied to a system with an arm extended towards the floor, it results in a negative x displacement of the end effector. Instead, with the proposed method the end effector does not show this behavior and it follows the ideal response, by keeping as low as possible the tracking error. On z and y directions very low disturbances can be appreciated.
The physical explanation is that the proposed control can cancel the effect of pitch and roll on robot arm center of mass and end effector position. So, quadrotor can accelerate without producing any disturbances either on the goal, the end effector position, or on the center of mass of the robot, which would cause undesired dynamics coupling. This is obtained by a variation of controllable variables, robot arm joints, yaw and quadrotor z position, which perform a motion while the control keeps the variables $\boldsymbol{\beta}$ to zero.
In Fig. 2.9 the trajectory of the second and third arm joints is reported, together with pitch behavior. In the algorithm a weight matrix W_T has been applied, as defined in (2.4), such that roll and pitch motion is mainly compensated by robot arm joints with respect to yaw and z degrees of freedom. As it can be noticed by equation (2.17), if roll or pitch are different from zero, at the equilibrium $\boldsymbol{\xi} \neq \boldsymbol{\xi}_r$. This is why the robot joints keep a constant difference with respect to the reference positions.
In Fig. 2.10, the comparison between the error obtained with the proposed

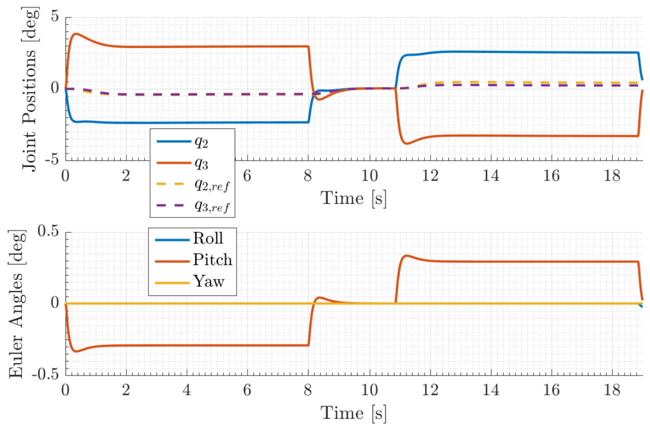


Figure 2.9: *Second and third arm joints trajectory (upper figure) and Euler angles (lower figure) during the first stage of the experiment.*

method and the sID method is reported. It turns out that the rates between the maximum tracking error of the proposed method and sID is equal to about one third (35%) and one tenth (12%) for x and z directions, respec-

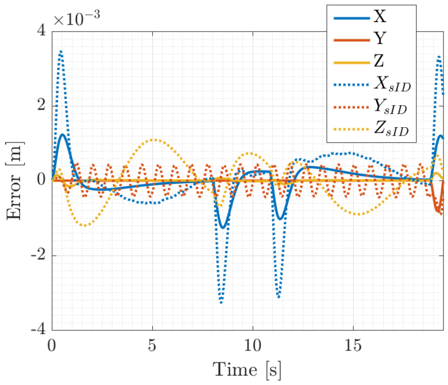


Figure 2.10: *End effector tracking error for the proposed method (solid lines) and sID method (dotted lines) during the first stage of the experiment.*

tively. Remarkably, with the sID method, in y direction an undamped sinusoidal perturbation is triggered, while the proposed method is able to keep y displacement of the end effector at zero. Notice that the magnitude of the tracking error is small because it is related to a small value of pitch, $0.3°$. When higher quadrotor acceleration are imposed, or external forces and disturbances are applied, higher pitch angles are required and the benefit of the proposed method increases accordingly. For example, with a pitch of just $3°$, the proposed control would show a lower tracking error of about 2.5 cm. Five experiments have been simulated, with different initial acceleration, and for each of them the maximum tracking error for the x end effector coordinate has been computed. The maximum errors for the proposed method and sID method have been reported in Fig. 2.11. Notice that the ratio between maximum errors is almost constant.

Second stage The goal of the second stage of the experiment is to analyze the effect of robot arm motion, and consequently the motion of its center of mass, on quadrotor position.
In Fig. 2.12 the detail of quadrotor position tracking error during the second stage of the experiment is reported, compared with the sID performance. Notice that disturbances on quadrotor position are absent, especially if compared with sID method. This result is obtained thanks to the approach in (2.23). In fact, torques are computed in order to minimize the acceleration error, considering the under-actuation on quadrotor translational degrees of freedom.

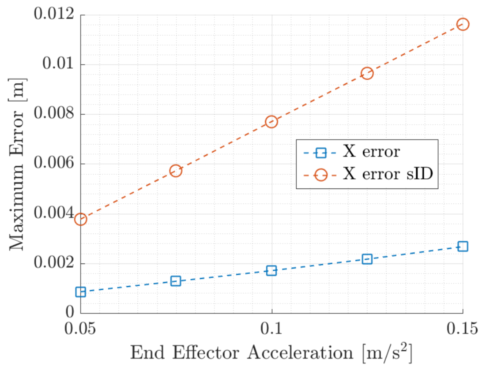


Figure 2.11: *Maximum tracking error in x direction during the initial phase of the experiment, for different values of initial end effector acceleration. The blue line represents the maximum error obtained with the proposed method, while red line represents the maximum error obtained with the sID method.*

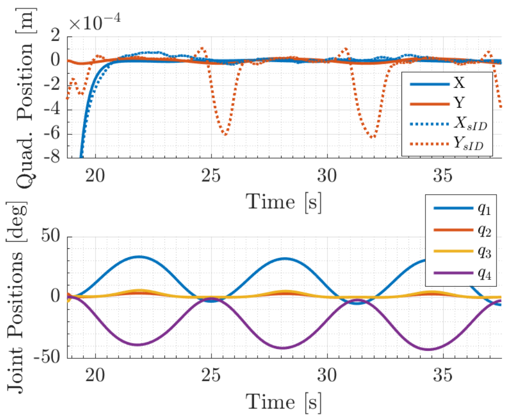


Figure 2.12: *Quadrotor position tracking error in the horizontal plane and joints trajectory during the second stage of the experiment. Dotted lines represent quadrotor position error with the sID method.*

Third stage The third stage of the experiment is designed to test the stability of the system in operating points far from steady condition. In fact, in Sec. 2.4 the stability was proven for a regulation problem, without guarantees on the tracking problem. In this stage both the arm and the quadrotor perform a motion that tests the stability of the whole system. In Fig. 2.13, the trajectory of arm joints and quadrotor Euler angles is reported during the third stage of the experiment. It is interesting to notice the super-position of the joint trajectory to track the reference and the joint displacement to compensate pitch variations. The system is stable and the control shows good performance. Remind that the tracking error during the complete experiment has been reported in Fig. 2.6.

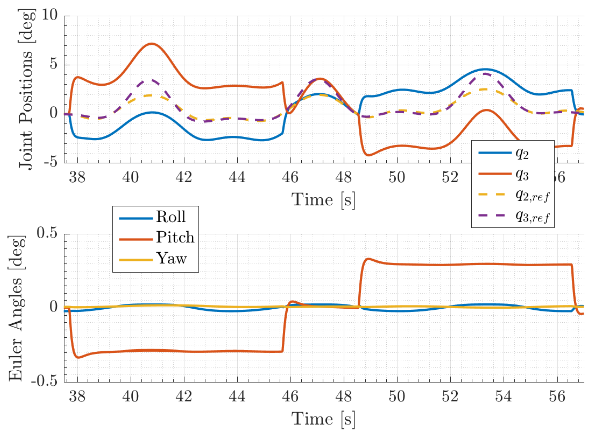


Figure 2.13: *Second and third arm joints trajectory (upper figure) and Euler angles (lower figure) during the third stage of the experiment.*

Finally, the system has been tested with typical estimation errors for sensing systems commonly installed on UAVs. In particular, a Gaussian error with standard deviation of 0.005 m and 0.02 $^{\mathrm{m}}/_{\mathrm{s}}$ on quadrotor position and velocity have been added, respectively. In addition, a first order low pass filter has been applied to sensor measures in order to obtain a smooth estimated profile as input to the control function. The resulting end effector trajectory and tracking error are reported in Fig. 2.14. The system is stable also with estimation errors. From the previous analysis it is clear that the tracking error derives from the estimation inaccuracy. So, the more accurate is the sensing systems, the closer the response will be to the theoretical linear response of (2.61).

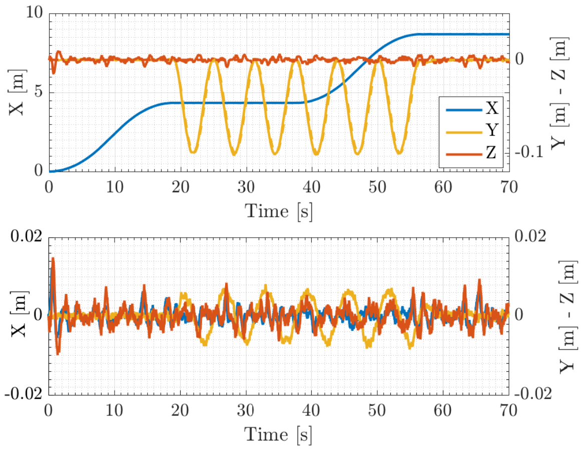


Figure 2.14: *End effector trajectory (upper figure) and tracking error (lower figure) during the experiment with estimation errors. In the upper figure solid lines show the actual displacement, while dashed lines represent the reference. Notice that x displacement in the upper box has different scale.*

Remark As it can be noticed in Fig. 2.7, the first joint of the robot arm is aligned with the z axis of the quadrotor. Thus, yaw quadrotor degree of freedom and the first arm joint are redundant. From this reason, the minimum number of arm joints required to track the end effector trajectory is 4, in order to have matrix $J_{ec1,\alpha}$ full rank. A higher number of joints would be beneficial to keep this matrix full rank in a larger joint configurations set and reduce the required displacement of robot joints for unitary variation of roll and pitch angles.

2.5.3 Application to an Interaction Task

A second series of experiments are addressed to demonstrate the applicability of the proposed inverse dynamics control to tasks performed in contact with the environment. To this end, a simple impedance control scheme is devised on top of the proposed inverse dynamics control, and simulations are performed to test the response of the system to external forces applied at the end effector.

Integration of an Impedance Control Scheme

In the literature, a number of works applied impedance control strategies to aerial manipulators in order to deal with the interaction with the external environment. We here apply the approach in [68], adapting it to be consistent with the proposed inverse dynamics control. In particular, the Cartesian impedance control substitutes the so-called feedback kinematic controller presented in Sec. 2.3.3, corresponding to block K in Fig. 2.2. In order to assign a desired dynamics to the end effector, while simultaneously stabilizing the motion in the null space of the main task, the following control action $\boldsymbol{u}_{t\beta}$ can be applied:

$$\boldsymbol{u}_{t\beta} = \boldsymbol{u}_{t\beta,C} + P\boldsymbol{u}_{t\beta,N} \tag{2.62}$$

Matrix P is a projection matrix, while $\boldsymbol{u}_{t\beta,C}$ and $\boldsymbol{u}_{t\beta,N}$ are computed as follows:

$$\boldsymbol{u}_{t\beta,C} = J_{t\beta}^{\dagger} M_d^{-1}(M_d \ddot{\boldsymbol{x}}_{e,r} + D_d \dot{\tilde{\boldsymbol{x}}}_e + K_d \tilde{\boldsymbol{x}}_e - M_d \dot{J}_{t\beta}\boldsymbol{\eta}_{t\beta}) \tag{2.63}$$

$$\boldsymbol{u}_{t\beta,N} = K_{t\beta}\left(\boldsymbol{\eta}_{t\beta,r} - \boldsymbol{\eta}_{t\beta}\right) \tag{2.64}$$

where $\tilde{\boldsymbol{x}}_e = \boldsymbol{x}_{e,r} - \boldsymbol{x}_e$ is the end effector position errors, $\boldsymbol{x}_{e,r}$ the end effector position reference, M_d, D_d and K_d the inertia, damping and stiffness of the desired dynamics (for simplicity we will assume that they are scalar values), and $J_{t\beta}$ is the submatrix of the end effector Jacobian $J_{e\eta}$ defined in (2.8):

$$\dot{\boldsymbol{x}}_e = J_{e\eta}(\boldsymbol{\xi})\boldsymbol{\eta}$$

$$J_{e\eta} = \begin{bmatrix} J_{e\eta,t} & 0 & J_{e\eta,\beta} \end{bmatrix}, \quad J_{t\beta} = \begin{bmatrix} J_{\eta,t} & J_{\eta,\beta} \end{bmatrix}.$$

Matrix P is projecting the control action $\boldsymbol{u}_{t\beta,N}$ in the null space of the end effector jacobian $J_{t\beta}$:

$$P = I - J_{t\beta}^T (J_{t\beta} J_{t\beta}^T)^{-1} J_{t\beta}.$$

Notice that variable $\boldsymbol{\eta}_{t\beta,r}$, computed as in (2.17), is the reference of motion of the aerial manipulator in the null space of the end effector.

Under the assumptions enforced above, about perfect knowledge of the model and of the state, and considering that the attitude controller is sufficiently fast with respect to the desired dynamics imposed by coefficients M_d, D_d and K_d, the following relation is obtained:

$$\dot{\boldsymbol{\eta}}_{t\beta} = \boldsymbol{u}_{t\beta} + B_{t\beta}^{-1} J_{t\beta}^T \boldsymbol{f}_e$$

Performing simple mathematical operations, the relation governing the end effector dynamics can be computed:

$$M_d \ddot{\tilde{\boldsymbol{x}}}_e + D_d \dot{\tilde{\boldsymbol{x}}}_e + K_d \tilde{\boldsymbol{x}}_e = M_d B_e^{-1}(\boldsymbol{\xi})\boldsymbol{f}_e \tag{2.65}$$

where B_e is the reflected mass of the aerial manipulator at the end effector:

$$B_e^{-1}(\boldsymbol{\xi}) = J_{t\beta}(\boldsymbol{\xi}) B_{t\beta}(\boldsymbol{\xi})^{-1} J_{t\beta}^T(\boldsymbol{\xi}).$$

(2.65) shows that the desired dynamic behavior is obtained. However, the relation between force at the end effector and corresponding displacement depends on the reflected mass $B_e(\boldsymbol{\xi})$ too. Notice that eigenvalues of $B_e(\boldsymbol{\xi})$ are upper and lower bounded, as Jacobian $J_{t\beta}$ is a full rank matrix for every robot configuration.

Interaction Experiments

The impedance control scheme has been tested in two different scenarios. First, a step variation of the force at the end effector has been applied. Second, the interaction with an infinitely rigid surface has been studied.

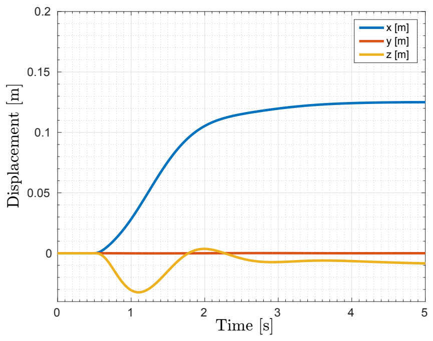
The following numerical values have been selected.

$$M_d = 1.0, \quad D_d = 4.0, \quad K_d = 3.6 \tag{2.66}$$

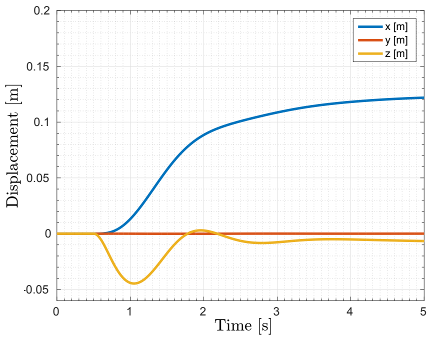
Impedance dynamic parameters have been chosen such that the poles of the equivalent mass-spring-damper system are complex conjugate, with natural frequency equal to $2\frac{\text{rad}}{\text{s}}$ and damping coefficient 0.9.

First Experiment: external force applied to the end effector A step variation of the force at the end effector is applied along the axis x, with a magnitude of 0.1 N. In Fig. 2.15, the response of the end effector and the behavior of all the degrees of freedom of the aerial manipulator are reported. As expected, the end effector shows a settling time equal to $T_s \simeq \frac{5}{2 \cdot 0.9} = 2.8$ s, see Fig. 2.15(a). In Fig. 2.15(c) it can be noticed that pitch angle asymptotically tends to a constant vale, in order to apply the equilibrium force in x direction. Thus, the system evolves with the desired dynamic behavior and the stability is preserved.

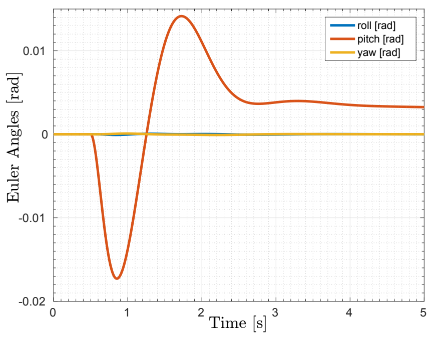
Second Experiment: robot in contact with a rigid surface In the second simulation, a position reference is given to aerial manipulator, such that the end effector comes into contact with a rigid surface, whose normal vector is directed in x direction. The behaviors of the end effector and degrees of freedom of the aerial manipulator are reported in Fig. 2.16. Notice that the end effector, after some bounces, can establish a stable contact with the surface, see Fig. 2.16(a). Higher values of the pitch angle with respect the first experiment are observed, due to the impact with the surface.



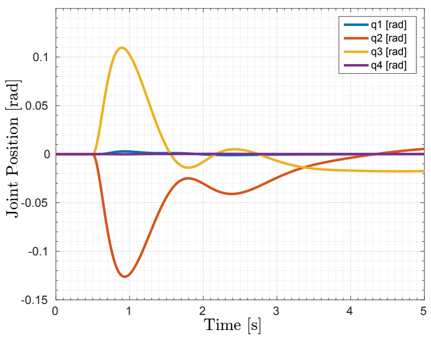
(a) End-effector position



(b) Quadrotor position

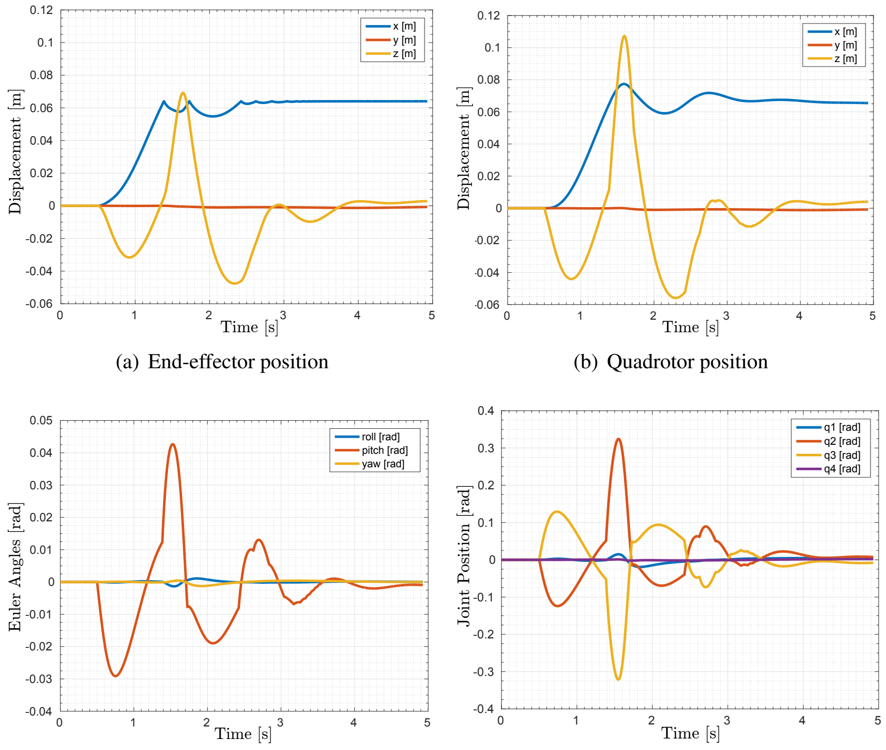


(c) Quadrotor attitude



(d) Robotic arm joints position

Figure 2.15: *Response of the aerial manipulator to a force step variation at the end effector.*



(a) End-effector position

(b) Quadrotor position

(c) Quadrotor attitude

(d) Robotic arm joints position

Figure 2.16: *Response of the aerial manipulator when in contact with a rigid surface.*

Final Remarks

The proposed inverse dynamic control, integrated with an impedance control scheme, can effectively cope with external forces applied at the end effector, preserving stability. Notice that the discrepancy of the system behavior with respect to the desired response dynamics, selected with the parameters M_d, D_d and K_d, is due to the coupling term $B_e(\boldsymbol{\xi})$. In fact, it is well-known that, in order to assign the desired inertia to the closed-loop system with an impedance control strategy, a measure or an estimation of the applied force is necessary.

2.6 Discussion

The proposed control approach is based on a novel variables representation of the system. Decoupling the attitude of the flying base from the robotic arm center of mass and end effector position, roll and pitch variables can be employed as control variables, similar to control approaches for quadrotor aerial vehicles without an attached robotic arm. An inverse dynamics control conceived on the basis of the proposed change of variables, is developed. The control strategy is proven to be stable by means of Lyapunov analysis. Notice that the method is model-based, thus its effectiveness can be decreased by inaccurate sensors information, errors in model parameters, communication delays, and unmodeled external disturbances. However, the improvement in the tracking performance is shown by simulation results, which validate its stability in various operating points in free motion and performing a simple interaction task.

CHAPTER 3

Trajectory Generation with Redundancy Resolution

3.1 Introduction

Aerial Manipulators have to accomplish the specific mission and tasks they are used for. For instance, the end effector of the system should track a prescribed path in order to point a particular sensor towards a target surface. In general, the structure of aerial manipulators is characterized by a kinematic redundancy with respect to the end effector position and orientation. Thus, infinite different motion solutions to perform the required mission exist. On the other hand, the complexity of the physical structure of the system imposes specific constraints and bounds to the motion. For example, self-collisions must be avoided, and the arm should assume configurations that do not compromise stability of the flying base. Therefore, the generation of a trajectory for all the variables of the system, which allows to accomplish the required mission while satisfying specific constraints and bounds, is a complex problem.

In the present Chapter two different methods are proposed, which can solve the redundancy of the system by means of optimization algorithms.

State of the art algorithms for the solution of complex optimization problems, together with the available miniaturized and powerful commercial CPUs, are the technical and technological assets that allow the implementation and resolution of such complex problems as trajectory generation for UAMs on line and on board. The on line implementation enables the system to change the mission strategy during different phases of the mission, or as a response to changes in the surrounding environment, like the presence of a unforeseen obstacle, detected with on board sensors. Moreover, when the computation is performed on the platform itself, the system turns out to be more robust to communication losses with the ground station.

In the following, the two proposed systems will be compared with the state of the art methods in the field. Then, the characteristics of UAMs will be analyzed in order to make explicit the tasks, constraints and bounds that should be integrated in both optimization methods.

3.1.1 State of the Art

Even though 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. [44] and [77] describe methods for 3D optimal trajectory generation and control, however their works are focused only on UAVs, thus considering few DoFs. In [102], 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 [39], where a nonlinear model predictive control scheme is proposed to achieve pick-and-place operations. Another example is [41] 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.

The first method that will be presented in this Chapter takes advantage of a quadratic programming technique, and it draws inspiration from other robotic fields (*e.g.*, [32, 60]). Specifically we use the online active set strategy ([33, 86]) which analyzes 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. A similar approach is [110], 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 [110], 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.

The second method that will be proposed is a trajectory generation based on a Non Linear Model Predictive Control (NMPC) framework. The work of [41] and [39], mentioned above, are the main examples of optimal control schemes applied to UAMs. However, both methods are used for offline trajectory generation and online performance is left for future work. Moreover, none of these methods aims to accomplish several tasks simultaneously by exploiting the redundancy of UAMs. The redundancy of UAM systems (*i.e.* 4 DoFs from the quadrotor and, at least, 3 DoFs from the arm system), can be exploited not only to achieve a primary task (*e.g.*, trajectory tracking) but also a lower priority task by optimizing some given quality indexes (*e.g.*, improving manipulability or avoiding joint limits) as in [10,23]. In these works, as in most of recent approaches (*e.g.*, [11,66,95]) the problem is solved using hierarchical task composition control. In the proposed approach, a non linear model predictive control algorithm is employed to accomplish several tasks with a redundant aerial manipulator, while satisfying prescribed constraints.

To the knowledge of the author these are the first trajectory generation algorithms applied to such robots with all computations done in real time, and on board a computational unit with limited capabilities.

3.1.2 Problem Formulation

Variables

The goal of both approaches is to generate feasible trajectories for the state 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 will not include the platform tilt in the trajectory generation algorithm (roll and pitch angles will be assumed negligible in our analysis). Then, the robot n DoFs, are $\boldsymbol{\mu} = \begin{bmatrix} {}^w\boldsymbol{p}_b^T & {}^w\psi_b & \boldsymbol{q}^T \end{bmatrix}^T$, $\boldsymbol{\mu} \in \mathbb{R}^n$, where ${}^w\boldsymbol{p}_b \in \mathbb{R}^3$ and ${}^w\psi_b \in \mathbb{R}$ are the platform position and its yaw orientation in an inertial frame (w), and $\boldsymbol{q} \in \mathbb{R}^r$ are the angles of the r arm joints. In the following, we will assume that the inner control loop of the system can perfectly track the computed references.

Tasks

We can consider several objective functions for UAMs, in order to accomplish a given task or to preserve stability. A list of tasks that will be considered in the following is here proposed . The actual mathematical formulation of tasks, constraints and bounds will be presented separately for the two algorithms.

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. The trajectory can either have dimension 3, when just the translational motion is relevant ot the application, or 6, when the orientation has a predefined track to follow.

Robot center of mass alignment When the arm center of mass (CoM) is not aligned with the geometrical center 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.

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. This task can be fulfilled by limiting the motion of arm center of mass.

Limiting quadrotor accelerations When rapid end effector motions are required, it is better to distribute the motion in the arm joints rather than in the platform.

Manipulability During a manipulation task, the arm should perform a motion without getting close to kinematic singularities.

Desired joint position We can choose a set of desired arm joint positions to achieve some favorable conditions such as distance from joint limits, a high manipulability or a limited center of mass displacement. This configuration can be set as a reference during the navigation phase, or just used as attractor to guide the optimization solution when the main tasks present a nonempty null space.

Velocity minimization If different weights to joint velocities are applied, the motion can be arbitrarily distributed on the UAM joints.

Bounds and Constraints

Aerial manipulators are characterized by a complex structure which impose hard constraints on the set of joint configurations that can be taken by the system. In the following, the constraints and bounds considered in the approach are described.

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.

Obstacle avoidance Another obvious constraint of the system is to avoid external objects during the trajectory. In the following, obstacles with a sphere or a cylindrical shape will be considered.

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 degree of freedom has an obvious lower bound, while arm joint bounds are given by the physical arm structure.

3.2 Constraint Based Quadratic Optimization

The first approach presented is based on the solution of a quadratic programming problem. The main idea of the algorithm is to express tasks and constraints in the acceleration space, obtaining a local second order approximation of the system. Then, the optimization problem will be expressed with respect to the second order derivative of state $\boldsymbol{\mu}$. The approach will be presented and validated both with simulations and experiments.

3.2.1 Algorithm

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

$$\min f_{t,i}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, t) \quad \text{and} \quad f_{c,j}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, t) \leq 0\,, \tag{3.1}$$

where $\min f_{t,i}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, t)$ represents the ith generic task and $f_{c,j}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, t) \leq 0$ 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, $f_{t,1}(\boldsymbol{\mu}, t) = ||\boldsymbol{p}_e^d(t) - \boldsymbol{p}_e(t)||$, where $\boldsymbol{p}_e^d(t)$ and $\boldsymbol{p}_e(t)$ are the desired and actual end effector positions, respectively.

The key idea of this approach is to assign desired dynamics to $f_{t,i}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, t)$ and $f_{c,j}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}, t)$. In fact, by using the Gronwall inequality as in [15], a constraint expressed as $f \leq 0$ is satisfied if

$$\dot{f} \leq -\lambda_1 f, \tag{3.2}$$

where λ_1 is a positive scalar gain. Notice that this is just a sufficient condition, since the inequality of (3.2) is more restrictive than the original one.

By applying iteratively the Gronwall inequality on $\dot{f}+\lambda_1 f \leq 0$ from (3.2), we have

$$\ddot{f} \leq -\lambda_2 \left(\dot{f} + \lambda_1 f\right) - \lambda_1 \dot{f}, \tag{3.3}$$

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

$$\min ||\ddot{f} + (\lambda_2 + \lambda_1)\dot{f} + \lambda_2\lambda_1 f||^2. \tag{3.4}$$

Notice that if the cost function in (3.4) 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 $f_{c,j}(\boldsymbol{\mu}) \leq 0$ depending only on robot configuration, and applying this approach, we end up with

$$A_j \ddot{\boldsymbol{\mu}} \leq u_{Aj}, \tag{3.5}$$

where

$$A_j = \frac{\partial f_{c,j}}{\partial \boldsymbol{\mu}}, \tag{3.6a}$$

$$u_{Aj} = -\lambda_2\lambda_1 f_{c,j} - \left(\dot{\boldsymbol{\mu}}^T \frac{\partial^2 f_{c,j}}{\partial \boldsymbol{\mu}^2} + (\lambda_1 + \lambda_2)\frac{\partial f_{c,j}}{\partial \boldsymbol{\mu}}\right)\dot{\boldsymbol{\mu}}\,. \tag{3.6b}$$

On the other hand, when a constraint also depends on joint velocities, as $f_{c,j}(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}) \leq 0$, the Gronwall inequality should be applyied only once. While the constraint expression is the same as in (3.5), in this case the terms A_j and u_{Aj} are computed as

$$A_j = \frac{\partial f_{c,j}}{\partial \dot{\boldsymbol{\mu}}}, \quad u_{Aj} = -\lambda_2 f_{c,j} - \frac{\partial f_{c,j}}{\partial \boldsymbol{\mu}}\dot{\boldsymbol{\mu}}\,. \tag{3.7}$$

Similarly to the linear formulation obtained for the constraints, see (3.5), the cost functions can be defined as

$$\min ||F_i \ddot{\boldsymbol{\mu}} - b_i||^2, \tag{3.8}$$

where the computations of F_i and b_i are straightforward.

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

Quadratic problem formulation

The formulation of the optimization problem is quite straightforward. The cost function in (3.8) results in the quadratic form

$$\min_{\boldsymbol{z}} \|F_i \boldsymbol{z} - b_i\|^2 = \min_{\boldsymbol{z}} \left(\boldsymbol{z}^T F_i^T F_i \boldsymbol{z} - 2 b_i^T F_i \boldsymbol{z} \right) , \tag{3.9}$$

where the regressor variable $\ddot{\boldsymbol{\mu}}$ has been replaced by $\boldsymbol{z}$ 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.*, [32]). Alternatively, as in our case, if no hierarchy applies, a weighted sum of the N_T objective functions can be considered with

$$\begin{aligned} \min_{\boldsymbol{z}} \sum_{i=1}^{N_T} G_i &= \min_{\boldsymbol{z}} \sum_{i=1}^{N_T} \frac{w_i}{h_i} \left(\boldsymbol{z}^T F_i^T F_i \boldsymbol{z} - 2 b_i^T F_i \boldsymbol{z} \right) \\ &= \min_{\boldsymbol{z}} \sum_{i=1}^{N_T} \frac{w_i}{h_i} \left(\boldsymbol{z}^T H_i \boldsymbol{z} + \boldsymbol{m}_i^T \boldsymbol{z} \right) , \end{aligned} \tag{3.10}$$

where w_i and h_i are weights and normalization factors, respectively. When a weighted sum is employed, it is 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, defined the square root of the largest eigenvalue of the product matrix $H_i^T H_i$. 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 (3.9).

Remark I In order to effectively use the normalization factor h_i, it is beneficial to split the tasks that are not dimensionally consistent. 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

$$\begin{aligned} \min_{\boldsymbol{z}} G = \min_{\boldsymbol{z}} &\left(\frac{1}{2}\boldsymbol{z}^T H \boldsymbol{z} + \boldsymbol{m}^T \boldsymbol{z}\right) \\ s.t. \quad & \boldsymbol{l}_b \leq \boldsymbol{z} \leq \boldsymbol{u}_b \\ & \boldsymbol{l}_A \leq A\boldsymbol{z} \leq \boldsymbol{u}_A \,, \end{aligned} \tag{3.11}$$

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 $A = \begin{bmatrix} A_1^T & A_2^T & \dots & A_{N_C}^T \end{bmatrix}^T$ 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.

We obtain a solution $\boldsymbol{z}\left(k\right)$ of the system in (3.11) for every time step k, thus the position and velocity references can be updated, for example, with an Euler integration as

$$\boldsymbol{\mu}\left(k\right) = \boldsymbol{\mu}\left(k-1\right) + \dot{\boldsymbol{\mu}}\left(k-1\right)\delta t + \boldsymbol{z}\left(k\right)\frac{\delta t^2}{2} \tag{3.12a}$$

$$\dot{\boldsymbol{\mu}}\left(k\right) = \dot{\boldsymbol{\mu}}\left(k-1\right) + \boldsymbol{z}\left(k\right)\,\delta t. \tag{3.12b}$$

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 with computational units with limited capabilities. 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 functions weights, depending on the mission phase or external triggers. Notice that the trajectories generated applying the method in (3.11) and (3.12a) are continuous in the velocity, since it results from the integration of a bounded acceleration profile.

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 loop control is not ideal and dynamics between desired ($\boldsymbol{\mu}^d$) and actual ($\boldsymbol{\mu}$) 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 dynamic analysis will not be performed here.

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 [111], to avoid that the system takes a configuration which originate an unfeasible problem. Moreover, the closed loop control can affect the stability of the system. The nonlinearity of the optimization method makes it really difficult to find stability bounds and conditions.

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 using instead the currently computed value of $\boldsymbol{\mu}$, and parameters λ_1 and λ_2 are chosen according to the inner control bandwidth and time step δt.

3.2.2 UAM Tasks

In this Section we consider a number of possible cost functions along with the corresponding methods to compute the H and $\boldsymbol{m}$ matrices. The weighting factors will not be mentioned because they depend on the particular navigation phase and mission objectives, as explained in Sec. 3.2.4.

End effector tracking

In order to track a desired end effector trajectory $\boldsymbol{p}_e^d \in \mathbb{R}^6$, the following cost function can be defined

$$G_1 = \|\boldsymbol{e}_e\|^2 , \tag{3.13}$$

where $\boldsymbol{e}_e = \boldsymbol{p}_e(t) - \boldsymbol{p}_e^d(t)$ is the tracking error, which can be written in acceleration form, following the procedure described in Sec. 3.2.1, as

$$F_1 = 2\boldsymbol{e}_e^T J_e , \tag{3.14}$$

with $J_e \in \mathbb{R}^{6\times n}$ the end effector Jacobian matrix. The expression of b_1 to be used in (3.8) 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. 3.2.1, normalization factors for the end effector task should be computed separately for the translational and rotational parts of the Jacobian.

Robot center of mass alignment

As mentioned before, it is important to favor the alignment of the arm CoM with quadrotor geometrical center. Thus, we can easily compute the cost function G_2 and the related Jacobian F_2 as

$$G_2 = {}^b\boldsymbol{p}_c^T\,{}^b\boldsymbol{p}_c\,, \qquad \text{and} \qquad F_2 = 2\,{}^b\boldsymbol{p}_c^T\,{}^bJ_c\,, \tag{3.15}$$

where ${}^b\boldsymbol{p}_c \in \mathbb{R}^2$ is the displacement of the center of mass 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 [66, 95].

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

Forces on quadrotor horizontal plane

In order to limit the forces exerted by the robotic arm on the quadrotor horizontal plane we can consider the following objective function:

$$G_3 = \frac{1}{2}\ddot{\boldsymbol{\mu}}^T\,{}^bJ_c^T\,{}^bJ_c\,\ddot{\boldsymbol{\mu}} + \frac{1}{2}\dot{\boldsymbol{\mu}}^T\,{}^b\dot{J}_c^T\,{}^b\dot{J}_c\,\dot{\boldsymbol{\mu}}^T + \ddot{\boldsymbol{\mu}}^T\,{}^bJ_c^T\,{}^b\dot{J}_c\,\dot{\boldsymbol{\mu}}. \tag{3.16}$$

In this case, we can compute directly the terms H_3 and $\boldsymbol{m}_3$ from expression in (3.16). This cost function penalizes the inertial forces exerted by the arm, by considering the acceleration of its center of mass.

Limiting quadrotor accelerations

Penalizing quadrotor accelerations allows to perform rapid end effector motions with arm joints instead of quadrotor platform. This goal is achieved with

$$G_4 = {}^w\ddot{\boldsymbol{p}}_b^T\,{}^w\ddot{\boldsymbol{p}}_b\,. \tag{3.17}$$

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.

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

$$G_5 = \frac{1}{\det\left(J_{e,q} J_{e,q}^T\right)}\,, \tag{3.18}$$

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 if all the 6 DoFs of the end effector are considered, an arm with 6 joints is required. Moreover, it is almost impossible to analytically compute the required matrices to apply this objective function. 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 [112].

Desired joint position

The goal of driving the arm to desired arm joint positions $\boldsymbol{q}^d \in \mathbb{R}^r$ when the main tasks present a nonempty null space, 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 just preserve the joint bounds if infinite values are assigned to the p factor and the weight w_6.

Velocity minimization

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

$$G_{7,i} = \dot{\mu}_i^{\,2}\,. \tag{3.19}$$

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.

3.2.3 UAM Bounds and Constraints

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

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 $\boldsymbol{p}_{l1}$ and $\boldsymbol{p}_{l2}$, the distance d_l between the leg and a point of the arm $\boldsymbol{p}$ can be computed with

$$d_l = ||D_l\left(\boldsymbol{p}_{l1} - \boldsymbol{p}\right)||\,, \tag{3.20}$$

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

$$d_l^2 \geq d_{\text{min}}^2\,, \tag{3.21}$$

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

Assuming $\boldsymbol{p}$ the arm end effector position with respect to the quadrotor base, we have that $\boldsymbol{p} = {}^bJ_{e,q}\dot{\boldsymbol{q}}$, where ${}^bJ_{e,q}$ 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

$$2(\boldsymbol{p}_{l1} - \boldsymbol{p})^T D_l\, {}^bJ_{e,q}\,\dot{\boldsymbol{q}} \leq -\lambda_1\left(d_l^2 - d_{min}^2\right) \tag{3.22}$$

from which $A_1 = 2(\boldsymbol{p}_{l1} - \boldsymbol{p})^T D_l\, {}^bJ_{e,q}$, 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.

Obstacle avoidance

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

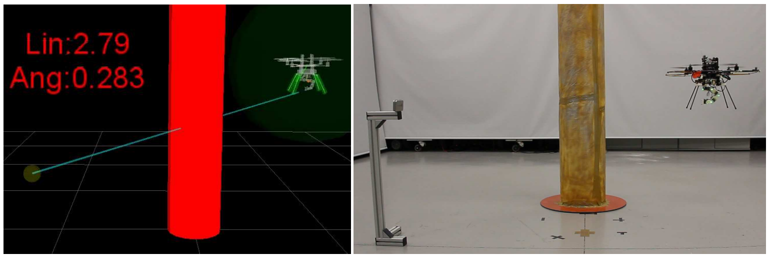


Figure 3.1: *Experiment setup with the flying arena and a cylinder pilar used as obstacle (the left frame corresponds to the telemetry visualization).*

quadrotor positions) $\boldsymbol{p} \in \mathbb{R}^3$, we can define the inequality

$$(\boldsymbol{p}_o - \boldsymbol{p})^T (\boldsymbol{p}_o - \boldsymbol{p}) \geq d_{min}^2 \, . \tag{3.23}$$

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

Position, velocity and acceleration bounds

The position, velocity and acceleration limits of the UAM joints are considered. The bounds have to be reported in acceleration form with the strategy presented in Sec. 3.2.1. Given an upper bound $\mu_{M,i}$ on the ith joint position, the Gronwall inequality can be applied twice, obtaining the following position bound in acceleration form

$$\ddot{\mu}_i \leq -\left(\lambda_1 + \lambda_2\right) \dot{\mu}_i - \lambda_1 \lambda_2 \left(\mu_i - \mu_{M,i}\right) \, . \tag{3.24}$$

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

$$\ddot{\mu}_i \leq -\lambda_2 \left(\dot{\mu}_i - \dot{\mu}_{M,i}\right) . \tag{3.25}$$

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.

3.2.4 Validation

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

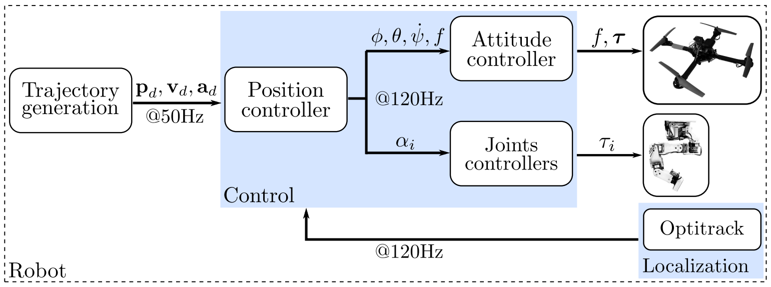


Figure 3.2: *Overview of the architecture pipeline for trajectory generation and UAM control with all algorithms running on board.*

other methods because, to the author 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 Robot Operating System (ROS) have been performed, their results are here avoided for the sake of conciseness.

The quadrotor used in the experiments is the ASCTEC Pelican research platform shown in Fig. 3.1 and its control architecture is shown in Fig. 3.2. 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 200 g including batteries . The complete UAM shows an accuracy of about 0.2 rad and 0.05 m. Each joint has its own proportional-integral-derivative controller. The Denavit-Hartenberg parameters of the arm are reported in Tab. 3.1. The arm base is 10mm below the body frame along the vertical axis. All algorithms are running on board in real-time at 50 Hz using an Intel Atom CPU (@1.6GHz) with Ubuntu 14.04LTS and ROS Indigo. The optimized high-rate C++ implementation takes advantage of QPoases as quadratic programming solver [34]. All experiments validating the present method have been performed at Institut de Robòtica i Informàtica Industrial (IRI), CSIC-UPC, equipped with an Optitrack motion capture system running at 120 Hz and used in the low-level quadrotor position control. In all the experiments the quadrotor is autonomously taken off and landed, and both maneuvers are considered out of the scope of the work. Fig. 3.1 shows the experiment setup with the flying arena and the cylindrical pilar used as obstacle.

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 3.1: *Denavit-Hartenberg parameters of our 6 DoFs arm.*

In the real platform, a subset of cost functions and constraints of Sections 3.2.2 and 3.2.3 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 CoM, 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 2 m in both x and y directions, and 0.2 m 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 0.05 m and 0.2 rad of linear and angular Euclidean errors, respectively.

In order to show the effects of the different tasks and constraints, the mission is split 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 in order to maximize stability and minimize disturbances, *e.g.*, torques produced by displacement of the arm CoM. 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

	Tasks				
	EE	CoM	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 3.2: *Tasks weightings depending on mission phases.*

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. 3.2. In the subsequent figures, the transition between navigation and close interaction phases is shown with a black vertical dashed line.

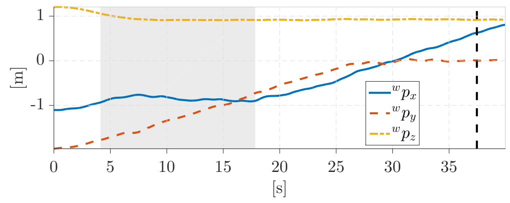
As presented in Sec. 3.2.1, the dynamics of the joints, tasks and the approach to constraints is governed by parameters λ_1 and λ_2. For the described system, they have been chosen equal to $0.8\frac{1}{s}$ and $4\frac{1}{s}$, respectively, in accordance with lower-level control loop bandwidth.

In Fig. 3.3 the behavior of all UAM joints during the complete experiment with the obstacle is reported. Fig. 3.4 shows the task errors during the mission, whereas in Fig 3.5 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.

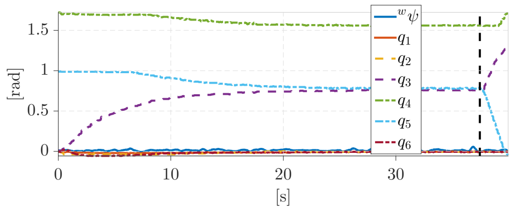
Navigation phase

During the navigation, large weights are assigned to the cost functions of CoM alignment and desired joint positions. Thus, long displacements are performed by the quadrotor and the arm is driven to a desired position while minimizing CoM misalignment. These profiles are reported in Fig. 3.3. The behavior of the end effector task error is reported in Fig. 3.4(a) and Fig. 3.4(b), for the translational and rotational parts, respectively. In Fig. 3.4(c) and 3.4(d), the profiles of the CoM alignment and the arm joints positioning can be analyzed. Notice that the CoM 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. 3.6, where frame (a) is just after



(a) 3D position of the quadrotor.



(b) Quadrotor yaw angle and arm joint values.

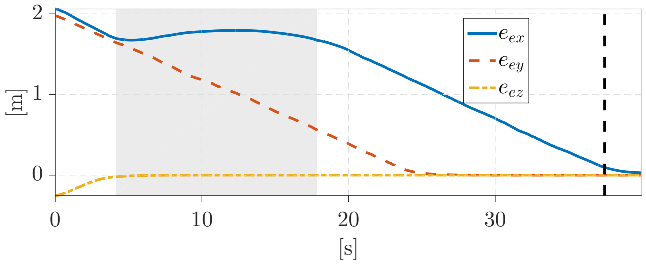
Figure 3.3: *Values of the robot degrees-of-freedom during a real experiment with an obstacle inline between the initial and desired end effector positions. The gray region in figure (a) corresponds to the activation of the obstacle avoidance constraint. Notice how the x axis (continuous blue line) is blocked. The vertical dashed line indicates the point where the weights of the arm joints positioning and arm CoM alignment tasks are reduced, and quadrotor motion is penalized.*

the take off, and frame (b) represents the robot configuration during navigation where the CoM alignment and arm joints positioning tasks prevail. Frame (c) represents the arm configuration during the interaction phase, that will be described in the following.

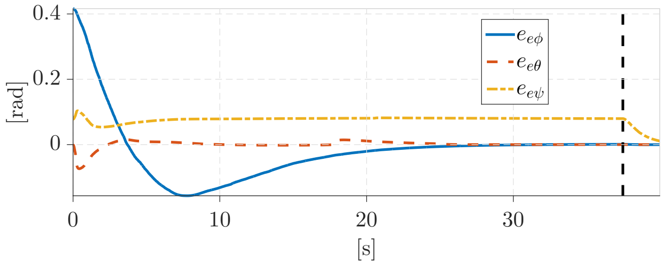
To show the need for the collision avoidance constraint, an obstacle in the middle of the trajectory (*i.e.* shortest path) has been added. The quadratic programming solver generates a feasible trajectory, and the UAM avoids the obstacle with the minimum assigned distance of 0.6 m (inflation radius around the platform). To clearly see how the obstacle avoidance works, a comparison between two trajectories with the same parameters is shown in Fig. 3.5. 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. 3.3(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.

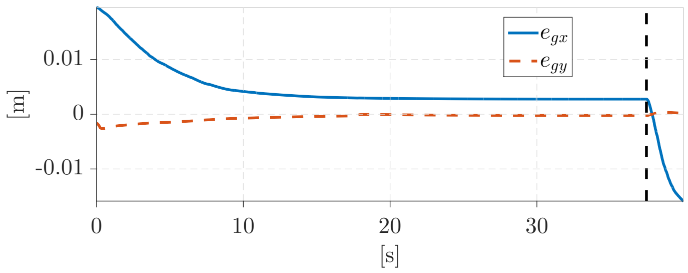
As it can be seen in Fig. 3.3, the quadrotor motion is clearly performed



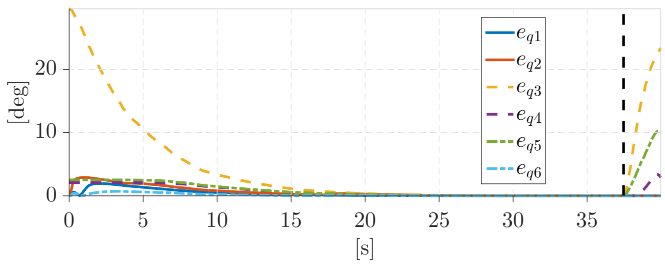
(a) Linear values of the end effector task error $\boldsymbol{e}_e$ from Eq.(3.13).



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



(c) Arm center of mass alignment error.



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

Figure 3.4: *Task errors corresponding to the real experiment of Fig. 3.3 where there exists 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 CoM alignment tasks are reduced in favor of the end effector positioning task.*

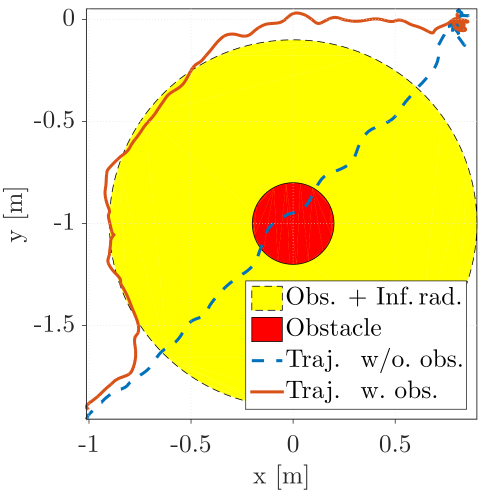


Figure 3.5: *Comparison between two real trajectories with the same initial and final positions, but without any obstacle (dashed blue line) and with an obstacle lying between the initial and desired end effector positions (continuous brown line). The small red circle corresponds to the actual obstacle and the yellow area (yellow circle with a dashed edge) includes the inflation radius applied to the obstacle.*

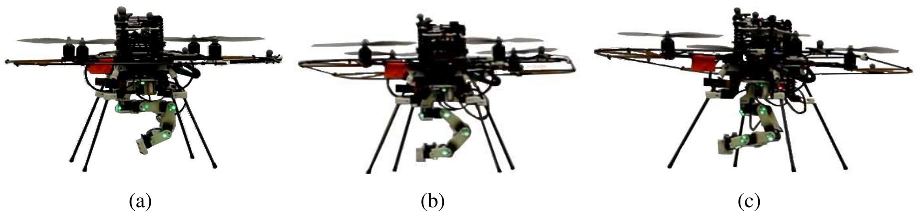


Figure 3.6: *Samples of arm configurations depending on tasks weightings during a real experiment: (a) After taking off, (b) during navigation and (c) in the close interaction zone.*

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 6 s and 7 s, while the theoretical settling time $\frac{5}{\lambda_1}$ is equal to 6.25 s.

Interaction phase

The second phase starts when the end effector is 15 cm far from the desired position. At this point, weights are changed to the values of Tab. 3.2. Notice how the arm joints move towards the goal, see Fig. 3.6(c) that represents the robot configuration during the interaction phase, and Fig. 3.3, which reports the behavior of the joint variables. As a consequence, CoM and joint positioning task errors are increasing, as reported in Fig. 3.4(c) and Fig. 3.4(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 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.

3.3 Nonlinear Model Predictive Control

3.3.1 Algorithm

Model Predictive Control (MPC) is a control technique that consists in numerically solving an optimization problem at each time step. A model of the process is used to predict the future evolution of the system in order to optimize the control signal. This future time horizon, in which the algorithm is computed, is called prediction horizon. Non Linear Model Predictive Control refers to particular MPC problems where the process model is nonlinear, the cost functional is non-quadratic or general nonlinear constraints are used.

Considering a generic dynamic system, which has a state $\boldsymbol{x}$ and is controlled by the variables in $\boldsymbol{u}$, the solution of the optimal control problem at time $t_k = kT_s$, where T_s is the time step, over a finite prediction horizon of N steps, is defined by

$$\begin{aligned} \min_{\boldsymbol{u}} \quad & h(\boldsymbol{x}_k, \boldsymbol{u}, t_k) \\ s.t. \quad & \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ & \boldsymbol{y}_{min} \leq g(\boldsymbol{x}_k, \boldsymbol{u}_k) \leq \boldsymbol{y}_{max} \\ & \boldsymbol{x}_{min} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{max} \\ & \boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max} \end{aligned} \tag{3.26}$$

where $\boldsymbol{u} = \left[\boldsymbol{u}_k^T, \boldsymbol{u}_{k+1}^T, \ldots, \boldsymbol{u}_{k+N-1}^T\right]^T$ is a vector of control variables, f is the model of the system, g represents a generic constraint, and h is a generic cost function defined by

$$h(\boldsymbol{x}_k, \boldsymbol{u}, t_k) = \sum_{i=0}^{N-1} \left(\sum_{j=0}^{N_h-1} h_j + ||\boldsymbol{u}_{k+i} - \boldsymbol{u}_{k+i-1}||^2_{W_u} \right) , \tag{3.27}$$

where $h_j = h_j(\boldsymbol{x}_{k+i}, \boldsymbol{u}_{k+i}, t_k)$ are N_h generic positive cost functions to be minimized, and W_u weights on control actions. Notice how in (3.26) the optimization process is constrained by the dynamic model of the system, $\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k)$, subject to lower and upper bounds on state, control actions, and output variables defined by $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$, $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$ and $\boldsymbol{y}_{min}$ and $\boldsymbol{y}_{max}$, respectively.

System model in the optimization problem

For the optimization problem we can consider the state $\boldsymbol{x} = \boldsymbol{\mu} = \begin{bmatrix} {}^w\boldsymbol{p}_b^T & {}^w\psi_b & \boldsymbol{q}^T \end{bmatrix}^T$ while the control action is directly the state derivatives $\boldsymbol{u} = \dot{\boldsymbol{\mu}} = \begin{bmatrix} {}^w\dot{\boldsymbol{p}}_b^T & {}^w\dot{\psi}_b & \dot{\boldsymbol{q}}^T \end{bmatrix}^T$. Therefore the function of the system model f is just an integrator of input $\boldsymbol{u}$. The non-linearity of the approach appears in the definition of the cost function and in the constraints. The choice of this reduced system model was made to guarantee online computation on board the platform with limited resources.

Hereafter, we assume that the inner control loop of the system can perfectly track the computed references.

3.3.2 UAM tasks

We can consider several cost functions for UAMs (3.26), in order to accomplish the tasks mentioned in Sec. 3.1.2. In this Section we present a few of such task functions. Note the differences in the formulation of the cost functions with respect to the previous approach. In particular, the cost functions will consider and weigh the evolution of the objective variables along a specified horizon. Moreover, a terminal cost, weighing the last value of the objective function along the evolution horizon, will be added.

End effector tracking

In order to track a desired end effector trajectory, the following cost function can be defined

$$h_1 = \sum_{i=0}^{N-1} ||\boldsymbol{e}_e\left(\boldsymbol{x}_{k+i}\right)||^2_{W_1} + ||\boldsymbol{e}_e\left(\boldsymbol{x}_{k+N}\right)||^2_{W_{s1}}, \tag{3.28}$$

where $\boldsymbol{e}_e\left(\boldsymbol{x}_{k+i}\right) = {}^w\boldsymbol{p}_e(\boldsymbol{x}_{k+i}) - {}^w\boldsymbol{p}_e^d\left(t_{k+i}\right)$ is the end effector tracking error, ${}^w\boldsymbol{p}_e^d$ is the desired end effector position over the predicted horizon, and W_1 and W_{s1} are the weight matrices for the task and the terminal cost. Although h_1 encloses the end effector position error, notice how this cost function can also consider its orientation by augmenting with ${}^w\boldsymbol{\phi}_e$ and ${}^w\boldsymbol{\phi}_e^d$ the current and desired end effector orientation, respectively.

Notice that each cost function that will be presented in the following includes a terminal cost. In a number of works, collected in [73], it is pointed out that terminal cost weights are a key ingredient to achieve stability with NMPCs. However, stability proof of the proposed NMPC is outside the scope of the present work.

Robot center of mass alignment

Similar to the first method, a task is designed to favor the alignment of the arm CoM with the quadrotor center.

$$h_2 = \sum_{i=0}^{N-1} ||{}^b\boldsymbol{p}_c(\boldsymbol{x}_{k+i})||^2_{W_2} + ||{}^b\boldsymbol{p}_c\left(\boldsymbol{x}_{k+N}\right)||^2_{W_{s2}} \tag{3.29}$$

where ${}^b\boldsymbol{p}_c(\boldsymbol{x}_{k+i})$, is the vector describing the position of the arm CoM projected onto the xy plane of the body reference frame during the prediction horizon. This expression can be obtained as in [66, 95]. Notice that we are assuming that the quadrotor is internally balanced. Otherwise, a different equilibrium point should be assigned for the arm CoM.

The formulation of (3.29) can effectively reduce the static torques produced on the quadrotor. However, an additional cost function can be considered in order to reduce the disturbances produced by inertial forces on the quadrotor. In fact, the following cost function accounts for the velocity of the CoM:

$$h_3 = \sum_{i=0}^{N-1} ||{}^b\boldsymbol{v}_c\,(k+i)\,||^2_{W_3} + ||{}^b\boldsymbol{v}_c\,(k+N)\,||^2_{W_{s3}} \tag{3.30}$$

where ${}^b\boldsymbol{v}_c\,(k+i) = {}^bJ_c(\boldsymbol{x}_{k+i})\,\boldsymbol{u}_{k+i}$ and ${}^bJ_c(\boldsymbol{x}_{k+i})$ the arm CoM Jacobian. Penalizing the motion of the CoM has the goal of reducing the inertial forces resulting on the quadrotor plane, thus on axes orthogonal to the thrust direction.

Arm manipulability

A first direct way of expressing the manipulability cost function is

$$h_4 = \sum_{i=0}^{N-1} W_4 \frac{1}{||D\,(\boldsymbol{x}_{k+i})\,||^2} + W_{s4} \frac{1}{||D\,(\boldsymbol{x}_{k+N})\,||^2}\,, \tag{3.31}$$

where $D\,(\boldsymbol{x}_{k+i})$=det $\Big(J_{e,q}\,(\boldsymbol{x}_{k+i})\,J_{e,q}\,(\boldsymbol{x}_{k+i})^T\Big)$ and $J_{e,q}\,(\boldsymbol{x}_{k+i})$ is the submatrix of the end effector Jacobian $J_e\,(\boldsymbol{x}_{k+i})$ composed by the columns corresponding to the arm joints.

A second cost function useful to maximize the manipulability of the arm is obtained by minimizing the condition number of matrix $J_{e,q}J_{e,q}^T$:

$$h_5 = \sum_{i=0}^{N-1} W_5||1-\rho_{J_{eq}}(k+i)||^2 + W_{s5}||1-\rho_{J_{eq}}(k+N)||^2\,, \tag{3.32}$$

where $\rho_{J_{eq}}(k+i) = \rho\Big(J_{e,q}\,(\boldsymbol{x}_{k+i})\,J_{e,q}\,(\boldsymbol{x}_{k+i})^T\Big)$ and $\rho(A) = \lambda_{A,M}/\lambda_{A,m}$ is the condition number of matrix A, with $\lambda_{A,M}$ and $\lambda_{A,m}$ the maximum and minimum eigenvalues, respectively. Notice that, far from singularity points, matrix $J_{e,q}J_{e,q}^T$ is symmetric positive definite, then its eigenvalues are real and positive. Notice that the closer the cost function h_5 in (3.32) is to zero, the more the manipulability ellipsoid is similar to a generalized sphere. Notice how, in this case, the weights W_4, W_5, W_{s4} and W_{s5} are scalar values.

The alternative formulation of [112] in this case would be expressed with a cost function on the control action $\boldsymbol{u}$.

Cost function on the control action

The cost function $h(\boldsymbol{x}_k, \boldsymbol{u}, t_k)$ includes the term $||\boldsymbol{u}_{k+i} - \boldsymbol{u}_{k+i-1}||^2_{W_u}$ penalizing the control action. Notice that the control action for our system consists in the derivatives of UAM joint positions. Then, when weight matrix W_u is properly set, motion can be arbitrarily distributed on the UAM joints. For instance, it can be desirable to penalize quadrotor motion, and perform manipulation tasks with the arm DoFs. Notice how the cost function on the control action is formulated in differential form. It is a well-known method to obtain zero steady state error.

3.3.3 UAM Bounds and Constraints

We introduce a first constraint consisting in avoiding the specific self-collision with the quadrotor base. To do so, we impose a safety distance between the joints and the aerial base:

$$\begin{aligned} {}^{b}z_i\left(\boldsymbol{x}_k\right) \geq 0.1\,[m] \\ {}^{b}z_e\left(\boldsymbol{x}_k\right) \geq 0.1\,[m] \end{aligned} \tag{3.33}$$

where ${}^{b}z_e$ is the position of the end effector in the z direction of the body reference frame and, similarly, ${}^{b}z_i$ is the distance of the i-th joint. Notice that just a limited number of points of the arm can potentially have a collision with the aerial base, reducing the number of required constraints. These constraints will be integrated in the optimization through the term g. Thanks to the flexibility of the method, a generic constraint can be added to the problem.

Moreover, we imposed bounds to both robot joints and velocities. Robot joints bounds are expressed as state bounds, by imposing specific values to $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$. On the other hand, velocity bounds correspond to control action constraints, thus they can be obtained by setting values of $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$.

3.3.4 Validation

Preliminary simulations using a reduced model with a weighting task strategy

In order to validate the described controller, we first present simulation case studies programmed in MATLAB. A simplified model is considered. In particular, the system consists of a planar quadrotor, thus described by three DoFs, roll angle, y displacement and z altitude, and a 2 DoFs robotic arm. As described in Sec. 3.1.2, roll angle is not considered in the state of the NMPC model. Therefore, the model of the system has 4 DoFs. The aim of this preliminary test is to show how several cost functions can be integrated using different weights, and solved exploiting the redundancy. The main cost function is the end effector trajectory tracking h_1. The end effector desired trajectory is described by the two translational positions y_e and z_e. In addition, the CoM cost functions h_2 and h_3 are integrated in the model.

The low cardinality of the model state allows to include the manipulability cost functions h_4 and h_5 too, that is quite difficult for systems of higher order, due to the required computational burden. This is possible for optimization solvers that can obtain a solution to the problem even without the Jacobians of the cost functions, such that a numerical computation of required terms is sufficient. The desired end effector trajectory is in y direction and it is divided in two phases. In the first phase a small sinusoidal trajectory is required, while the second phase is characterized by a bigger sinusoid.

The chosen optimization solver for the non linear problem is the ACADO Toolkit [46]. The system has been tested with different weight ratios among the cost functions, in order to analyze the interactions between them. For lack of space we present here the results of the test in which all the five cost functions have weights different from zero.

In Fig. 3.7, the end effector trajectory is reported. The reference trajectory is correctly tracked in both directions. The z reference is not reported because it is always null. In Fig. 3.8, the control actions, output of the NMPC algorithm, are represented. Notice that bounds on maximum velocity are respected. It is interesting to show how the motion is distributed on different joints. In fact, analyzing the blue line, corresponding to the control action on y, it is evident that the small sinusoidal motion is mainly performed with robot joints, while the large sinusoid is performed with quadrotor degrees of freedom. The behavior of the condition number $\rho_{J_{e,q}}$, defined in Sec. 3.3.1, is reported in Fig. 3.9. The results of three experiments, performed with different weights, are reported. The blue line

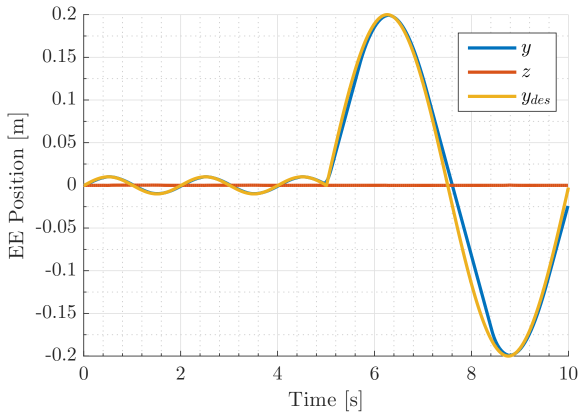


Figure 3.7: *End effector trajectory compared with the reference signal. Notice how the z reference is not reported because it is always null.*

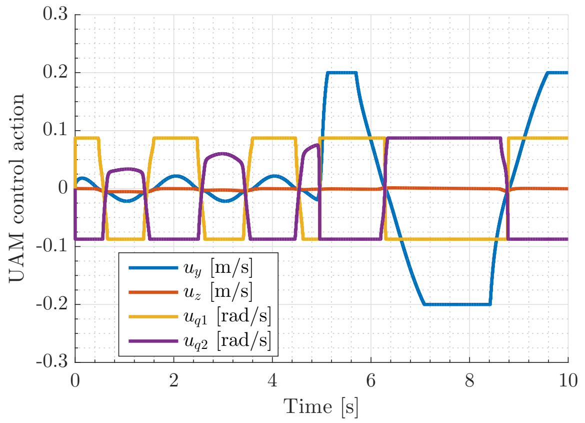


Figure 3.8: *Control actions on the four degrees of freedom computed by the NMPC. Blue and red lines represent quadrotor translational DoFs, while yellow and violet lines represent arm joint velocities.*

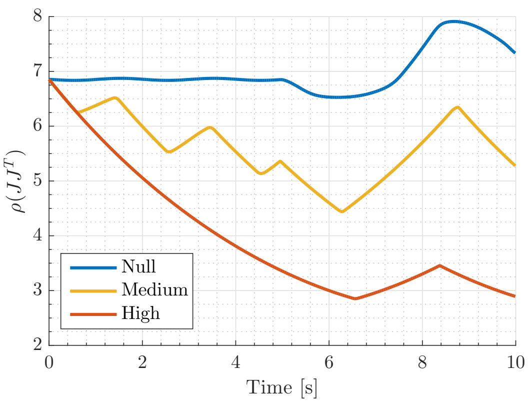


Figure 3.9: *Behavior of the condition number $\rho_{J_{e,q}}$ in three different experiments, differing in the weight assigned to the condition number cost function. The red line represents the experiment with higher assigned weight, achieving a smaller $\rho_{J_{e,q}}$, whereas the blue line is the result of the experiment with zero weight assigned to the cost function, thus not reducing $\rho_{J_{e,q}}$.*

represents an experiment when the cost function h_5 is not included in the optimization. The green line is a result of an experiment in which a large weight was assigned to this cost function, while the yellow line is the case of the experiment presented in the previous Figures. The approach can effectively reduce the condition number $\rho_{J_{e,q}}$ with a proper choice of the associated weight.

Simulations using a complete model and a hierarchical task strategy

In this Section a simulation of a fully actuated UAM consisting on a quadrotor plus a 3 DoFs serial arm (then a system with 7 DoFs overall) is used to show how the secondary tasks can also be executed with a different approach. We can impose a hierarchy between these tasks, similarly to [95] and [66], but in this case using optimal control. The required scheme includes nested optimizations as explained in the following.

Drawing inspiration from [92], we propose to compute a cascade of optimizations for each time step in order to minimize different cost functionals, each one related to a different task. However and differently from [92], we do not require to linearize the system. Initially, we solve the optimal control problem considering the cost functional of a main task (*e.g.*, the end effector trajectory tracking in (3.28)). From this minimization we are able to compute the predicted values of the cost functional for the N future predicted steps. Then, the hierarchy is imposed by adding a constraint in the secondary tasks resolutions. In particular, the value of the cost function at

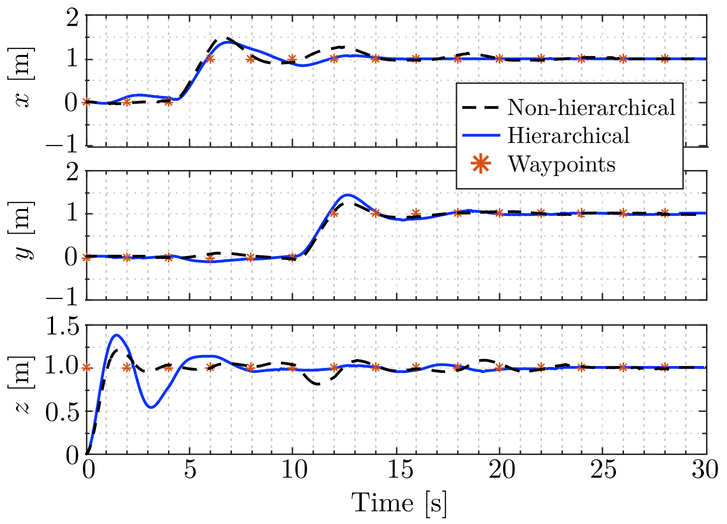


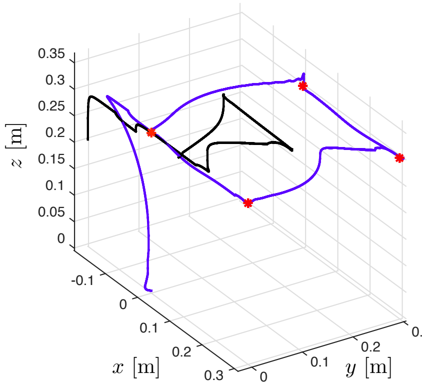
Figure 3.10: *3D end effector positioning. Comparison of performance between applying or not the hierarchical NMPC during a task consisting in following a trajectory with the arm end effector.*

higher priority is set as a constraint to the next optimization iteration.
For a secondary task, the minimization of the cost functional (*e.g.*, CoM alignment in (3.29)) follows the same procedure as for the primary task, but this time incorporating the new constraint on the cost function value. This constraint is defined by

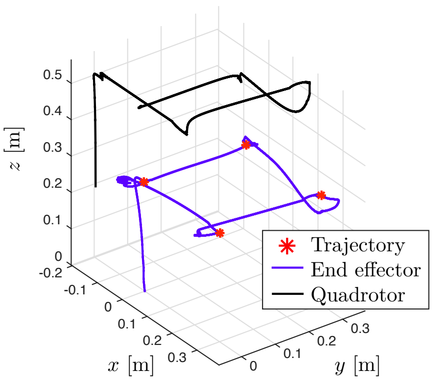
$$g_1(\boldsymbol{x}_k, \boldsymbol{u}, t_k) = 0, \quad \forall k. \tag{3.34}$$

This method can be repeated as many times as wanted for several subtasks imposing the hierarchy, depending on the redundancy of the robot (*i.e.* depending on the arm DoFs).

The effect of adding a secondary task hierarchically is shown in Fig. 3.10. In this simulation the controller is set with a prediction horizon of 8s. In both cases the optimal control drives the end effector through the required waypoints. Although the addition of the secondary task seems to add a small overshoot, this behavior can be avoided with a finer tuning. This comparison is then extended to Fig. 3.11 with a different trajectory, where it is clearly shown how the quadrotor trajectory (black line) performs quite better when a secondary task to align the arm CoM is present (Fig. 3.11(b)). In these particular simulations we removed all motion bounds to show how in the case with only the tracking task (Fig. 3.11(a)) the quadrotor and end effector are moving in similar vertical positions, thus with the arm fully extended in a forward position and close to singularities.



(a) Non-hierarchical NMPC



(b) Hierarchical NMPC

Figure 3.11: *3D trajectories done by the arm end effector using non-hierarchical and hierarchical NMPCs. In both cases the main task consists in following 4 desired way-points with the arm end effector. For the hierarchical approach a secondary task is added to vertically align the arm CoM and improve the platform flight behavior.*

Experiments

Here we present the results of the implementation of the NMPC strategy on the real UAM shown in Fig. 3.12 and described in Appendix A.

The platform is equipped with a Pixhawk board to allocate lower level controllers for the quadrotor and the arm (in this case PID controllers are employed), which are tuned to track the references provided by the NMPC. The optimization algorithm has been implemented using the ACADO toolkit [46], already used to generate complex 3D trajectories for quadrotors [19], and runs on board an ODROID XU3 at 2 Hz with a prediction horizon of 5 s (*i.e.* $N = 10$). Notice that, in order to compute the optimization solution in the required time, just the first three joints of the arm have been employed, obtaining a total of 7 DoFs. In the following experiments, we use the UAM kinematics as a prediction model in the optimization process because simulation case studies using the dynamic model revealed large computation times, far from real time performance, hence impractical for our application.

Fig. 3.13 shows a trajectory following experiment with the arm end effector during a 13 minutes flight. The control on both x and y directions work properly, however the vertical axis shows slower dynamic response, requiring finer thrust tuning.

In these experiments, we also imposed a main constraint consisting on avoiding self-collisions between the arm end effector and the platform base.

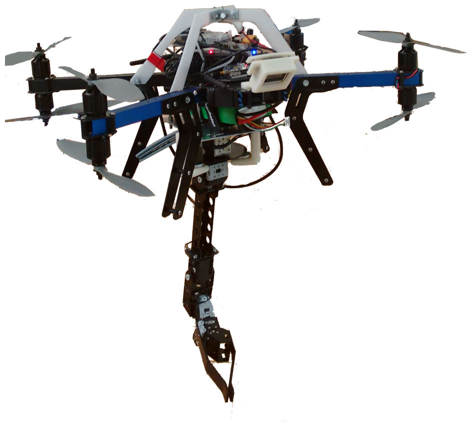


Figure 3.12: *Aerial manipulator, used in the experiments, consisting on a 3DR-X8 coaxial multirotor platform with a custom-built* 4 *degrees-of-freedom serial arm attached below.*

Fig 3.14 shows the positions of the joints, where it is clear how some joints reach their bounds but without overpassing them. Moreover, we added bounds in the solver for the quadrotor positioning. The resulting quadrotor movements are shown in Fig. 3.15.

3.4 Concluding Remarks

Two different methods capable of generating a trajectory for UAMs have been presented and validated through simulations and experiments. It is interesting to draw a comparison between the different characteristics of the two methods.

The main advantage of NMPC approach is that it considers in the optimization a longer horizon for the state evolution, which can lead to obtain a better long-term motion strategy. The Quadratic Programming Approach (QPA) instead, is based on a second order approximation of the system, which provides a local description of the motion evolution.
The counterpart to this characteristic is represented by the high difference in the required computational time. As the NMPC has to optimize on a larger problem, limits of the computational power restrict the practical implementation to a smaller set of states and lower loop rate. As reported in the validation Sections for both the methods, NMPC approach could be implemented with a rate of 2 Hz, while the QPA method could run at 50 Hz. This represents a relevant performance gap between the two approaches,

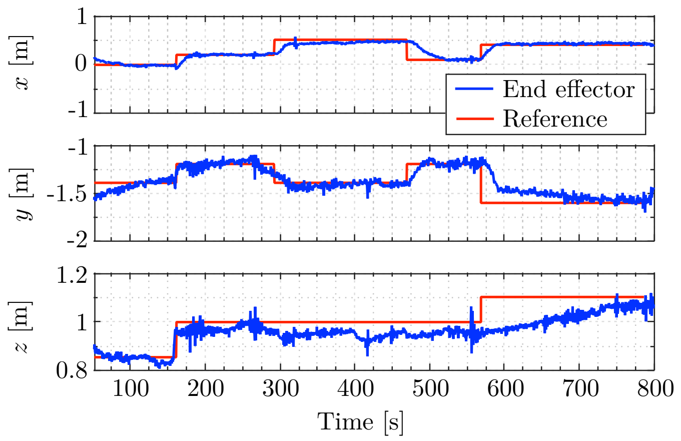


Figure 3.13: *Real arm end effector positions during a trajectory following task using the NMPC with the kinematics model.*

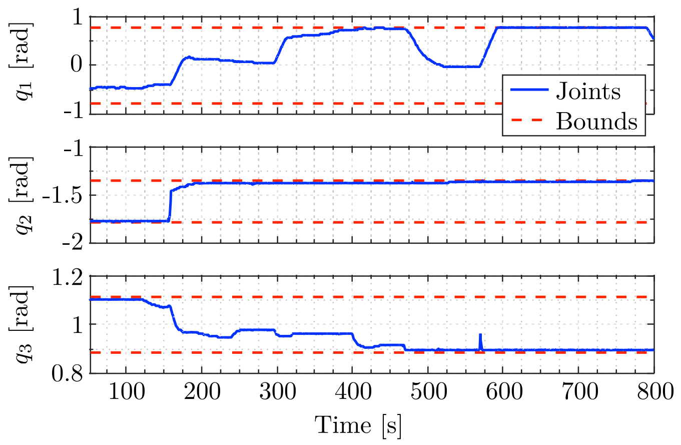


Figure 3.14: *Real arm joint positions (q_i) during a trajectory following task. The horizontal dashed lines correspond to lower and upper bounds of a constraint, set to avoid self-collisions.*

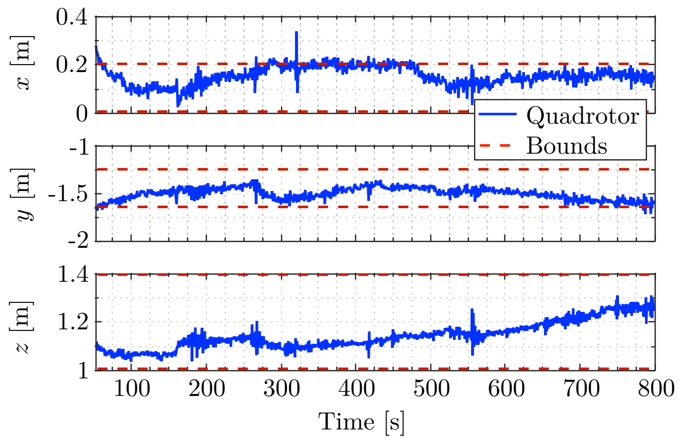


Figure 3.15: *Real quadrotor positions during a trajectory following task. The horizontal dashed lines correspond to lower and upper positioning bounds.*

and it entails direct impact on other performance metrics.
For instance, in the NMPC approach, with the available computational unit, motion velocity was chosen as control input of the system in order to reduce the cardinality of the states of the system. As a result, the generated joint trajectory presents discontinuities in the velocity profile and unbounded accelerations. On the contrary, the structure of the QPA approach enables to obtain a smooth velocity profile with bounded acceleration.
Moreover, the required computational time for NMPC makes impossible to apply in real applications a nested optimization method, which could be useful to solve hierarchical tasks. This is instead possible for QPA approach.

The structure of NMPC method and the solver employed for the practical implementation do not impose restrictive conditions on the cost function formulation. In fact, in general, the solver can minimize cost functions whose closed-form analytical formulations are unknown. This is an advantage with respect to QPA structure, where computation of cost function Jacobians is needed. On the other hand, non linear solvers can just provide a suboptimal solution, which can depend on the specific numeric method adopted in the algorithm and initial conditions provided. Therefore, it is really difficult to make assumptions on performance and stability with respect to this method.

As a conclusion, while NMPC approach is characterized by some advantages, as the longer optimization horizon, the trajectory generation method based on quadratic programming appears to be more robust and to perform better for the goals of aerial manipulators applications.

CHAPTER 4

State Estimation

4.1 Introduction

The estimation of the complete state of an aerial manipulator is a challenging task, which requires several different sensors and suitable fusion algorithms. In particular, the state to be estimated is composed of the following terms:

$$\begin{aligned} \boldsymbol{\xi} &= \begin{bmatrix} \boldsymbol{x}_q^T & \boldsymbol{\phi}^T & \boldsymbol{q}^T \end{bmatrix}^T \\ \dot{\boldsymbol{\xi}} &= \begin{bmatrix} \dot{\boldsymbol{x}}_q^T & \dot{\boldsymbol{\phi}}^T & \dot{\boldsymbol{q}}^T \end{bmatrix}^T \end{aligned} \tag{4.1}$$

where $\boldsymbol{x}_q \in \mathbb{R}^3$ is the quadrotor translational position $\boldsymbol{x}_q = \begin{bmatrix} x_q & y_q & z_q \end{bmatrix}^T$, $\boldsymbol{\phi} \in \mathbb{R}^3$ are the Euler angles $\boldsymbol{\phi} = \begin{bmatrix} \varphi & \theta & \psi \end{bmatrix}^T$ and $\boldsymbol{q} \in \mathbb{R}^{n_r}$ are the robot joint position.

Several different set of sensors can be employed in order to obtain a robust and complete information on the states. The initial design of the sensing system has to be performed by analyzing different possible sensors alternatives, and choosing a final set that provides the required information, while satisfying the constraints of the system and the requirements of the application.

Then, the information obtained by all the sensors has to be fused, in order to obtain a robust estimation of the states. The choice of a suitable estimation algorithms is a second relevant problem for the state estimation, and it depends on the sensor characteristics and on the adopted model of the system.

In the following, the design of a suitable sensing system will be discussed and the adopted estimation methods will be presented. A method to calibrate the camera, a sensor employed in the estimation, will be presented, and validating experiments will be shown.

4.2 Design of the Sensing System

4.2.1 Requirements and Constraints

The selection of the most suitable type of sensors and the algorithms used to obtain an estimation strongly depends on the requirements and constraints of the specific system and its application.

In the analysis of constraints and available solutions, we will make reference to the characteristics of small aerial manipulator systems. The analysis can be extended to aerial systems of bigger size, with typically less stringent constraint.

The main characteristics taken into account for the design can be summed up in the following list:

- Accuracy estimation at the end effector;
- Area of interest and choice between indoor and outdoor application;
- Frequency at which measures are available;
- Total weight of the sensing system;
- Limitation on system cost;
- Simplicity of the implementation.

The accuracy of the end effector position is obviously dependent on the application. When a manipulation activity has to be performed by a flying base, an accuracy of 1 cm at the end effector is a reasonable value for the performance, and estimation, target.

Another application dependent characteristic of the required estimation system is the area on which the activity has to be performed. In particular, sensors designed for indoor positioning in a pre-defined workspace are not always suitable for outdoor localization goals, where, in some application cases, the workspace can be potentially unlimited.

A number of constraints on the estimation system design are imposed by the characteristics of aerial manipulators, as the following ones.

Estimates of the state have to be updated according to the bandwidth of the control loop on which they have to be employed. Typical control loops for small UAVs are around $1\,\mathrm{kHz} - 200\,\mathrm{Hz}$ for attitude control and $100\,\mathrm{Hz} - 20\,\mathrm{Hz}$ and $10\,\mathrm{Hz} - 1\,\mathrm{Hz}$ for translational velocity and position control loops, respectively. This affects the required sample rate of sensors.

In order to save power consumption and to increase the flight time allowed by battery capacity, it is really important to decrease as much as possible the weight of any added component on board. The smallest UAVs that can be used for manipulation aims have a payload of about $0.5\,\mathrm{kg} - 2\,\mathrm{kg}$, battery included. Then, a maximum weight of $0.1\,\mathrm{kg} - 0.4\,\mathrm{kg}$ for the overall sensing system is a strict requirement.

Finally, cost is an obviously important decision variable with respect to the available sensing solutions.

4.2.2 Analysis of Alternatives and Solution Adopted

The analysis of the specific requirements for the UAM sensing system, performed in the previous Section, will be employed to evaluate the characteristics of the sensors that can be considered for the estimation of the state.

In Tab. 4.1 the state components have been associated to the sensors that can be employed for their estimation.

Joint positions and velocities of the robot arm can be directly measured. In fact, it is assumed that servos are equipped with suitable encoders. On the other hand, the estimation of the flying base states raises more issues.

Common UAVs are provided with Global Positioning System (GPS) module and an Inertial Measurement Unit (IMU) that includes a gyroscope, three accelerometers and, commonly, a magnetometer. The gyroscope measures the angular velocity of the quadrotor, and, when gyroscope information are integrated with accelerometer measures, it is possible to estimate roll and pitch angles (see [62, 70]). As a result, even low-cost IMUs allow to estimate quadrotor angular velocities and roll and pitch absolute values with reasonable accuracy, at adequate rate.

Making reference to Tab. 4.1, the states corresponding to the three lower rows can be estimated with sensors provided with a common flying platform and a robotic arm.

Conversely, the translational position and velocity of the quadrotor, to-

Table 4.1: *State of the system and sensors that can be employed for their estimation. The sensors marked with an asterisk are selected as the most suitable set to estimate the complete state of a aerial manipulator with a generic purpose.*

State Variables	Sensors
$\boldsymbol{x}, \dot{\boldsymbol{x}}$	GPS, Accelerometer*, Optical flow*, Sonar*, Vision system*, Motion capture, UWB
ψ	Magnetometer, Vision system*, Motion capture, UWB
ϕ, θ	Accelerometer*, Gyroscope*
$\dot{\boldsymbol{\phi}}$	Gyroscope*
$\boldsymbol{q}, \dot{\boldsymbol{q}}$	Robot Arm Encoders*

gether with the yaw position, have to be measured with additional sensors. In fact, the accuracy of GPS sensors generally embedded in common UAVs are far too low than what is required, and they can not be used indoor. The same considerations can be applied to magnetometer, that can measure yaw position just outdoor and with low accuracy.

Sensors that can provide relative, and partial, measures for these variables, are optical flow modules and sonar. Optical flow modules measures lateral and forward velocity of the quadrotor, while sonar gives the relative distance between the quadrotor and the ground.

In order to obtain the inertial localization of the quadrotor, three main groups of technologies should be considered: vision systems, motion capture modules and wireless localization systems as, for instance, Ultra Wide Band (UWB). In Tab. 4.2, the three technologies have been reported with their average performance and main limitations.

Table 4.2: *Technologies for the absolute localization of the quadrotor.*

Localization System	Accuracy	Frequency	Main Limitation
Vision	10^{-2} m	$10-20$ Hz	Lightness conditions
Motion Capture	10^{-3} m	10^2-10^3 Hz	Area predefined, indoor measurement, cost
Wireless Localization	10^{-1} m	10^1-10^3 Hz	Area predefined

Motion capture systems are the technology with the highest performance among the three groups considered. The main limitation of this technology is that they can be used just in a pre-defined area and rigorously indoor. In addition, the implementation cost is generally high.

Wireless localization systems can be considered for the lower cost of implementation with respect to motion capture. A scheme of the existing wireless localization technologies can be found in [69], which is reported in Fig. 4.1. While some examples of Ultra Wide Band localization systems

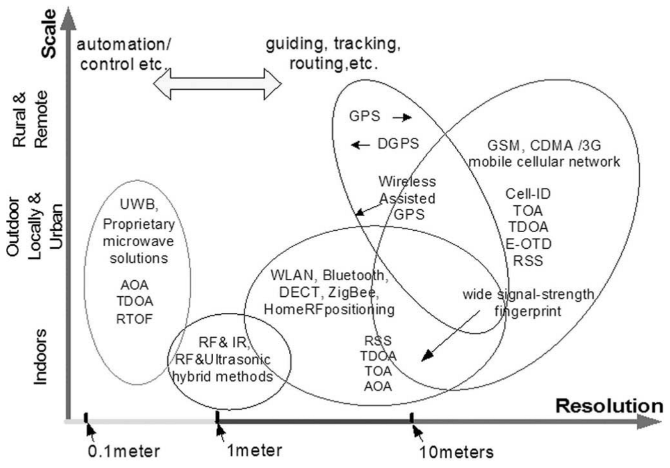

Scale
Rural & Remote
Outdoor Locally & Urban
Indoors
automation/ control etc.
guiding, tracking, routing,etc.
GPS
DGPS
Wireless Assisted GPS
GSM, CDMA /3G mobile cellular network
Cell-ID
TOA
TDOA
E-OTD
RSS
wide signal-strength fingerprint
UWB, Proprietary microwave solutions
AOA
TDOA
RTOF
WLAN, Bluetooth, DECT, ZigBee, HomeRFpositioning
RSS
TDOA
TOA
AOA
RF& IR, RF&Ultrasonic hybrid methods
Resolution
0.1meter
1meter
10meters

Figure 4.1: *Available Wireless localization technologies, from [69].*

under development can get to an accuracy of about 10^{-2} m, the commercial wireless systems are not as accurate as needed for manipulation activities.

Vision systems can estimate position with a reasonable accuracy and frame rate. Since it is possible to fix the camera on board of the aerial manipulator itself, there are no limitations on the workspace. Notice that vision systems can be applied in a structured scenario, by using known target as digital marker, or in unknown and unstructured scenarios by applying ego-localization methods. On the other hand, lightness conditions can affect vision performance in some cases. The frame rate of a common camera is quite low with respect to the other localization technologies, but it can be considered sufficient. In addition, camera images requires a complex post processing to obtain a localization estimate, but a number of state of the art algorithms dealing with the issue are available.

Considering the requirements listed in the previous Section and the characteristics of the considered technologies, vision system is the most reasonable choice to obtain the absolute localization of the quadrotor in an aerial manipulator system with a generic purpose. Therefore, a set of sensors from which it is possible to estimate the complete state of an UAM has been defined. The set includes all the sensors in Tab. 4.1 that are marked with an asterisk.

4.3 Sensors Fusion

The high amount of sensors available to measure the state of the system raises the need of an adequate method of sensors fusion capable of filtering data and estimating the complete state. In addition, sensors measures are noisy, data are given at different frequency rate, and they are characterized by different delays. A proper algorithm has the goal of obtaining a robust estimate of the system state, regardless of temporary sensors failures.

Sensor fusion for UAVs is a topic extensively investigated in the literature, especially in the last decade. Among Kalman based algorithms, a characteristic that distinguishes the different methods is the modeled coupling between sensors output. To this extent, those algorithms can be distinguished in tightly-coupled sensors model, as in [80] and loosely-coupled systems, as in [96] or [37, 76]. The former approaches fuse raw measurements directly, while the latter fuse processed information from individual sensors taking advantage from the modeled decoupling. Alternative method to the Kalman filtering is represented by optimization-based approaches as in [63, 100]. In general, optimization methods consider a tightly-coupled model.

Flying base of UAMs are not meant for aggressive maneuvers, so some assumptions can be made on its motion. Thus, small values approximation can be considered for roll and pitch angles and low velocity for yaw degree of freedom, then a loosely-coupled system can be considered. In particular, it is possible to decouple the tilting estimation (roll and pitch angles), from quadrotor translational position and yaw angle estimation. The two problems will be described in the following two Sections.

4.3.1 Quadrotor attitude

The most used state of the art algorithm to estimate quadrotor attitude is the well known non linear complementary filter on the special orthogonal group, presented in its passive and direct forms in [70]. Different approaches are based on Kalman filtering, as in [62], where even the drag coefficient is considered in the model and estimated.

Since the estimation of the attitude of the aerial manipulator designed in Appendix A is performed with the explicit passive complementary filter with bias correction, the method is here briefly report.
The strength of the method is the possibility of updating the current attitude rotation matrix estimate $\hat{R}$ with vectorial measurements, as they are generally available from IMU. For example, from the accelerometer a measure of

the gravity vector can be obtained, while from vision the heading direction is used to correct the drift for yaw angle.

Model of considered sensors

The sensors employed for the attitude estimation are the gyroscope, the accelerometer and the camera.

Gyroscope The gyroscope provides a measure of the angular velocity of the quadrotor in the body frame $\boldsymbol{\omega}_q^b$, and its output Ω_y can be modeled as follows:

$$\Omega_y = \boldsymbol{\omega}_q^b + \hat{\boldsymbol{b}}_\omega + \boldsymbol{v}_\omega \tag{4.2}$$

where $\hat{\boldsymbol{b}}_\omega$ is the gyroscope bias and $\boldsymbol{v}_\omega$ is the sensor noise.

Accelerometer Three axial accelerometers are used to estimate the direction of gravity vector. In particular, in [62] it is explained in detail why the low frequency component of gravity should be filtered, in order to obtain a unbiased estimate of gravity vector direction $\boldsymbol{m}_g^b$ in body frame.

Camera Output of vision odometry are used to obtain a measure of the quadrotor heading direction, i.e. the vector $\boldsymbol{m}_h^b = R^T\,\boldsymbol{e}_1$ in the body frame, where R is the actual attitude rotation matrix, $\boldsymbol{e}_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ and a East-North-Up (NED) convention is used.

Method

Given a measure of the angular velocity Ω_y from the gyroscope and n_m measures of attitude vectors $\boldsymbol{m}_i$ to be compared with current estimate $\hat{\boldsymbol{m}}_i$, with $n_m \geq 2$, the attitude estimation update is computed as follows:

$$\dot{\hat{R}} = \hat{R}\left(\left(\Omega_y - \hat{\boldsymbol{b}}_\omega\right)_\times + \left(k_p\,\boldsymbol{\omega}_q^b\right)_\times\right) \tag{4.3}$$

$$\dot{\hat{\boldsymbol{b}}}_\omega = -k_I\,\boldsymbol{\omega}_{mes} \tag{4.4}$$

$$\boldsymbol{\omega}_{mes} := \sum_{i=0}^{n_m} k_i \boldsymbol{m}_i \times \hat{\boldsymbol{m}}_i \tag{4.5}$$

where k_P, k_I and k_i are suitable positive scalar gains. For the correct selection of the gains value, refer to [70]. In this framework, considered vectors $\boldsymbol{m}_i$ are the gravity direction $\boldsymbol{m}_g^b$ and the heading direction $\boldsymbol{m}_h^b$.

4.3.2 Absolute Localization

The translational degrees of freedom of the flying base can be estimated with a different approach. As the system is modeled as loosely-coupled, the estimation can be split in the three Cartesian directions, considering the subsystems independent. Therefore, the approach presented in [37, 76] can be used as framework to fuse the available sensors information. In addition, it has the advantage of requiring small implementation effort, as it has been designed on purpose for the experimental platform that will be employed in the work. In fact, a discrete Kalman filter with varying time step system is applied, which relies on the precise timebase provided by the chosen hardware [75].

Sensors model

Sensors that will be employed for the absolute localization of the flying base will be a camera, an optical flow module, and a sonar.

Vision In the context of this work it is assumed that localization is performed in a structured and known environment. Then, it is possible to apply one or more known markers in the environment, and obtain the ego-motion of the camera by applying simple algorithms of marker recognition and visual odometry. In the following, the kinematic transformations needed to obtain an inertial localization of the quadrotor from the camera localization are presented.

The considered frames are the inertial frame W, the quadrotor frame Q, the camera frame C and the marker frame M. The model of the system is simply represented by the roto-translations of the kinematic chain linking the four different frames in Fig. 4.2. The orientation of the quadrotor with respect to the inertial frame $R_W^Q(t)$ can be computed from the measured orientation of the marker with respect to the camera frame $R_C^M(t)$ and pre-multiplying and post-multiplying the constant marker orientation with respect to the inertial frame $\bar{R}_W^M$ and the constant quadrotor orientation with respect the camera frame $\left(\bar{R}_Q^C\right)^T$, respectively:

$$R_W^Q(t) = \bar{R}_W^M R_C^M(t)^T \left(\bar{R}_Q^C\right)^T \tag{4.6}$$

Similar to the orientation, we can close the translational kinematic chain by computing the relative position between the inertial frame and the quadrotor x_{WQ}^W [1], expressed in absolute frame, by composing the translation between

[1]Notice that x_{IJ}^K represents the vector from frame I to frame J expressed in frame K.

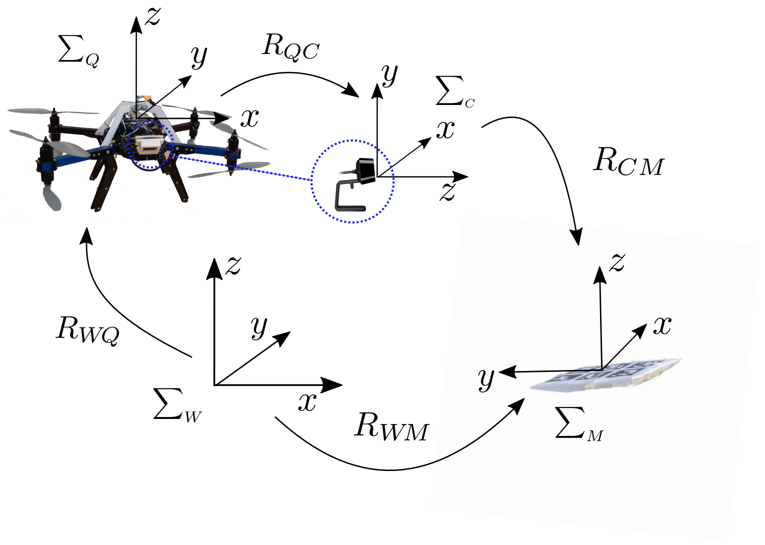


Figure 4.2: *Kinematic chain of the system between the inertial frame W, the frame of the quadrotor Q, the frame of the camera C and the frame of the marker M. Notice that the camera is fixed to the quadrotor, while the two images are disjoint for the sake of clarity.*

the inertial frame and the marker x^W_{WM}, the relative position between the marker and the camera x^W_{CM}, and finally the relative position between the camera and the quadrotor x^W_{QC}, as follows:

$$x^W_{WQ}(t) = \bar{x}^W_{WM} - R^Q_W(t)\,\bar{R}^C_Q \bar{x}^C_{CM} - R^Q_W(t)\,\bar{x}^Q_{QC}, \tag{4.7}$$

where the following relations have been used:

$$x^W_{CM} = R^Q_W(t)\,\bar{R}^C_Q\,x^C_{CM}(t)\,, \quad x^W_{QC} = R^Q_W(t)\,\bar{x}^Q_{QC}. \tag{4.8}$$

The matrix $R^M_C\,(t)$ and the translation of $x^W_{WE}(t)$ can be obtained from the monocular vision odometry algorithm, while the constant terms $\bar{R}^C_E$ and $\bar{R}^M_W$, $\bar{x}^W_{WM}$ and $\bar{x}^E_{EC}$ are assumed to be known from a calibration procedure, which will be presented in Sec. 4.4. Therefore, a direct measure of the inertial position and rotation of the quadrotor frame $x^W_{WQ}\,(t)$ and $R^Q_W(t)$ can be obtained by means of vision odometry algorithm.

Optical Flow Optical flow modules is composed of a camera, a distance sensor, which can either consist on a lidar or a sonar, and a gyroscope, and it can measure its own translational velocity from the motion of pixels in the image. The angular velocity given by the gyroscope is combined with the information obtained by the distance sensor to cancel the pixel motion component due to the rotation of the module itself, and estimate an unbiased translational velocity. The output value of an optical flow module is its own translational motion velocity, in the directions orthogonal to the camera axis. Through a simple kinematic transformation, it is possible to obtain an estimate of the quadrotor velocity in the horizontal plane. In fact, given $\boldsymbol{v}_f^b \in \mathbb{R}^3$ the optical flow velocity measure, where the third component is null, $\boldsymbol{\omega}_q^b \in \mathbb{R}^3$ the angular velocity of the quadrotor, $\boldsymbol{r}^b \in \mathbb{R}^3$ the offset of the optical flow sensor with respect to the quadrotor geometrical center, $\boldsymbol{v}_q^b \in \mathbb{R}^3$ the quadrotor velocity in the body frame, the first two components of quadrotor velocity in the absolute frame $\boldsymbol{v}_{q,xy}^w \in \mathbb{R}^2$ can be computed as follows:

$$\boldsymbol{v}_q^b = \boldsymbol{v}_f^b - \boldsymbol{\omega}_q^b \times \boldsymbol{r}^b \tag{4.9}$$

$$\boldsymbol{v}_{q,xy}^w = S_{xy}\, R_W^Q\, \boldsymbol{v}_q^b \tag{4.10}$$

where S_{xy} is a selection matrix that extracts the x and y components of the measured quadrotor velocity. Notice that, considering the small angles approximation, just the yaw angle plays a role in the rotation matrix of Eq. (4.10).

Distance sensor The distance sensor embedded in the optical flow module provides as output a measure of its relative distance with the ground. When the sensor is mounted with an offset $\boldsymbol{r}^b$ with respect to the quadrotor geometrical center, and the quadrotor attitude is described by rotation matrix R_W^Q, the measured distance d_s is reported to the absolute quadrotor altitude h_s by the simple following relation:

$$h_s = \boldsymbol{e}_3^T R_W^Q \left(\boldsymbol{e}_3\, d_s - \boldsymbol{r}^b\right) \tag{4.11}$$

where $\boldsymbol{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

Method

A simple linear Kalman filter is used to estimate the translational states of the quadrotor and the yaw. The method is based on the work in [76]. The states to be estimated are the following:

$$\boldsymbol{x}_{q,k}^a = \begin{bmatrix} x_{q,k} \\ \dot{x}_{q,k} \end{bmatrix} \quad \boldsymbol{y}_{q,k}^a = \begin{bmatrix} y_{q,k} \\ \dot{y}_{q,k} \end{bmatrix} \quad \boldsymbol{z}_{q,k}^a = \begin{bmatrix} z_{q,k} \\ \dot{z}_{q,k} \end{bmatrix} \quad \boldsymbol{\psi}_{q,k}^a = \begin{bmatrix} \psi_{q,k} \\ \dot{\psi}_{q,k} \end{bmatrix} \tag{4.12}$$

The adopted model is decoupled, linear with no input:

$$\boldsymbol{x}^a_{q,k+1} = A\boldsymbol{x}^a_{q,k} + \boldsymbol{w}_{x,k} \tag{4.13}$$

$$\boldsymbol{y}_{s_x,k} = C_x\boldsymbol{x}^a_{q,k} + \boldsymbol{v}_{x,k} \tag{4.14}$$

where $A = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$, $\boldsymbol{w}_{x,k}$ is a Gaussian process disturbance, $\boldsymbol{v}_{x,k}$ is a Gaussian sensor noise, $\boldsymbol{y}_{s_x}$ is the sensor output for the state $\boldsymbol{x}^a_q$, and the model is applied to the other three states similarly. Notice that the model assumes constant velocity. It could be extended with accelerometer information, but measures are noisy and no significant improvement can be appreciated.
The prediction step for the Kalman filter is the usual one:

$$\begin{aligned} \hat{\boldsymbol{x}}^a_{q,k} &= A\hat{\boldsymbol{x}}^a_{q,k-1} \\ P_{x,k} &= AP_{x,k-1}A^T + Q_x \end{aligned} \tag{4.15}$$

where $\hat{\boldsymbol{x}}^a_{q,k}$ is the estimate at step k, $Q = \text{var}\left(\boldsymbol{v}_{x,k}\right)$ is the process covariance matrix and $P_{x,k}$ is the filter covariance matrix. The correction step is performed as follows:

$$\begin{aligned} K_{x,k} &= P_{x,k}C_x^T\left(C_xP_{x,k}C_x^T + R_{x,k}\right)^{-1} \\ \hat{\boldsymbol{x}}^a_{q,k+1} &= \hat{\boldsymbol{x}}^a_{q,k} + K_{x,k}(\boldsymbol{y}_{s_x,k} - C_x\hat{\boldsymbol{x}}^a_{q,k}) \\ P_{x,k+1} &= \left(I - K_{x,k}C_x\right)P_{x,k} \end{aligned} \tag{4.16}$$

where $R_{x,k}$ is the noise covariance matrix, that is assumed diagonal, and $K_{x,k}$ is the Kalman gain.

Notice that the form of matrix C_i and, consequently, of noise covariance matrix and output vector $\boldsymbol{y}_{s_i}$, depends on the sensors used for the specific state i. In particular, for states $\boldsymbol{x}^a_q$ and $\boldsymbol{y}^a_q$, the measures considered will be the vision odometry output for the position and the optical flow measure for the velocity. The state $\boldsymbol{z}^a_q$ will be estimated fusing the odometry output and the distance sensor, while the measures available for the yaw state $\boldsymbol{\psi}^a_{q,k}$ are the one obtained by the camera, and the gyroscope information for its derivative.
Notice that sensors are available at different frame rate. In this case, it is a common method to perform the correction step every time that a measure is available. Therefore, matrices C_i and $R_{x,k}$ take the suitable structure, related to the specific sensor measure. This approach is not in contrast with system in (4.16), where C_x are not changing with step k. In fact, when sensors information are not available, suitable diagonal terms of matrix $R_{x,k}$

take infinite value, and corresponding elements of $\boldsymbol{y}_{s_x,k} - C_x\hat{\boldsymbol{x}}^a_{q,k}$ are ignored by the correction phase.

4.4 Camera Calibration with an Industrial Robot

In the previous Section, the output of visual odometry algorithm were used to obtain an absolute localization of the quadrotor. This operation, which converts a relative information to an absolute localization, can be applied if a preliminary calibration of the camera has been performed.

In order to perform an extrinsic calibration of a camera, and then evaluate its performance, a Ground Truth is required. For visual odometry systems, common Ground Truth can be classified as sensor-based systems, physics-based simulation, and platform-based (or fixture-based) systems, as in [43]. For sensor based systems several different sensors can be used, including radio frequency, optics and acoustics. A very popular technology used in research lab is the camera-based motion capture system. However, in industry and in the majority of the research labs, expensive sensors as motion capture system are not available, thus it is difficult to evaluate the performance of the vision system. On the other hand, the uncertainty of such system is a critical parameter for most of its applications.

A possible alternative is to use an industrial robot in order to calibrate a camera and to validate a visual odometry algorithm. In particular, the relative pose of the camera target object to the robot and the pose of the camera with respect the end effector of the robot itself will be estimated by means of a quadratic optimization with quadratic constraints, which is commonly called Quadratically Constrained Quadratic Program (QCQP). Once these relative poses are computed, the operation presented in the pevious Section can be performed, and the visual odometry algorithm can be evaluated in different conditions. The proposed approach is presented in the following.

4.4.1 Modeling

The system is composed of an industrial robot, to which the vision sensor is mounted, generally at its end effector, by means of a support. The roto-translation between the frame at the end effector and the frame of the camera is not known precisely. The system is completed with a marker, whose relative pose with the camera can be estimated by the odometry algorithm. The exact pose of the marker in the absolute frame is not known precisely either. This is a common industrial setup, where a visual odometry system is available, usually by means of cameras mounted on the robot, but there are not complex or expensive structures as motion capture systems to validate the acquired measures.

The model of the system is simply represented by the roto-translations of the kinematic chain linking the four different frames in Fig. 4.3. The orientation of the camera with respect to the inertial frame $R_W^C(t)$ can be

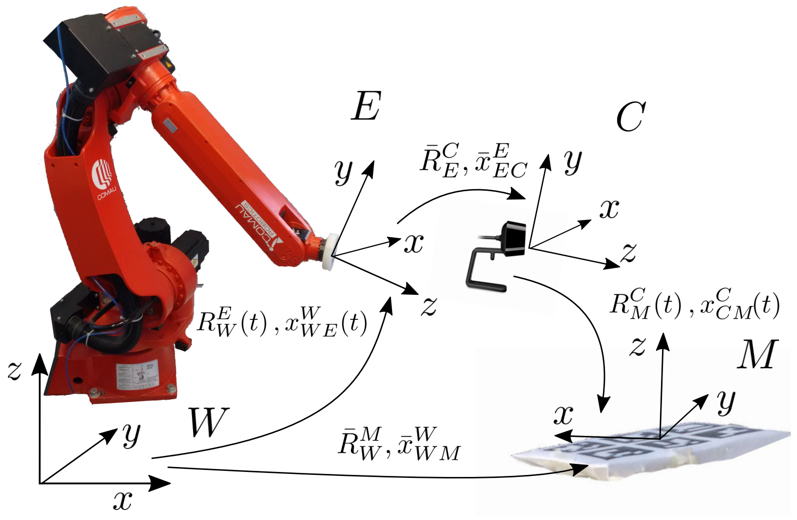


Figure 4.3: *Kinematic chain of the system between the inertial frame W, the frame of the robot end effector E, the frame of the camera C and the frame of the marker M.*

computed from the orientation of the end effector with respect to the inertial frame $R_W^E(t)$ and the constant camera orientation with respect to the end effector frame $\bar{R}_E^C$. Equivalently, it can be computed from the constant marker orientation with respect to the inertial frame $\bar{R}_W^M$ and the camera orientation with respect to the marker frame $R_C^M(t)^T$, as follows:

$$R_W^C(t) = R_W^E(t)\,\bar{R}_E^C = \bar{R}_W^M R_C^M(t)^T \tag{4.17}$$

Similar to the orientation, we can close the translational kinematic chain by computing the relative position between the inertial frame and the marker x_{WM}^W, expressed in absolute frame, by summing the translation between the inertial frame and the end effector x_{WE}^W, the relative position between the end effector and the camera x_{EC}^W, and finally the relative position between the camera and the marker x_{CM}^W, as follows:

$$\bar{x}_{WM}^W = x_{WE}^W(t) + R_W^E(t)\,\bar{x}_{EC}^E + R_W^E(t)\,\bar{R}_E^C\,x_{CM}^C(t)\,, \tag{4.18}$$

where the following relations have been used:

$$x^W_{EC} = R^E_W(t)\,\bar{x}^E_{EC}, \quad x^W_{CM} = R^E_W(t)\,\bar{R}^C_E\,x^C_{CM}(t)\,.$$

The matrices $R^E_W\,(t)$ and $R^M_C\,(t)$ can be measured from the robot encoders (applying the direct kinematic of the robot) and the monocular vision odometry algorithm, respectively, while the other two matrices $\bar{R}^C_E$ and $\bar{R}^M_W$ are not known with sufficient accuracy. Equivalently, the measures of $x^W_{WE}(t)$ and $x^C_{CM}(t)$ are available, but no precise information of the constant terms $\bar{x}^W_{WM}$ and $\bar{x}^E_{EC}$ are available.
We want to accurately estimate the matrices $\bar{R}^C_E$ and $\bar{R}^M_W$ and the vectors $\bar{x}^W_{WM}$ and $\bar{x}^E_{EC}$: in this way an unbiased comparison between the camera pose estimated with robots sensors and the pose estimated with vision odometry algorithm can be performed.

4.4.2 Approach

The goal of the present Section is to propose a method to estimate the unknown matrices $\bar{R}^C_E$, $\bar{R}^M_W$, and the vectors $\bar{x}^W_{WM}$, $\bar{x}^E_{EC}$. The problem can be defined as “triangularly” decoupled, because translational terms are not affecting the relation in (4.17). Thus, it is possible to estimate the rotational terms independently from the translational terms. We present how to estimate the unknown rotational and translational terms separately, and finally we integrate the two estimations in a single one.

Identification of the Rotational Terms

The main difficulty of the matrices estimation of the presented problem is given by the fact that the two unknown rotations are not consecutive one to the other. In fact, the model in (4.17) can be written as follows:

$$A_e(t)\,X + Y\,B_c(t) = 0 \tag{4.19}$$

where $A_e(t) = R^E_W(t)$, $B_c(t) = -R^M_C(t)^T$, $X = \bar{R}^C_E$, $Y = \bar{R}^M_W$. The given equation has not a closed form solution suitable for the estimation of the unknown matrices. The key aspect of the proposed method is to write the relation in (4.19) in the form of a Sylvester equation and then to apply its closed form solution.

In order to obtain the equivalent Sylvester equation, the change of variables $Z = \begin{bmatrix} X & 0 \\ 0 & Y \end{bmatrix}$ can be performed and (4.17) can be reformulated as:

$$
\begin{aligned}
A(t)\, Z + Z\, B(t) &= 0 \\
A(t) &= \begin{bmatrix} 0 & 0 \\ A_e(t) & 0 \end{bmatrix} \\
B(t) &= \begin{bmatrix} 0 & 0 \\ B_c(t) & 0 \end{bmatrix}
\end{aligned}
$$

where $A \in \mathbb{R}^{6\times 6},\ B \in \mathbb{R}^{6\times 6}$.
The previous equation has a useful closed form solution. In fact, it is equivalent to:

$$
\begin{aligned}
G_k(t)\, z_k &= 0 \\
G_k(t) &= I_6 \otimes A(t) + B(t)^T \otimes I_6
\end{aligned}
$$

where $\otimes$ applies the Kronecker product, $G_k(t) \in \mathbb{R}^{36\times 36}$ and $z_k = \text{vec}\,(Z) \in \mathbb{R}^{36}$ is a column vector built with all the columns of Z. The obtained form allows to compute z_k as the solution of an optimization problem that minimizes the product $G_k(t)\, z_k = 0$. The vector z_k is built as $z_k = \begin{bmatrix} X_1^T & 0_3^T & X_2^T & \dots & Y_3^T \end{bmatrix}^T$ where $0_3 \in \mathbb{R}^3$ are null vectors and X_i, Y_i are the columns of X and Y, respectively. Thus, it is possible to reduce the dimension of the problem by removing the elements of z_k that are null by construction and the corresponding column of G_k, obtaining $z \in \mathbb{R}^{18}, \quad z = \begin{bmatrix} X_1^T & X_2^T & \dots & Y_3^T \end{bmatrix}^T$ and $G \in \mathbb{R}^{36\times 18}$.

When a series of measures for the rotation matrices $R_W^E\,(t)$ and $R_C^M\,(t)$ are available, i.e. the robot end effector performs a trajectory in space with different orientations, the matrix $G(t)$ can be evaluated in different configurations.

Therefore, with a time series of N data, the problem of estimating z can be formulated as follows:

$$
\min_z \|G_S\, z\|
$$

where $G_S = \begin{bmatrix} G(t_1)^T & G(t_2)^T & \dots & G(t_N)^T \end{bmatrix}^T$. Matrices X and Y are orthogonal, and this will become a constraint of the optimization problem. It is easy to formulate the orthogonality constraints on the columns of X and Y in the shape of $z^T Q_i z = r_i$ and $r_i = 1$ or $r_i = 0$ with $i = 1, \dots, 12$ (six constraints for both matrices X and Y). The matrix Q_i will be built such that the constraint i is in the form $X_j^T X_k = r_i$ (or $Y_j^T Y_k = r_i$), where

r_i is equal to 1 if j is equal to k or it is equal to 0 otherwise.
Finally, the problem is a QCQP:

$$\begin{aligned} &\min_z \|G_S\, z\| \\ \text{subject to:} \quad & z^T Q_i z = r_i \quad i = 1, \dots, 12 \end{aligned}$$

This class of problems can be solved with existing commercial or open source solvers.
Notice that the analyzed case is a non-convex problem, thus the choice of the initial solution z_0 is particularly critical. Nevertheless, a sufficiently good estimate of the unknown term to be used as initialization is in general available. In fact, an initial reasonable value of the matrices $\bar{R}_E^C$ and $\bar{R}_W^M$ can be obtained by simply comparing the frames of marker, camera and robot end effector.

Identification of the Translational Terms

The identification of the matrix $\bar{R}_E^C$ allows to compute the unknown terms $\bar{x}_{WM}^W$ and $\bar{x}_{EC}^E$. In fact, the equation in (4.18) can be written as follows:

$$D(t)\, x = b(t) \tag{4.20}$$

where $D(t) \in \mathbb{R}^{3\times 6}$, $x \in \mathbb{R}^6$, $b(t) \in \mathbb{R}^6$:

$$x = \begin{bmatrix} \bar{x}_{WM}^W \\ \bar{x}_{EC}^E \end{bmatrix}, \; b(t) = x_{WE}^W\,(t) + R_W^E\,(t)\, \bar{R}_E^C\,(t)\, x_{CM}^C\,(t)\,, \tag{4.21}$$

$$D(t) = \begin{bmatrix} I_3 & -R_W^E\,(t) \end{bmatrix} \tag{4.22}$$

With a time series of N data, the following system can be built:

$$D_S x = b_S \tag{4.23}$$

where:

$$D_S = \begin{bmatrix} D(t_1)^T & D(t_2)^T & \dots & D(t_N)^T \end{bmatrix}^T \tag{4.24}$$

$$b_S = \begin{bmatrix} b(t_1)^T & b(t_2)^T & \dots & b(t_N)^T \end{bmatrix}^T \tag{4.25}$$

Finally, an estimate of the unknown term x can be easily computed:

$$x = \left(D_S^T D_S\right)^{-1} D_S^T\, b_S \tag{4.26}$$

Simultaneous Identification of the Rotational and Translational Terms

Obviously the minimizations performed in the previous Sections, which estimate rotational and translational terms, can be performed together. The procedure is just more involved in building the required matrices.

Considering the relation in (4.18), it is equivalent to:

$$D(t)\,x - H(t)\begin{bmatrix}X_1^T & X_2^T & X_3^T\end{bmatrix}^T = x_{WE}^W(t) \tag{4.27}$$

where $H(t) \in \mathbb{R}^{3\times 9}$ is defined as follows:

$$H(t) = x_{CM}^C(t)^T \otimes R_W^E(t)\,, \tag{4.28}$$

the matrix $D(t)$ and x have been defined in (4.21) and X_i are the columns of X, which is the unknown matrix corresponding to $\bar{R}_E^C$. Notice that the product $H\,(t)\begin{bmatrix}X_1^T & X_2^T & X_3^T\end{bmatrix}^T$ is equal to the product $R_W^E(t)\,\bar{R}_E^C\,x_{CM}^C(t)$. It is now straightforward to build the matrices of the extended problem. In fact, defining $y = \begin{bmatrix}z^T & x^T\end{bmatrix}^T$, the problem has now this form:

$$G_y(t)\,y - b_y(t) = 0 \tag{4.29}$$

where $G_y(t) \in \mathbb{R}^{39\times 24}$ and $b_y(t) \in \mathbb{R}^{39}$. Matrix G_y and b_y are built by expanding matrix G in order to include the equation in (4.27):

$$G_y(t) = \begin{bmatrix} G(t) & & 0_{36\times 6} \\ -H(t) & 0_{3\times 9} & D(t)\end{bmatrix} \tag{4.30}$$

$$b_y(t) = \begin{bmatrix} 0_{36} \\ x_{WE}^W(t)\end{bmatrix}. \tag{4.31}$$

When a time series of data is acquired, the optimization can be written as follows:

$$\begin{aligned} &\min_y \|G_{yS}\,y - b_{yS}\| \\ \text{subject to:}\quad & z^T Q_i z = r_i \quad i = 1,\dots,12 \end{aligned}$$

where $G_{yS} = \begin{bmatrix}G_y(t_1)^T & G_y(t_2)^T & \dots & G_y(t_N)^T\end{bmatrix}^T$ and $b_{yS} = \begin{bmatrix}b_y(t_1)^T & b_y(t_2)^T & \dots & b_y(t_N)^T\end{bmatrix}^T$. The constraints are expressed in the variables z, which is a sub-part of the variables y, just for clarity of exposition. The problem is still a QCQP, with a slightly bigger dimension. The proposed method is preferable to the two separate optimizations because more information are used to obtain the estimate of the required rotation matrices.

Notice that the overall method presented is applicable even when the target of the sensor (the marker) is mounted on the industrial robot, and the camera is fixed in the environment. This is another common situation for industries where, for example, surveillance cameras are used to monitor the workspace.

4.4.3 Validation

The validation of the method has been performed in both the described situations. In the first case, the marker was mounted on the end effector of the robot and the camera was fixed to the ground, while in the second case the camera was in the eye-in-hand configuration, fixed to the end effector, while the marker was on the ground. In Fig. 4.4, the second configuration is reported. The experiments have been performed employing the facilities of the Mechatronics and Robotics Laboratory for Innovation at Politecnico di Milano [7]. An industrial robot Comau Smart Six is used, on which a

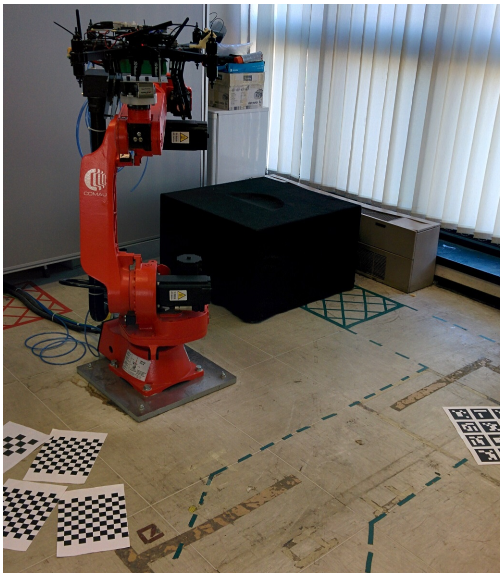


Figure 4.4: *Experimental Set-Up. A quadrotor is fixed to the end effector of the Comau Smart Six and a marker is on the ground.*

quadrotor X8+ 3DR is mounted. A camera Microsoft HD 5000 (1280×720 pixels) is fixed to the quadrotor. Notice that the base of the camera on the quadrotor is rotated of about $45°$ around the lateral axis of the quadrotor in the horizontal plane (the y axis in a NED reference frame).

The image analysis is performed on board, on a ODROID XU3, using the open source software ARUCO [40]. The data sheet of the robot reports that its repeatability is less than 0.05 mm.

For both cases of eye-in-hand and fixed camera, a 3D trajectory is imposed to the robot, while the camera acquires the relative pose of the marker. The acquired data are synchronized, under-sampled (the odometry algorithm has a frame rate of 20 Hz while the encoders data can be read at 500 Hz) and passed to the optimization algorithm described in the previous Section. The minimization is performed in order to obtain the required roto-translations. The results of the optimization relative to the second case have been reported (this case is more interesting because from the obtained parameters it is possible to recognize the geometry of the quadrotor and of the camera support, useful to the localization problem mentioned in Sec. 4.3.2):

$$\begin{aligned}
\bar{R}_W^M &= \begin{bmatrix} 0.9996 & -0.0099 & 0.0251 \\ 0.0095 & 0.9998 & 0.0160 \\ -0.0252 & -0.0157 & 0.9996 \end{bmatrix}, \\
\bar{R}_E^C &= \bar{R}_E^Q \bar{R}_Q^C \\
\bar{R}_E^Q &= \begin{bmatrix} -0.7181 & 0.6960 & 0 \\ -0.6960 & -0.7181 & 0 \\ 0 & 0 & 1.0000 \end{bmatrix} \\
\bar{R}_Q^C &= \begin{bmatrix} 0.9999 & -0.0116 & -0.0110 \\ 0.0000 & -0.6875 & 0.7262 \\ -0.0160 & -0.7261 & -0.6874 \end{bmatrix} \\
\bar{x}_{WM}^W &= \begin{bmatrix} 0.7901 & 1.3055 & -0.0615 \end{bmatrix}^T \\
\bar{x}_{EC}^E &= \bar{x}_{EQ}^E + \bar{R}_E^Q \bar{x}_{QC}^Q \\
\bar{x}_{EQ}^E &= \begin{bmatrix} 0 & 0.03 & 0.175 \end{bmatrix}^T \quad \bar{x}_{QC}^Q = \begin{bmatrix} 0 & 0.17 & 0.04 \end{bmatrix}^T
\end{aligned}$$

Notice that the roto-translation between the end effector and the camera has been divided in two steps, in order to obtain the transformation between the camera and the centre of the quadrotor, useful for the navigation. The results obtained by the minimization have been used for evaluating the performance of the visual odometry algorithm.

The first validation experiments were addressed to test the algorithm and the hardware of the case study in a number of different conditions. The same trajectories have been performed by the robot, while the following conditions have been tested:

i. effect of direct lightening or back lit;

ii. rotation of the marker with respect the camera;

	Mean Error [m]			Standard Deviation [m]		
	x	y	z	x	y	z
A	$0.5833e^{-03}$	$0.0194e^{-03}$	$0.9866e^{-03}$	0.0053	0.0023	0.0054
B	$-0.8631e^{-03}$	$-0.5880e^{-03}$	$0.3640e^{-03}$	0.0084	0.0060	0.0075
C	$0.2401e^{-03}$	$0.0043e^{-03}$	$0.3376e^{-03}$	0.0054	0.0017	0.0012
D	$-0.5990e^{-04}$	$0.2685e^{-04}$	$-0.3365e^{-04}$	0.0045	0.0070	0.0012
E	$-0.2269e^{-04}$	$0.6160e^{-04}$	$0.6523e^{-04}$	0.0060	0.0020	0.0036

Table 4.3: *Mean Error and Standard Deviation relative to Camera Position - Fixed Camera.*

	Mean Error [rad]			Standard Deviation [rad]		
	ψ	θ	ϕ	ψ	θ	ϕ
A	-0.0041	-0.0027	-0.0001	0.0438	0.0438	0.0099
B	-0.0106	-0.0077	-0.0010	0.0348	0.0216	0.0093
C	0.0020	-0.0010	-0.0014	0.0111	0.0089	0.0034
D	-0.0017	-0.0026	0.0003	0.0083	0.0098	0.0024
E	0.0018	0.0002	-0.0017	0.0253	0.0183	0.0034

Table 4.4: *Mean Error and Standard Deviation relative to Camera Euler Angles - Fixed Camera.*

iii. testing the lens distortion by keeping the marker near the edges of the camera image.

We are now presenting the results of the case with fixed camera and marker mounted on the robot. The results have been reported in Tables 4.3 and 4.4, where condition B corresponds to edge effect, C is affected by the rotation of the marker (that causes a reduced visibility of its shape), while cases A, D and E represent different lightening conditions (lateral, back lit and direct light, respectively). The computed errors are referred to the pose of the marker, acquired by the camera and by the encoders of the robot. Rotations are represented with yaw-pitch-roll Euler angles. It is possible to notice that the estimation uncertainty shows small variation in the different tests. The estimation seems to be particularly affected by the relative position of the marker to the edges of the camera image (test B). In Fig. 4.5 and 4.6 the trajectories computed from encoder data (subscript r in the legend of the plot) and from the visual odometry algorithm (subscript c) are compared. The system is in the condition A. Again, rotations are represented with yaw-pitch-roll Euler angles. The overlapping of the two data sets is clear both for translation and rotation coordinates.
The procedure relative to the case of eye-in-hand camera is exactly the same as the previous one. Therefore, it is interesting to compare the numerical

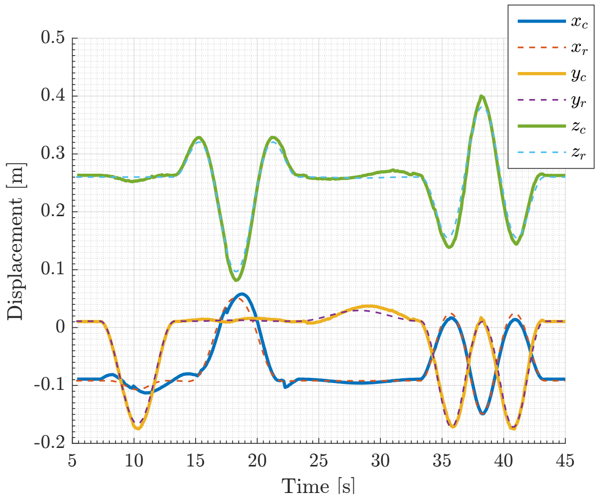


Figure 4.5: *Comparison between the position of the camera computed from the encoders measurements, and the position result of the odometry algorithm.*

	Mean Error			Standard Deviation		
	$x-\psi$	$y-\theta$	$z-\phi$	$x-\psi$	$y-\theta$	$z-\phi$
Translation [m]	0.0024	-0.0078	-0.0092	0.0184	0.0153	0.0169
Rotation [rad]	0.0216	0.0303	-0.0071	0.0056	0.0101	0.0156

Table 4.5: *Mean Error and Standard Deviation - Eye-in-Hand Camera.*

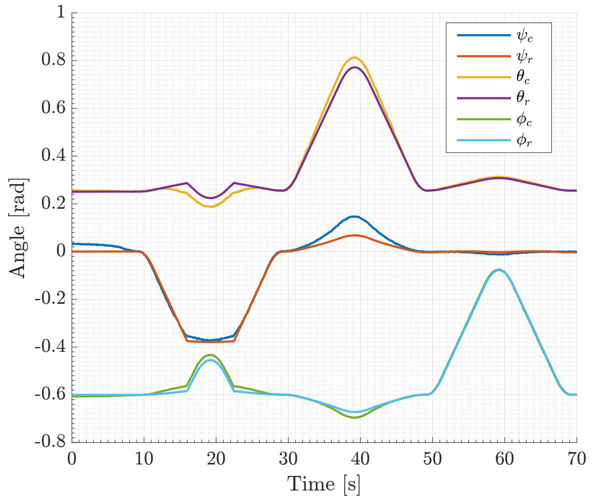


Figure 4.6: *Comparison between the Euler angles describing the attitude of the camera computed from the encoders measurements, and the Euler angles result of the odometry algorithm.*

results describing the estimation error, obtained in the validation tests, with the values of the errors of the previous fixed-camera case. In this case, the comparison is performed on the pose of the camera itself.

In Tab. 4.5 the estimation error is reported. As expected, the mean error of the egomotion estimation problem is greater than the case of fixed camera and moving marker.

4.5 Implementation and Validation

The estimation algorithms presented in Sec. 4.3 have been implemented on an aerial manipulator system. In particular, the complete experimental setup have been described in Chapter A.

The localization of the quadrotor have been performed with a camera, an optical flow sensor module and a sonar. A camera Microsoft HD 5000 (1280 $\times$ 720 pixels) is fixed to the quadrotor, and the relative pose of the camera with respect to the geometrical center of the quadrotor has been obtained in the previous Section. The needed rotation and translation between the quadrotor and the camera are reported here:

$$\bar{R}_Q^C = \begin{bmatrix} 0.9999 & -0.0116 & -0.0110 \\ 0.0000 & -0.6875 & 0.7262 \\ -0.0160 & -0.7261 & -0.6874 \end{bmatrix}, \quad \bar{x}_{QC}^Q = \begin{bmatrix} 0 \\ 0.17 \\ 0.04 \end{bmatrix}$$

	Mean Error			Standard Deviation		
	$x-\psi$	$y-\theta$	$z-\phi$	$x-\psi$	$y-\theta$	$z-\phi$
Translation [m]	0.0031	-0.0092	-0.0122	0.0224	0.0164	0.0187
Rotation [rad]	0.0253	0.0415	-0.0112	0.0370	0.0401	0.0173

Table 4.6: *Mean Error and Standard Deviation - Quadrotor position estimation with Kalman filter.*

Notice that quadrotor frame follows the ENU convention. The PX4Flow module is installed in the back of the quadrotor, and it integrates a 752×480 pixels camera as an optical flow sensor and a sonar Maxbotix MB1043. The relative position of the optical flow camera with respect the quadrotor center is the following:

$$\bar{x}^{Q}_{QO} = \begin{bmatrix} 0 \\ -0.20 \\ -0.02 \end{bmatrix}$$

The module includes a 168 MHz Cortex M4F CPU and a L3GD20 3D gyroscope, such that a pre-conditioned velocity information is obtained by the module. The image analysis is performed on board, on a ODROID XU3, using the open source software ARUCO [40], which is based on OpenCV libraries.

4.5.1 Validation on an Industrial Robot

Similar to Sec. 4.4, the quadrotor is mounted on a industrial robot, and a prescribed trajectory at end effector is performed. By applying the procedure presented above, it is possible to compare the quadrotor positions obtained by the arm encoders' measures, and the estimated position computed with the Kalman filter, as in Fig. 4.7. Notice that estimated quadrotor position appears more noisy if compared with Fig. 4.5, where just the output of the vision localization where used. Mean error and standard deviation of the estimated quadrotor position have been reported in Tab. 4.6. Thus, the fusion of camera information with optical flow sensor and sonar slightly increases the error variance. However, the use of optical flow and sonar is necessary, in order to have an high frequency information for the velocity, and to be robust to temporary occlusion of the marker and, in general, unavailability of the camera measures.

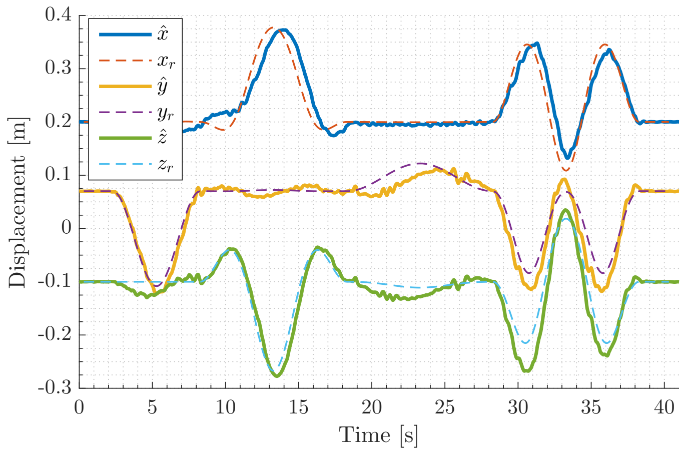


Figure 4.7: *Comparison between the position of the camera computed from the encoders measurements, represented with dashed lines, and the position estimated with the Kalman filter algorithm.*

4.5.2 Validation on a Flight Experiment

The estimation algorithm of the quadrotor localization has been employed for the autonomous flight of the system. In Fig. 4.8, the estimated quadrotor positions are reported. Note that a series of step signal are assigned as reference displacement, which are represented with dashed lines in the figure. There is no ground truth available to compare the obtained estimated data. In the flight experiment, estimated signals are more noisy than the result obtained during the algorithm validation on the industrial robot. Notice that two concurrent effects are relevant to this behavior. First, it is not possible to distinguish the oscillations due to erroneous estimation from the real motion of the platform caused by disturbances. Second, propellers rotation induces a vibration of the structure that is disadvantageous for the camera image acquisition, and it results in a decrease of the estimation performance.

For completeness, the quadrotor Euler angles estimated during the flight experiments have been reported in Fig. 4.9. Just a detail has been extracted for the sake of clarity. Remind that roll and pitch angles are used as control variable to drive x and y translation of the quadrotor, and yaw angle is regulated to zero. It can be noticed the difference in the main frequency components of roll and pitch angles with respect to yaw. Yaw absolute position is obtained with vision image, thus at lower frequency. Therefore, the control loop is tuned accordingly, showing a lower bandwidth disturbance

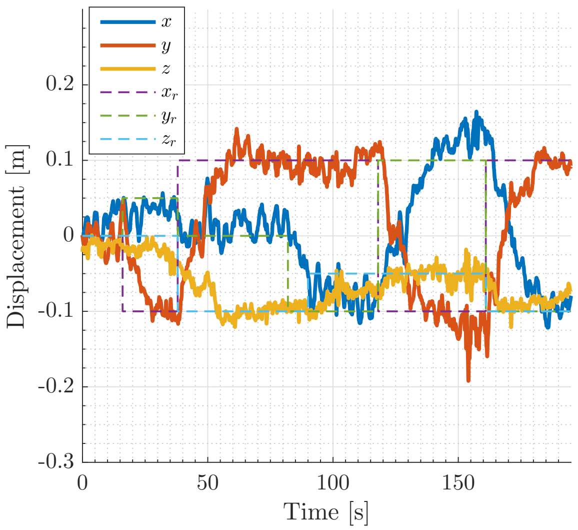


Figure 4.8: *Estimated quadrotor displacement during a flight experiment. A series of step references have been assigned to the position control. Estimated positions and reference signal are represented with solid and dashed lines, respectively.*

rejection.

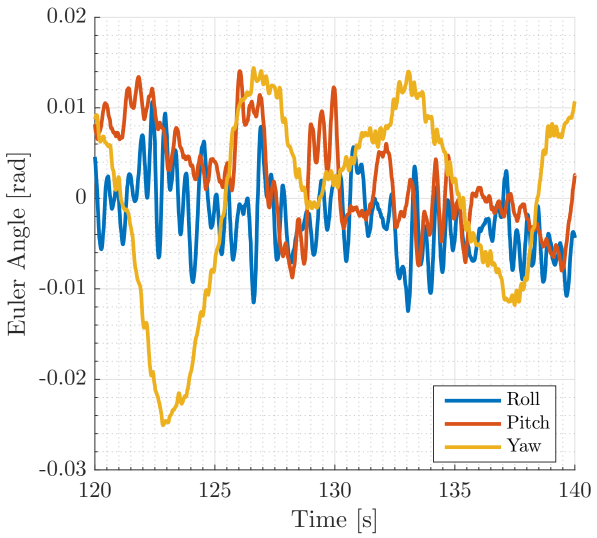


Figure 4.9: *Estimated quadrotor Euler angles during a flight experiment. Just a detail has been reported for the sake of clarity.*

CHAPTER 5

A Contribution to the Control of Interaction

5.1 Introduction

In order to enable aerial manipulation activities, a critical and challenging research issue concerns the physical interaction between the aerial system and the external environment. When an aerial manipulator comes into contact with the external environment, several factors should be considered in order to achieve a stable and effective manipulation. In particular, when an UAM is performing physical interaction, additional kinematic constraints can be established, which change the overall system dynamic behavior. Moreover, force and torque disturbances applied to the aircraft can undermine the stability of the system, when the maximum torques of the aircraft are not sufficiently high. In addition, the oscillations of the flying platform due, for instance, to wind disturbances have to be measured and compensated to accurately control the interaction dynamics.

Grasping and contact control have been firstly investigated in [78, 87], where robotic grippers directly installed on the aerial vehicle are employed. Guidelines on the gain tuning of the aircraft attitude controller are provided to preserve stability. The majority of state of the art approaches in the field of interaction control for aerial manipulation aim at providing a compliant behavior to the UAM when in contact with the external environment. To this extent, mechanical or control based solutions have been proposed. In [88,89] compliant grippers have been mounted on aerial vehicles in order to compensate positioning errors and to prevent high contact forces. Novel manipulator prototypes specifically designed to be mounted on an aerial vehicle have been proposed in a number of works [14,26,27,51,58,103,105]. In particular, the arms are designed to provide the required degrees of freedom to compensate the displacements of the robotic platform, even with hyper-redundant configuration [27]. Additionally, some arms are designed intrinsically compliant as in [51, 103]. On the other hand, compliant behavior can be assigned with a specific control law, as in several contributions [35,42,67,68,79,98,99].

Stability of the system during the interaction phase, and the reduction of contact forces obtained with passive and compliant design and control approaches, are the main concern of research studies on aerial manipulators interacting with the environment. Note that, with the mentioned approaches, no guarantees are provided on the maximum forces that are exchanged in the contact. Nevertheless, specific equipments to be installed or removed, or surfaces that are relatively fragile with respect to the aerial system, can be damaged during the manipulation phase. Taking into account the severity of a potential impact as a function of the dynamic parameters of the system, the aerial manipulator can be controlled in order to perform a motion which do not jeopardize the integrity of the contact object, or the manipulator body itself.

In this Chapter, a model-based damage index is proposed. It is based on the concept of dissipated kinetic energy in a potential inelastic impact. This quantity represents the fracture energy lost when a collision occurs, modeling both clamped and unclamped cases. The idea is to bound the proposed index under a certain threshold, such that, when the interaction occurs, the impact is not causing any damage. Notice that the considered damages include the effects of impacts of the system parts against the external environment, while the possible breakages caused by the propellers rotation is not taken into account. The proposed index depends on the robot reflected mass and velocity in the impact direction. It is expressed in analytical form suitable to be integrated in a constraint-based pre-collision control strategy. The exploited control architecture, based on a hierarchical quadratic programming approach analogous to the algorithm described in Chapter 3, allows to perform a given robot task while simultaneously bounding our damage assessment and minimizing the reflected mass in the direction of the impact.

The mentioned damage index has been conceived for safe human-robot interaction in the field of collaborative robotics. The goal is to preserve safety of humans via a control-based approach, which guarantees that the energy dissipated in potential impacts will not cause injuries to the human body. The approach can be applied to the aerial manipulator field, as a generic articulated manipulator is considered.

5.2 Damage Index

Since during an inelastic collision part of the kinetic energy of the system is dissipated in fracture or deformation energy, limiting the injuries on body tissues, or equipments damage, requires to bound the energy dissipated in an inelastic impact. To this end, it is here proposed to take into account the amount of dissipated energy in a potential inelastic impact in order to assess the injury or damage severity in a collision between a robot and the human or external environment. This concept is naturally derived from the Charpy impact test used to measure the amount of energy absorbed by a material during fracture and it is supported by the work of Povse et al. [91], where energy density exchanged in an impact has been experimentally related to the injuries caused to human bodies. The energy dissipated in an inelastic impact for the case of blunt collisions depends on the dynamic and kinematic properties of the robot and can model both clamped and unclamped collisions. Furthermore, this damage index will be exploited in order to devise a pre-impact safety control strategy by means of a constraint-based optimization algorithm, based on [109, 110]. In the following Section, the analytical formulation of the proposed damage index will be derived, while a constraint-based pre-collision control strategy for redundant robots is presented in Sec. 5.4.

5.3 Damage Assessment

This Section provides the formal computation of the amount of dissipated energy in a potential blunt inelastic impact between a robotic manipulator and an object[1] at rest. The computation is general for the clamped and unclamped case. The computed dissipated energy corresponds to the fracture and deformation energy that causes damages in a potential impact, thus a quantity that should be minimized. Consider an inelastic, blunt impact of a point of coordinates $\boldsymbol{x}_r \in \mathbb{R}^3$ on a manipulator with joint coordinates

[1]The generic “object” represents an human when the approach is applied in the context of physical human-robot interaction, while it represent a generic element of the environment that should not be damaged when the method is applied in the aerial manipulation field.

$\boldsymbol{q} \in \mathbb{R}^n$ against a point of coordinates $\boldsymbol{x}_e \in \mathbb{R}^3$ on a object, both expressed in the world frame. The value of the robot velocity in the direction of the object is given by:

$$\dot{x}_{\boldsymbol{n}} = \boldsymbol{n}^T J\left(\boldsymbol{q}\right)\, \dot{\boldsymbol{q}} = J_{\boldsymbol{n}}\left(\boldsymbol{q}\right)\, \dot{\boldsymbol{q}}$$

where $J \in \mathbb{R}^{3\times n}$ is the Jacobian of the linear velocity of the point, $\boldsymbol{n} \in \mathbb{R}^3$ is the direction of the segment from $\boldsymbol{x}_r$ to $\boldsymbol{x}_e$ and $J_n \in \mathbb{R}^{1\times n}$ is the jacobian of the velocity of the point $\boldsymbol{x}_r$ along the direction $\boldsymbol{n}$.
In an inelastic impact the momentum of the total system is conserved while part of the kinetic energy is dissipated. In addition, after a perfectly inelastic impact the velocities of the two bodies in the contact point are equal.
In order to calculate the dissipated energy, a suitable change of variables, linear with respect to $\dot{\boldsymbol{q}}$, is performed:

$$\begin{aligned}\boldsymbol{\beta} &= A\left(\boldsymbol{q}\right)\dot{\boldsymbol{q}}\\ A\left(\boldsymbol{q}\right) &= \begin{bmatrix} J_{\boldsymbol{n}}\left(\boldsymbol{q}\right) \\ \text{Ker}\left(J_{\boldsymbol{n}}\left(\boldsymbol{q}\right)\right)^T \end{bmatrix}\end{aligned}$$

where $\text{Ker}\left(J_{\boldsymbol{n}}\right)$ is a basis of the null space of $J_{\boldsymbol{n}}$, yielding:

$$\boldsymbol{\beta} = \begin{bmatrix} \dot{x}_{\boldsymbol{n}} \\ \boldsymbol{\alpha} \end{bmatrix}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{n-1}$ are the velocities of the null space of $J_{\boldsymbol{n}}$.
Therefore, the momentum $\boldsymbol{P}_{\boldsymbol{\beta}}$ and the kinetic energy $\boldsymbol{T}_{\boldsymbol{\beta}}$ of a robotic manipulator expressed in the new coordinates $\boldsymbol{\beta}$ result as follows:

$$\begin{aligned}\boldsymbol{P}_{\boldsymbol{\beta}} &= B_{\boldsymbol{\beta}}\left(\boldsymbol{q}\right)\boldsymbol{\beta}\\ T_{\boldsymbol{\beta}} &= \frac{1}{2}\boldsymbol{\beta}^T B_{\boldsymbol{\beta}}\left(\boldsymbol{q}\right)\boldsymbol{\beta}\end{aligned}$$

where $B_{\boldsymbol{\beta}}$ is inertia matrix calculated in the new variables:

$$B_{\boldsymbol{\beta}}\left(\boldsymbol{q}\right) = A\left(\boldsymbol{q}\right)^{-T} B\left(\boldsymbol{q}\right) A^{-1}\left(\boldsymbol{q}\right)$$

and $B\left(\boldsymbol{q}\right)$ the inertia matrix in the joint coordinates. From now on the dependence of the introduced matrices on the joint coordinates will be omitted.
The conservation of the momentum allows to calculate the velocities of the system composed of the robot and the object after the impact in the new coordinates.

$$\begin{aligned}&\boldsymbol{P}_0 = \boldsymbol{P}_1\\ &B_\beta\boldsymbol{\beta}_0 = B_\beta \begin{bmatrix} \dot{x}_{\boldsymbol{n},1} \\ \boldsymbol{\alpha}_1 \end{bmatrix} + \begin{bmatrix} m_e & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \dot{x}_{\boldsymbol{n},1} \\ 0 \end{bmatrix} = B_{\beta,re}\, \boldsymbol{\beta}_1 \qquad (5.1)\end{aligned}$$

where $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$ are the momentum of the whole system respectively before and after the impact, $\boldsymbol{\beta}_0$ and $\boldsymbol{\beta}_1$ are respectively the velocities of the robot before the impact and the velocities of the total system after the impact, m_e is an equivalent mass of the object in the point $\boldsymbol{x}_e$ in the direction $\boldsymbol{n}$ (the choice of m_e will be briefly discussed in Sec. 5.4) and $B_{\beta,re}$ is the inertia matrix of the total system after the impact as follows

$$B_{\beta,re} = B_\beta + \begin{bmatrix} m_e & 0 \\ 0 & 0 \end{bmatrix}$$

It has been assumed that the object is motionless before the impact occurs. The energy dissipated in the impact is equal to the difference between the kinetic energy before and after the impact:

$$\Delta T = \frac{1}{2}\boldsymbol{\beta}_0^T B_\beta \boldsymbol{\beta}_0 - \frac{1}{2}\boldsymbol{\beta}_1^T B_{\beta,re} \boldsymbol{\beta}_1 \tag{5.2}$$

Plugging (5.1) in (5.2), it is possible to obtain a very simple expression of the dissipated energy:

$$\Delta T = \frac{1}{2}\frac{m_e}{1+\frac{m_e}{m_{\boldsymbol{n}}}}\dot{x}_{\boldsymbol{n}}^2 \tag{5.3}$$

where $m_{\boldsymbol{n}}$ is the reflected mass of the robot in the point $\boldsymbol{x}_r$ along the direction $\boldsymbol{n}$, calculated in the following way $m_{\boldsymbol{n}} = \left(J_{\boldsymbol{n}} B^{-1} J_{\boldsymbol{n}}^T\right)^{-1}$ The case of impact with clamping can be derived from (5.3) when m_e is infinite $\Delta T_c = \frac{1}{2} m_{\boldsymbol{n}} \dot{x}_{\boldsymbol{n}}^2$. The value ΔT, in both cases with and without clamping, represents the energy dissipated in the impact and converted in fracture or deformation energy. It is worth pointing out that ΔT depends on the kinematic properties of the robot, thus it can be monitored during motion.
Finally, notice that model based injury indices used in literature are linear with reflected mass and robot velocity, thus they can have an infinite value in presence of singularity points of the reflected mass. With this respect, we can prove that in our approach the absolute value of ΔT, in presence of singularity points (with or without clamping), is upper limited by $\lambda_{B,max}||\dot{\boldsymbol{q}}||^2$, where $\lambda_{B,max}$ is the largest eigenvalue of B. Note that, while in general the reflected mass can assume infinite values for a serial manipulator fixed to the ground, the case of aerial manipulators is different. In fact, as the aircraft is a floating base, the jacobian J is always full rank, and the reflected mass can not assumed infinite values.

5.4 Pre-Impact Control Strategy

This Section provides a pre-impact control strategy for redundant manipulators, based on the concept of dissipated energy during a potential inelastic impact between an object and a robotic manipulator, by means of constraint-based and prioritized motion planning algorithm.
Following the approach in [109, 110], the proposed control architecture is depicted in Fig. 5.1.

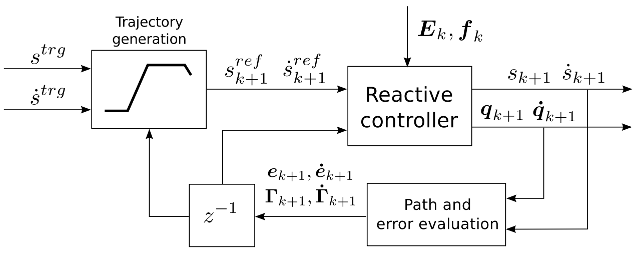

Trajectory generation
s^{trg}
$\dot{s}^{trg}$
s^{ref}_{k+1} $\dot{s}^{ref}_{k+1}$
$\boldsymbol{E}_k, \boldsymbol{f}_k$
Reactive controller
s_{k+1} $\dot{s}_{k+1}$
$\boldsymbol{q}_{k+1}$ $\dot{\boldsymbol{q}}_{k+1}$
$\boldsymbol{e}_{k+1}, \dot{\boldsymbol{e}}_{k+1}$
$\boldsymbol{\Gamma}_{k+1}, \dot{\boldsymbol{\Gamma}}_{k+1}$
z^{-1}
Path and error evaluation

Figure 5.1: *Block diagram of the proposed algorithm for path-following tasks*

Herein, a real-time trajectory generator is combined with a constraint-based optimization algorithm that feeds a lower-level position/velocity controller. Letting s be the natural coordinate of a prescribed end effector trajectory and $\ddot{\boldsymbol{z}}_k = \begin{bmatrix} \ddot{s}_k & \ddot{\boldsymbol{q}}_k \end{bmatrix}^T$ the vector of control variables to be computed and optimized at time instant k, the reactive control algorithm implements a hierarchical Quadratic Programming (hQP) solver. Each layer i of the optimization algorithm can be written as follows:

$$\min_{\ddot{\boldsymbol{z}}_k} \frac{1}{2} \ddot{\boldsymbol{z}}_k^T \boldsymbol{H}_k^{(i)} \ddot{\boldsymbol{z}}_k + \ddot{\boldsymbol{z}}_k^T \boldsymbol{g}_k^{(i)} \tag{5.4a}$$

$$\text{s.t. } j = 1, \dots, i : \; \boldsymbol{E}_k^{(j)} \ddot{\boldsymbol{z}}_k \leq \boldsymbol{f}_k^{(j)} \tag{5.4b}$$

Notice that the reactive controller here proposed has a similar structure to the trajectory generation algorithm presented in Sec. 3.2.1, whit a main difference. In fact, a hierarchical solver is used, which would require computational capabilities too large to be installed on an aerial manipulator but perfectly compatible with equipments employed in industrial robotics.

With respect to the described control architecture, the general idea behind our approach is to limit the dissipated energy to be under a certain threshold during a prescribed path-following task, while simultaneously exploit the kinematic redundancy to decrease the reflected mass of several points on the robot arm along the directions of potential impacts.

These goals can be achieved through a proper selection of constraints and cost-functions as detailed in the following.

5.4.1 Constraining the dissipated energy

We here consider a generic point i on the robot kinematic chain, a point j on the object, and the unit vector $\boldsymbol{n}_{ij}$ from i to j . From (5.3), the dissipated energy along the direction of a possible impact, is given by:

$$\Delta T_{ij} = \frac{1}{2}\,\frac{m_{e,ij}}{1+\frac{m_{e,ij}}{m_{r,ij}}}\,\dot{x}_{ij}^2 \tag{5.5}$$

where $m_{r,ij}$ and $m_{e,ij}$ are the reflected masses along the direction of potential impact of the point i on the robot and of the point j on the object, respectively, while $\dot{x}_{ij}$ is the Cartesian velocity of point i along $\boldsymbol{n}_{ij}$.
Notice that: $m_{r,ij}^{-1} = J_{ij}B^{-1}J_{ij}^T$ and $\dot{x}_{ij} = J_{ij}\dot{\boldsymbol{q}}$, where: $J_{ij} = \boldsymbol{n}_{ij}^T J_i$, being J_i the Jacobian of point i, while B is the robot inertia matrix. In addition, $m_{e,ij}$ can be computed as the reflected mass of a robotic manipulator equivalent to the object, in the case it presents an articulated structure. However, a conservative choice from a control perspective would be to use the overall object mass, for the unclamped case, or infinite mass for the clamped case. Considering N_i points on the robot arm and N_j points on the object body, our approach consists in limiting the dissipated energy for each pair of points i,j, computed in (5.5), to be under a certain threshold $\overline{\Delta T}_j$, whose value could depend on the object part where the point j is located:

$$\Delta T_{ij}(\boldsymbol{q},\dot{\boldsymbol{q}}) \leq \overline{\Delta T}_j \tag{5.6}$$

In order to integrate this constraint within a first level optimization algorithm, i.e. path-following task, we need to make explicit the dependence on $\ddot{\boldsymbol{q}}$, see (5.4b).
To this end, one can apply the Gronwall's lemma [15] to (5.6), as described in Chapter 3, yielding:

$$\frac{d\Delta T_{ij}}{dt} \leq -\alpha\left(\Delta T_{ij} - \overline{\Delta T}_j\right),\ \alpha > 0 \tag{5.7}$$

that, based on lemma, guarantees exponential convergence to (5.6), with a rate depending on α.
From (5.5), we obtain

$$\frac{d\Delta T_{ij}}{dt} = -\frac{1}{2}\,c_{ij}^2\,\frac{d\,m_{r,ij}^{-1}}{dt}\,\dot{x}_{ij}^2 + c_{ij}\,\dot{x}_{ij}\,\ddot{x}_{ij}$$

where

$$c_{ij} = \frac{m_{e,ij}}{1 + \frac{m_{e,ij}}{m_{r,ij}}}$$

and

$$\begin{aligned}\frac{d\, m_{r,ij}^{-1}}{dt} &= \nabla m_{r,ij}^{-1}\, \dot{\boldsymbol{q}} \\ &= \sum_{k=1}^{n} \left(J_{ij} \frac{\partial B^{-1}}{\partial q_k} J_{ij}^T + 2 J_{ij} B^{-1} \frac{\partial J_{ij}^T}{\partial q_k} \right) \dot{q}_k\end{aligned}$$

Therefore, all the terms in (5.7) can be analytically determined and 5.7 can be equivalently expressed as the following inequality: $E(\boldsymbol{q}, \dot{\boldsymbol{q}})\ \ddot{\boldsymbol{q}} \leq f(\boldsymbol{q}, \dot{\boldsymbol{q}})$, where $E(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is given by:

$$E(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{m_{e,ij}}{1 + \frac{m_{e,ij}}{m_{r,ij}}}\ \dot{\boldsymbol{q}}^T J_{ij}^T\ J_{ij}$$

while the expression of $f(\boldsymbol{q}, \dot{\boldsymbol{q}})$ has been omitted for brevity.

5.4.2 Minimization of the reflected mass

In order to enforce the constraint on the dissipated energy (5.6), the kinematic redundancy of the manipulator can be exploited to increase $m_{r,ij}^{-1}$, or alternatively its time derivative, i.e. to decrease the robot reflected mass. To this end, we introduce a second optimization stage, acting only in the null space of the task.
Considering N_i points on the robot arm and N_j points on the object body, a candidate metric function can be formulated as follows

$$L(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} w_{ij}\ \frac{d\, m_{r,ij}^{-1}}{dt} \tag{5.8}$$

In the above equation, w_{ij} is an arbitrary weighting function

$$w_{ij} = f\left(\dot{x}_{ij}\right) \frac{\dot{x}_{ij}^2\ m_{r,ij}}{d_{ij}^2 + \varepsilon}, \quad \varepsilon > 0$$

where ε is an arbitrarily small positive constant, $f\left(\dot{x}_{ij}\right)$ is defined as follows:

$$f\left(\dot{x}_{ij}\right) = \begin{cases} 1, & \dot{x}_{ij} \geq 0 \\ 0, & \dot{x}_{ij} < 0 \end{cases} \tag{5.9}$$

and d_{ij} is the Cartesian distance between points i and j.
Notice that, we need to explicitly express the dependence of the chosen metric function on the control variable $\ddot{\boldsymbol{q}}$, see (5.4a). Therefore, applying time derivative to (5.8), we obtain the new metric function $L'(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \frac{dL(\boldsymbol{q},\dot{\boldsymbol{q}})}{dt}$:

$$L'(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \left(\frac{dw_{ij}}{dt} \frac{d\, m_{r,ij}^{-1}}{dt} + w_{ij} \frac{d^2\, m_{r,ij}^{-1}}{dt^2} \right) \tag{5.10}$$

All the terms in (5.10) can be analytically determined, leading to the simple form:

$$L'(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) = \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \left(g_{ij}(\boldsymbol{q}, \dot{\boldsymbol{q}})\; \ddot{\boldsymbol{q}} + b_{ij}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \right) \tag{5.11}$$

where $g_{ij}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is given by:

$$g_{ij}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = w_{ij} \nabla m_{r,ij}^{-1} + 2 f\left(\dot{x}_{ij}\right) \frac{\dot{x}_{ij} m_{r,ij} \frac{d\, m_{r,ij}^{-1}}{dt}}{d_{ij}^2 + \varepsilon} J_{ij}$$

while the expression of $b_{ij}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ has been omitted for brevity.

5.5 Validation

A validation of the damage assessment and of the effectiveness of the control algorithm has been obtained through simulations and experiments. A series of simulations has been carried out to show the applicability of the approach with the aim of preserving the integrity of an aerial manipulator impacting with an external surface. An experimental campaign, performed with a collaborative robot, demonstrates the effectiveness of the method on a real scenario.

5.5.1 Simulated Impact of an Aerial Manipulator

The performed simulations consider the impact of an aerial manipulator with an external surface, modeled as a spring with high stiffness. The simulation platform has been presented in Section 2.5. The model of the aerial manipulator has been assembled in Matlab-Simulink environment, and it consists of a quadrotor equipped with a 4 DoFs robotic arm. The general structure of the modeled system is represented in Fig.5.2. In the present simulation, the optimization algorithm that is employed to generate the joint

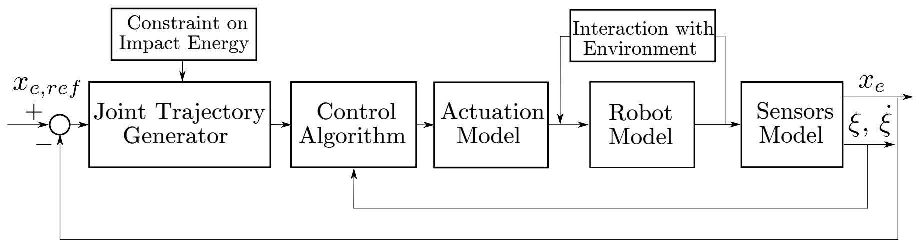


Figure 5.2: *Block diagram of the simulation scheme.*

trajectory can bound the index in (5.3), which represents the dissipated energy along the direction of a potential impact, under a certain threshold, according to the method presented in Section 5.4.1.

Designed Experiment and Results

The end effector reference trajectory consists in a linear path along the forward direction (x), with a trapezoidal velocity profile. A surface, whose normal is directed along the tracked path, is placed at 0.4 m from the aerial manipulator starting point. The surface is modeled as a spring with stiffness equal to $1000\,{}^{\mathrm{N}}\!/_{\mathrm{m}}$. The impact occurs between the surface and the robot end effector.

The path is tracked with two different strategies in subsequent experiments. In the first experiment, no constraints are imposed on the ΔT, while in the second one the index is bounded under a given threshold. The couple of points needed to define the index are the robot end effector and the point at the intersection between the surface and the path line.

The trajectory performed by the robot end effector in x direction, in the case where the index is not constrained, is represented in Fig. 5.3 (red solid line). The reference trajectory (dashed red line) can be tracked until the collision occurs. The contact force is represented with a blue line in the same figure, and its maximum value is 2.2 N. Notice that, under some assumptions, the following relation between the index ΔT and the maximum contact force in a clamped impact can be obtained: $F_M \approx \sqrt{K_{env}\Delta T}$, where K_{env} is the equivalent environment stiffness. This relation is further discussed in Section C.1.2. The maximum contact force F_M, computed with the approximated formula, is 1.5 N, while the real value is 2.2 N.

In Fig. 5.4, the time behavior of the reflected mass of the robot (solid blue line), its end effector velocity (dashed blue line) and ΔT index (solid red line) are represented. Notice that the value of the reflected mass changes along time due to the motion of robot joints. The increase of the reflected

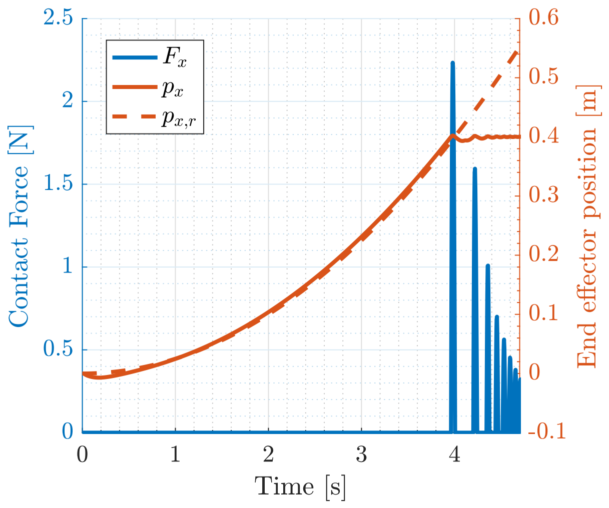


Figure 5.3: *Impact of the aerial manipulator against a rigid surface, when the ΔT index is not bounded. End effector trajectory and reference trajectory are represented with solid and dashed red lines, respectively. The contact force is reported with a blue line.*

mass obviously results in a higher impact energy ΔT. The red dashed line represents the threshold of impact energy that will be employed as a constraint in the second scenario.

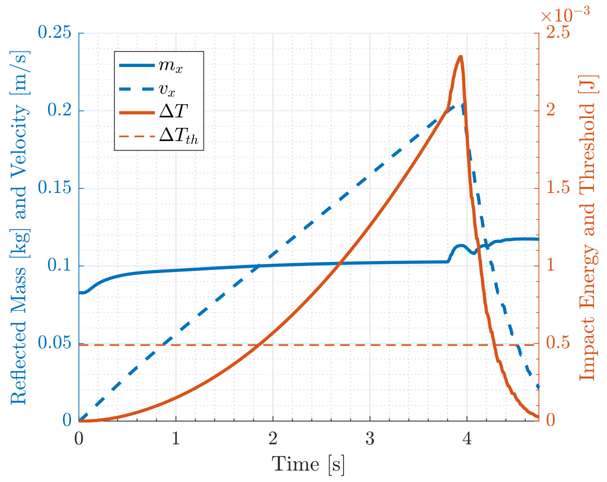


Figure 5.4: *Impact of the aerial manipulator against a rigid surface, when the ΔT index is not constrained. The reflected mass (blue solid line), the end effector velocity (blue dashed line), and the ΔT index (red solid line) are represented. Red dashed line represents the threshold of ΔT index that is employed in the constrained experiment.*

The second simulation is performed applying the constraint on the maximum value of the index ΔT. The threshold is set to $4.9\,10^{-4}$ J. In Fig. 5.5, the end effector trajectory and contact force are represented. When the

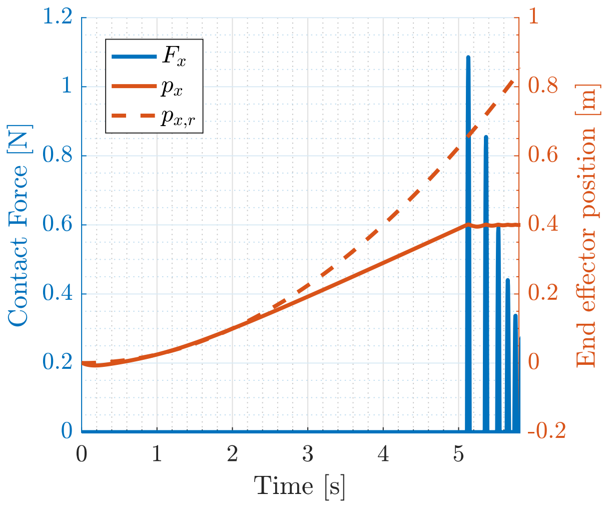


Figure 5.5: *Impact of the aerial manipulator against a rigid surface, when the ΔT index is bounded under a threshold. End effector trajectory and reference trajectory are represented with solid and dashed red lines, respectively. The contact force is reported with a blue line.*

constraint is active, the reference trajectory can not be correctly tracked. This happens after approximately 2.4 s. In this case, the resulting impact force is correctly lower than the previous one, in fact a maximum value of 1.1 N is obtained.

In Fig. 5.6, the reflected mass, the end effector velocity and the ΔT index are reported. The trajectory generation algorithm computes quadrotor and robot reference velocities, such that the energy impact threshold is not exceeded. When the impact occurs, robot configuration changes due to the interaction dynamics, and the increase of the reflected mass causes the index to violate the constraint. Nevertheless, no numerical problems can arise, since actual robot configuration is not employed in the trajectory generation algorithm.

5.5.2 Experimental Results on a Collaborative Robot

Experiments have been performed on a collaborative robot, since the algorithm was first conceived for the goal of safe human-robot interaction.

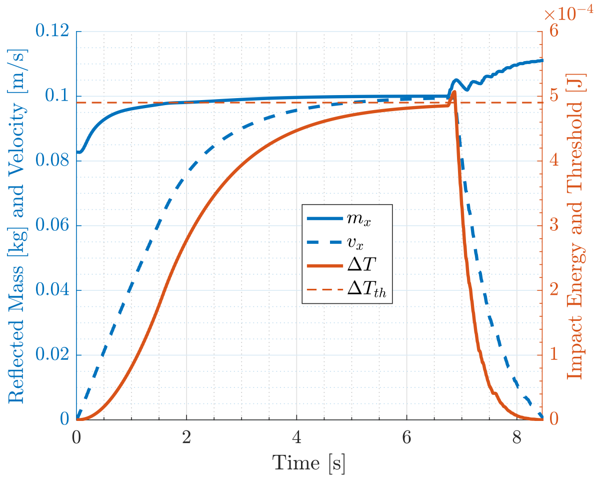


Figure 5.6: *Impact of the aerial manipulator against a rigid surface, when the ΔT index is bounded under a threshold ΔT_{th}. The reflected mass (blue solid line), the end effector velocity (blue dashed line), and the ΔT behavior (red solid line) are represented.*

The experimental campaign is detailed in Appendix C. However, the experimental activities allow to draw the following conclusions, relevant to both field of human-robot interaction and aerial manipulation:

i) The index ΔT correctly describes the fracture energy in an inelastic impact.

ii) The pre-impact control strategy implemented succeeds in bounding the ΔT under a desired threshold.

iii) The damage assessment has direct functional dependence with the force exchanged in an elastic impact with stiff environment.

iv) The control strategy succeeds in minimizing the reflected mass of the end effector in the direction of a potential impact, allowing increased task efficiency.

5.6 Discussion

The proposed model-based damage assessment has been conceived in the field of physical human-robot interaction. In this Chapter, the motivations of the potential application of the approach to the aerial manipulator control are described. In particular the approach can be useful when the integrity of the manipulated object can be undermined by the aerial robot itself. Notice that damages caused by propellers are not considered in the approach.

The method, as it is presented in this Chapter, can generate a safe trajectory for aerial vehicle, reactively modifying the motion in order to bound the damage index and the severity of the impact. Since the actual measure of the joints configuration are not passed in feedback to the method, oscillations of the aerial platform due to external disturbances, and consequent compensation of the arm, are not considered in the approach. However, note that the minimization of the reflected mass in the direction of the potential impact is beneficial also for the case of collision generated by external disturbances on the aircraft, because the potential damage is reduced.

Then, how damages caused by external disturbances can be integrated in the proposed approach? This issue can be addressed in two ways. First, the actual joint measures can be employed by the algorithm, in order to reactively compute control actions. Obviously, the structure of the inner velocity control algorithm has to be modified, and the stability of the overall system has to be analyzed. A different possible strategy consists in the application of a trajectory generation approach that is robust to state noise and input disturbances as in [111]. The second strategy is preferable because it guarantees stability and the fulfillment of prescribed constraints under certain conditions, and minimal modifications to the trajectory generation method have to be applied. As an obvious remark, this approach can exhibit a decrease in the performance in order to preserve robustness.

CHAPTER 6

Conclusions

The present dissertation dealt with the broad research area of aerial manipulation, which aims to extend the workspace of robotic manipulators to potentially unlimited 3D space employing a flying platform. Although promising results have been presented in the literature, aerial manipulation is still a recent research field which is characterized by low maturity and a number of open research challenges.

The present dissertation, in particular, provided contributions in the topics of tracking control, trajectory generation, state estimation and control of interaction. A system composed of a multi-rotor aircraft equipped with a robotic arm underneath has been considered.

A challenging issue for the control of aerial manipulators is represented by the dynamic coupling between the flying platform and the robotic arm. The employed multi-rotors aircraft are usually under-actuated, thus disturbance forces on specific directions cannot be canceled. While stability of the whole system can be guaranteed with state of the art control strategies, tracking performance is affected by the interaction of the two subsystems. In Chapter 2, a decoupling strategy has been proposed, which is based on a suitable change of variables that implicitly cancels the interaction between the flying base attitude and the robotic arm motion. An inverse dynamics control has been conceived to minimize the disturbances on under-actuated directions. A detailed simulation campaign has been performed, in order to test the control strategy in various operating points and modeling inaccurate state estimation. The control has been proven to be stable, and the main result achieved consists in the improvement of tracking performance with respect to state of the art approaches. Some limitations of the approach can arise in the technological implementation of the algorithm. The tracking performance improvement can be limited by inaccurate sensors information, model parameters errors or communication delays. Thus, the advantages provided by the proposed control strategy should be mainly exploited by systems provided with high quality equipments in applications that require accurate tracking control. Further research activities can be addressed to overcome this limitation and increase the robustness of the approach.

A second research line investigated in this thesis concerns the kinematic redundancy resolution for aerial manipulators. The dexterity of the platform and the high number of degrees of freedom of the whole system can be exploited to accomplish multiple tasks. In the literature, a number of trajectory generation approaches for aerial manipulators, based on null-space projection, have been proposed. Those strategies can hardly deal with bounds and inequality constraints. Nevertheless, aerial manipulators stability and integrity can be undermined when particular subsets of state configuration space are assumed by the system. First optimization-based approaches, dealing with inequality constraints, have been proposed in literature, however just simulation results or off line implementation have been shown. In Chapter 3, two different approaches based on optimization techniques to generate a feasible trajectory for the degrees of freedom of the aerial manipulator have been proposed.

A quadratic programming approach has been applied in the first strategy. A second order local approximation of the system is employed, in order to solve the optimization problem with respect to variables acceleration. Several tasks and constraints have been integrated in the algorithm and a weighing strategy has been employed in order to establish a soft hierarchy between tasks, and dynamically change the relative relevance of the objectives during different phases of the mission. The method has been implemented in a real platform and the real-time trajectory generation runs at 50 Hz. The performed experimental activities showed that the system can accomplish the required tasks while satisfying constraints, and dynamically change behavior between different mission phases with simple scaling of weights. In particular, an obstacle avoidance trajectory is performed, while tasks oriented to preserve stability and facilitate manipulation activities have been accomplished. Notice that, since the problem is solved in acceleration form, smooth trajectory variations are obtained. This is particularly useful for two reasons. First, the transition between different phases do not cause abrupt changes in the reference signals. Second, note that, given the quadrotor structure, roll and pitch angles are proportional to the second derivative of quadrotor position, thus smoother quadrotor trajectory are beneficial to reduce the attitude actuation effort.

The second trajectory generation algorithm is based on a non linear model predictive control strategy. In this case, the optimization problem has been formulated and solved with respect to variable velocities, in order to reduce the total number of states and consequently the computational burden of the method. Similar to the first approach, different tasks and constraints have been integrated in the algorithm. The main advantage of the method is the longer horizon of the state evolution considered in the optimization, which can lead to a better motion strategy. On the other hand, the higher computational burden of the method limits the practical implementation to lower loop rates. In fact, in the performed experimental activities, the algorithm could run on line with a rate of 2 Hz. Nevertheless, it has been shown that the method can effectively generate a trajectory for an aerial manipulator.

Both the proposed methods are the first on line implementations of optimization based trajectory generation algorithms on aerial manipulator systems. Thanks to this approach, dangerous state configurations can be avoided, and optimal motion policies can be adopted. From the author perspective, the easiness of integrating inequality bounds and constraints in the redundancy resolution problem is the main improvement with respect to the state of the art algorithms employed in the aerial manipulator field.

In the present dissertation, the state estimation of aerial manipulator systems has been considered under the design and implementation point of view. State of the art algorithms and different configurations of sensing systems have been analyzed and applied. The specific theoretical contribution provided in this field concerns the calibration and performance evaluation of a camera, mounted on the flying platform, by means of an industrial manipulator. The method is particularly useful in industry and in the majority of the research labs, where expensive sensors as motion capture systems are not available, thus it is difficult to evaluate the performance of a vision system.

The last main contribution provided in the thesis concerns the interaction between an aerial manipulator and the external environment. The majority of state of the art approaches in the field of interaction control for aerial manipulation have the goal of providing a compliant behavior to the system when in contact whit the external environment. With these control strategies, no guarantees are provided on the maximum forces that are exchanged in the contact. In Chapter 5, a model-based damage index has been formulated, which depends on the robot reflected mass and velocity in the impact direction. It represents the dissipated kinetic energy in a potential inelastic impact. Thus, the damage index has been integrated in a constraint-based control strategy, in order to limit the severity of impacts with external environment. While the mentioned damage index has been conceived for safe human-robot interaction in the field of collaborative robotics, it can be applied to aerial manipulators and it is easily integrable in the trajectory generation approach proposed in Chapter 3. In addition, in the control approach the minimization of the reflected mass has been integrated, which allows to increase the motion velocity while preserving the same value of damage index.

The possibility to define a maximum possible fracture energy during manipulation activities, allows to perform tasks while guaranteeing the integrity of the components of the system, which can be useful in operations with relatively fragile equipment. Notice that materials and shape of the tools or manipulated objects have not been considered. However, this represents an application dependent issue, which can be addressed with different threshold values for the damage index.

The present dissertation provided contributions to extend the autonomy and improve the performance of aerial manipulator systems. In the next years, the efforts of researchers will be addressed to enable the adoption of such systems in industry and real world activities. To this extent, the robustness of the perception and control algorithms will be a key enabling factor, together with the achievement of effective interfaces with humans, in order to perform semi-autonomous operations, and collaboration with other aerial manipulators, to provide the required payload capability.

Appendices

APPENDIX *A*

Design and Characterization of an Aerial Manipulator

In the present Chapter, the design of an aerial manipulator will be presented. In particular, special attention will be paid to the integration and synergy between the different subsystems. In the second part of the Chapter, the flying base and the robotic arm will be characterized, in order to obtain the information required from a control perspective. In particular, the inertial and actuation parameters of the two subsystems will be estimated.

A.1 Design of an Aerial Manipulator

The design of a complete aerial manipulator has to be carried out considering the requirements and constraints of all the subsystems composing it, and the possible conflicts between different requirements. As an obvious example, the computational unit cannot be selected without considering as critical characteristics its encumbrance and weight, which are related to the choice of the flying base. Flying base payload and system weights are the most critical constraints to be considered in the design. High payload quadrotors can sustain more complex and better performing sensors and servos actuators, as they are usually heavier. Nevertheless, the higher is the payload, the bigger is the quadrotor, limiting the employment of the system to specific environments and applications. The powering of the quadrotor, and battery capacity, is also a factor linked to the weight of the system. Thus, the weight of every component added on board of the system is a critical parameter.

The computational units to be integrated in the system should be dimensioned in order to provide the required computational capacity, while presenting the needed ports to communicate with all the elements of the system. In fact, several different sensors and actuators should be managed, at different frequency. Moreover, off-the-self components should provide open access to basic functionalities, in order to customize them and integrate the proposed algorithms. For instance, for some commercial quadrotors it is not possible to modify the internal control algorithms.

In this Chapter, a design solution for an aerial manipulator is discussed.

A.1.1 System Design

In this Section the complete system will be described. In particular, the first part will list the hardware components of the system, then the software architecture and communication will be described, and finally the strategy adopted for the overall power supply and the set up of the test bed will be addressed.

Components

In the following, a detailed description of the mechanical structure and the hardware components of the aerial manipulator will be provided. In Fig. A.1 the complete system, developed within this theses at the MERLIN Lab of the Department of Electronics, Information, and Bioengineering of Politecnico di Milano [7], is shown, whose structure is composed of an octacopter flying base and a robotic arm, and it is equipped with the needed computational units, battery and sensors to perform an autonomous flight and interaction task.

Flying Base The flying base of the aerial manipulator consists in a octacopter X8+ from 3DR, shown in Fig. A.2(a). Propellers are coaxial, then model and control for the octacopter are substantially equal to a quadrotor system. Coaxial propellers allows greater forces and torques with limited system encumbrance. The main difference with a quadrotor is relative to the produced aerodynamic effects on coaxial blades, which decrease the

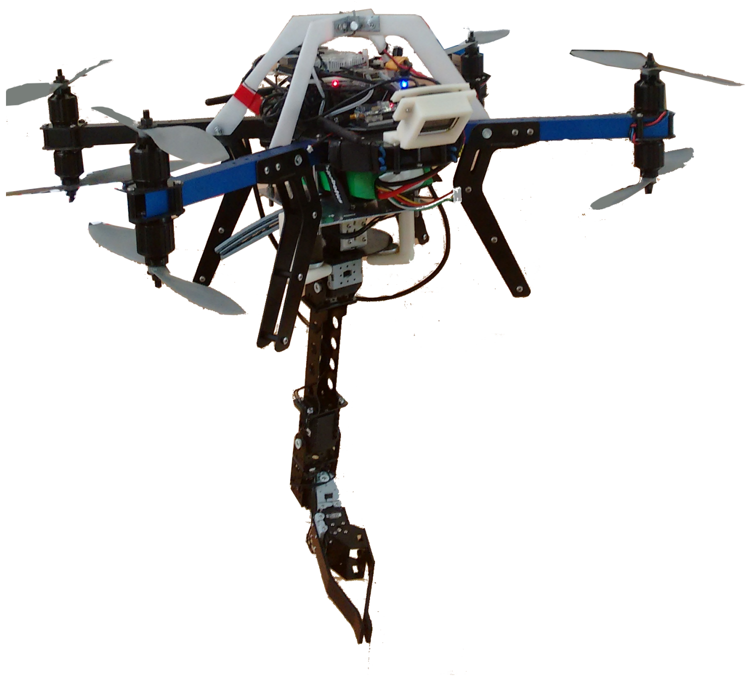
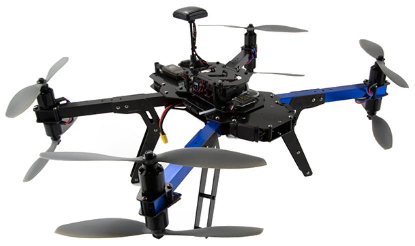


Figure A.1: *Complete aerial manipulator system.*

efficiency of the overall actuation. Thus, in the following, the flying base will be denoted as a generic quadrotor. The nominal payload of the drone is 1.0 kg, excluding battery, which weighs additional 0.8 kg. This is the most important parameter of the quadrotor, as it should sustain a robotic arm, together with additional sensors and computational units.

Micro-controller The drone is equipped with a micro-controller, specifically a Pixhawk Autopilot. The micro-controller has a 168 MHz CPU Cortex-M4F, with 264 kb RAM and 2 Mb of Flash memory. Notice that it integrates three axial gyroscopes and accelerometers, a magnetometer a barometer and a GPS. For the connection with external sensors and actuators, 14 Pulse Width Modulation (PWM) outputs, the I2C, UART and CAN buses, and generic programmable serial ports are available.

Localization Sensors In order to obtain the absolute localization of the quadrotor, the system has been equipped with two additional sensors: an optical flow module and a camera. The former is a PX4Flow module, which in-


(a) Flying Base

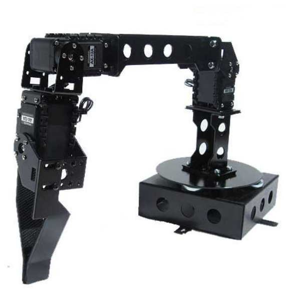

(b) Robot Arm

Figure A.2: *The image on the left represents the chosen flying base, a 3DR X8+ coaxial octacopter. On the right is it shown the CrustCrawler robotic arm, which has been assembled to be mounted underneath the flying base.*

i	a_{i-1} $[m]$	α_{i-1} [deg]	d_i $[m]$	θ_i
1	0	-90	0.1730	θ_1
2	0.1970	0	0	θ_2
3	0.0210	-90	0	θ_3
4	0.1780	0	0.0630	θ_4

Table A.1: *Denavit-Hartenberg Parameters of the Robotic Arm.*

tegrates a 752×480 pixels camera as an optical flow sensor and a sonar Maxbotix MB1043. It allows to estimate the translational velocity of the quadrotor. The camera is a Microsoft HD 5000 (1280 × 720 pixels) model. The design, sensors selection and estimation of quadrotor localization are reported in Chapter 4.

Robotic Arm The robotic arm was assembled using the elements of the CrustCrawler kit, reported in Fig. A.2(b). The arm has 4 degrees of freedom with a gripper at the end effector. The Denavit-Hartemberg parameters are reported in Tab. A.1, while frames are arranged as in Fig. A.3. The employed servos are Dynamixel AX-12A, which are characterized by a high gear ratio, equal to $1 : 254$, limited weight, 0.054 kg, and a relatively high maximum torque, 1.5 Nm. An encoder is provided in each servo, with a maximum range of $300°$ and a resolution of 1024 bit. Notice that the communication with the servos is performed on a serial TTL Half Duplex, on a single bus for all servos. The weight of the complete arm is 0.54 kg.

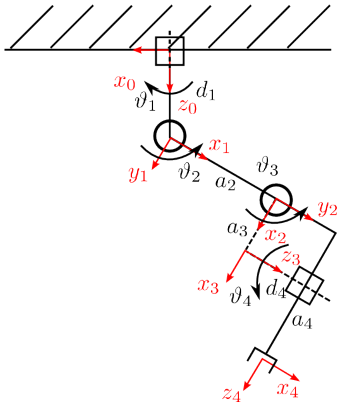


Figure A.3: *Denavit-Hartemberg Frames*

Computational Unit A computational unit is added on board the system, in order to perform the operations that require large computational burden. In fact, the additional board should perform the camera image analysis to obtain quadrotor localization measures, the on-line solution of an optimization problem for the joint trajectory generation, and the computation of the matrices needed to compute the dynamics and kinematics of the system. It has been chosen an Odroid XU3 characterized by two CPUs quadcore Cortex-A15 at 2 GHz and Cortex-A7 at 1.4 GHz, respectively, and 2Gb RAM at 933 MHz, with a Flash memory eMMC 5.0. The operating system Ubuntu 14.04 ARM version has been installed on the board.

Mechanical Supports The quadrotor, the arm, the additional sensors, and the computational units, have been assembled together to obtain the complete system. The assembly operation required the design of additional mechanical supports. The robotic arm has been fixed to the quadrotor by means of a 3D printed mechanical structure that can sustain the weight of the arm by means of bearings, reported in Fig. A.4. Camera and optical flow module are mounted on the front and the back, respectively, of the quadrotor, by means of 3D printed supports designed on purpose, reported in Fig. A.5.

Software Architecture and Communication

The software architecture has been designed in conjunction with the allocation of specific software applications on the physical computational units, and with the communication strategies between different units. Thus, the description of software architecture will be organized by grouping the different applications on the basis of the physical computational unit they are

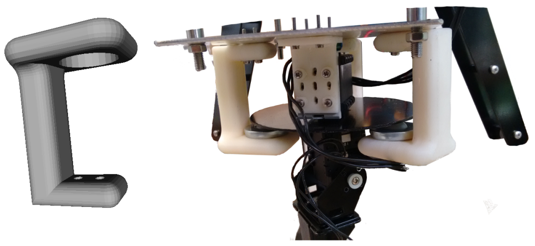
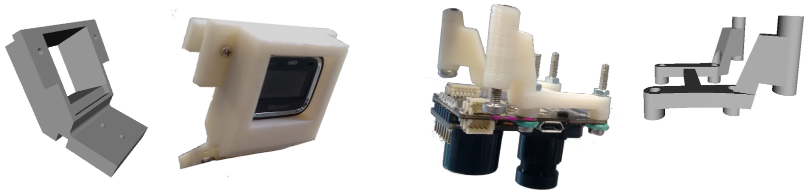


Figure A.4: *Support to sustain the robotic arm.*



(a) Camera support (b) Optical flow module support

Figure A.5: *3D Printed supports employed to mount the camera and the optical flow module on the quadrotor.*

programmed on. A description of the communication strategies will be presented. A scheme of the software architecture and communication protocols between units is reported in Fig. A.6.

Ground Station The ground station consists in a simple PC, where a ROS [94] framework is installed. A Wi-Fi connection is established with the computational unit on board of the aerial manipulator, and ROS packages are running on the ground station to send the end effector reference to the system and to receive specific data for log and visualization goals. A SSH connection is set with the computational unit, in order to start and stop programs running on it.

Computational Unit The computational unit, on board of the aerial manipulator, is provided with a ROS framework. It is physically connected with the camera by a serial USB, and with the micro-controller by two different buses, both serial USB. In the first channel, communication is performed with Micro Aerial Vehicle Communication Protocol (Mavlink), in order to exchange data between the two units, while the second channel is em-

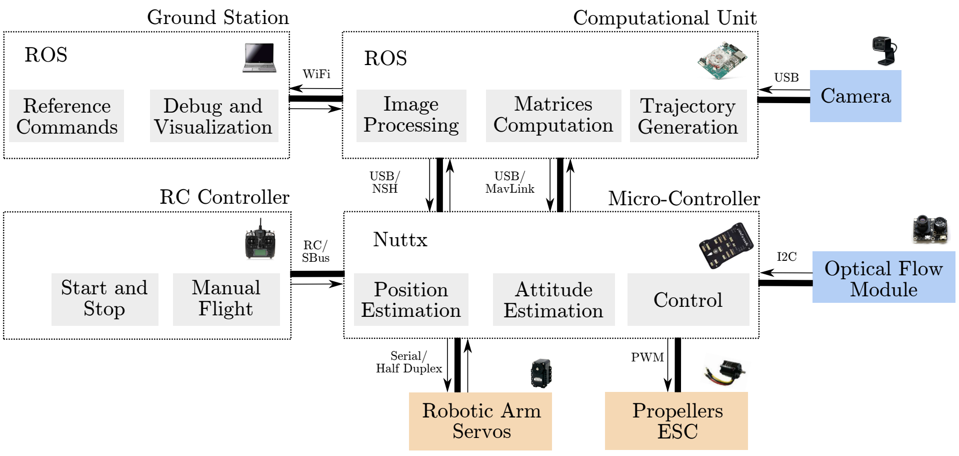


Figure A.6: *Software architecture and communication protocols between subsystems. The name of the unit is reported outside the corresponding box. Inside the box, the running operating system is reported, and gray boxes represent the most important tasks performed by the unit. Communication protocols and physical connections are near to the link icons. Arrows represent the direction of information transmission.*

ployed to get access to the micro-controller NSH[1] terminal, to perform debug and assign direct commands. On this unit, different ROS packages are running in parallel. First, image analysis and a simple odometry algorithm is performed to compute the quadrotor localization. The obtained absolute position of the quadrotor is sent to the micro-controller, while the current values of all the states of the aerial manipulator are received from the micro-controller itself. In addition, kinematics and dynamics matrices are computed and sent back to the micro-controller. Third, an optimization problem for trajectory generation is solved on line.

Micro-Controller The micro-controller is physically linked with the computational unit by USB serials, with the Electronic Speed Control (ESC) of the 8 propellers, and with the optical flow module by I2C bus. Then, a programmable serial port is used to communicate with robotic arm servos. In particular, the serial port has been programmed to perform the communication with the asynchronous Half-Duplex protocol with TTL logic levels. A logic level converter circuit, reported in Fig. A.7, was needed to match the $0-3.3$ V range of the micro-controller output with the $0-5$ V range servos. The native operating system of the micro-controller is a the real-time OS Nuttx for embedded applications. The PX4 Flight Stack [75] is a collection

[1]NSH stands for NuttX Shell, where NuttX is the real-time operating system running on the micro-controller.

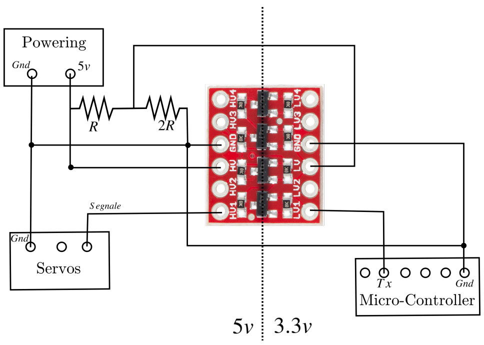


Figure A.7: *Logic level converter to establish the communication between robotic arm servos and micro-controller.*

of algorithms for autonomous drones, and it is the software infrastructure chosen to manage the low-level communication with sensors and actuators and to perform estimation and control. Such algorithms have been modified and extended in order to integrate the control approach presented in the work. Different parallel threads are running, and variables data sharing between threads is managed with uORB (micro Object Request Broker) application. The maximum update rate of the control loop is 250 Hz.

Powering

The powering of aerial manipulator is one of the most relevant technological issues. In fact, weights and battery capacity are complementary factors that determine the maximum flight time. Then, it is a critical requirement to have just one power source, in order to limit the total weight. However, notice that propellers ESC have to be powered at 16 V, arm servos at 12 V, while the micro-controller and the computational unit need a 5 V voltage. Thus, suitable DC-DC converters have been included in the primary circuit. The complete powering scheme is reported in Fig. A.8. During the experiments a LiPo battery 4s with a capacity of 10 Ah, or, alternatively, a programmable laboratory power supply EA-PSI 808-610 have been employed as power supply.

Test Bed

While the aerial manipulator was being developed and assembled, testing experiments have been performed to check the functionality of its subsystems. In particular, a test bed has been built to check the functioning of the quadrotor alone, and the quadrotor and the arm together, without jeopardizing safety. The test bed is reported in Fig. A.9(a), and it consists in a post

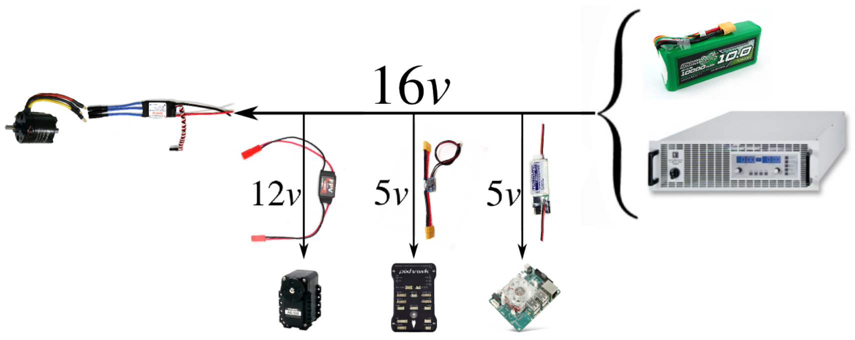


Figure A.8: *Powering circuit of the system.*

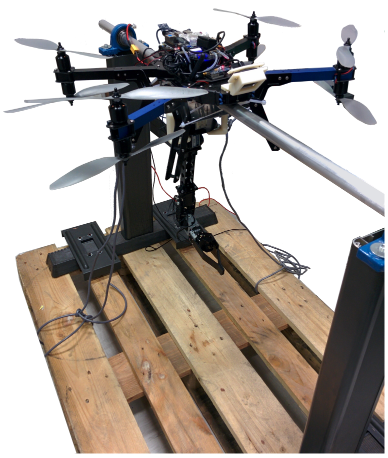
sustained by two bearings, such that the system keeps just one degree of freedom free. Then, flights experiments have been conducted indoor (see Fig. A.9(b)). Notice that a steel cable is fixed to the quadrotor, by means of a support shown in Fig. A.9(c), in order to prevent the quadrotor from crashing.

A.2 Estimation of Quadrotor and Arm Inertial Parameters

An estimation of the dynamic parameters of the system is needed in order to compute a model based control algorithm. In the following the center of mass and principal moments of inertia of the quadrotor will be estimated, while the dynamic parameters of the robotic arm will be obtained by a CAD model.

A.2.1 Quadrotor Inertia

The dynamical model of the quadrotor can be simplified assuming that the principal axes of inertia of the drone are parallel to the axes of its body reference frame. It is a reasonable assumption given the symmetry of the system structure. Thus, the inertia matrix can be considered diagonal. The parameters to be estimated are the mass of the system, the center of mass position with respect the geometrical center of the quadrotor and the three principal moments of inertia. The mass of the quadrotor, with all the additional sensors and without the battery, results equal to 2.37 kg. The estimation of CoM position and principal moments of inertia has been realized by means of an industrial robot, a 6 DoFs Comau SmartSix, equipped with a 6 axis force-torque sensor ATI gamma 130 - 10. The quadrotor is fixed


(a) Test Bed

(b) Flying environment

(c) Support for the quadrotor, used to sustain the system preventing crashes.

to the force sensor, which is mounted on the end effector of the industrial robot as in Fig. A.9. Performing a set of motion with the robot end effector and registering the corresponding inertial and gravitational forces and torques exchanged between the robotic arm and the quadrotor, the required parameters can be estimated. Forces $\boldsymbol{f}_s$ and torques $\boldsymbol{\tau}_s$ measured by the

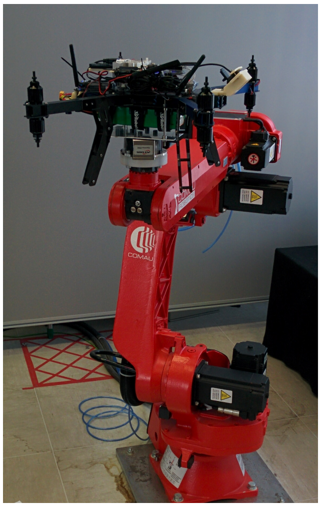


Figure A.9: *Experimental set-up employed to test the quadrotor inertial parameters. The 6 dofs industrial robot mounts to its end effector a forse sensor, to which the quadrotor is fixed.*

end effector and expressed in the world frame, can be computed as follows:

$$\begin{aligned} \boldsymbol{f}_s &= \boldsymbol{f}_g - m_q \dot{\boldsymbol{v}}_{cm} \\ \boldsymbol{\tau}_s &= R\left(-I_q {}^b\dot{\boldsymbol{\omega}}_q - {}^b\boldsymbol{\omega}_q \wedge I_q {}^b\boldsymbol{\omega}_q\right) + \left(R\,{}^b\boldsymbol{r}_{cm}\right) \wedge \left(\boldsymbol{f}_g - m_q \dot{\boldsymbol{v}}_{cm}\right) \end{aligned} \tag{A.1}$$

where $\boldsymbol{f}_g = \begin{bmatrix} 0 & 0 & -m_q g_a \end{bmatrix}^T$, with m_q the quadrotor mass and g_a is the gravity acceleration, $\boldsymbol{v}_{cm}$ is the center of mass velocity, R is the rotation matrix from the absolute frame to the quadrotor body frame, I_q is the diagonal quadrotor inertia matrix, ${}^b\dot{\boldsymbol{\omega}}_q$ and ${}^b\boldsymbol{r}_{cm}$ are the angular velocity of the quadrotor and the quadrotor center of mass position with respect the force sensor, respectively, both expressed in the quadrotor body frame. Kinematic quantities, as ${}^b\boldsymbol{\omega}_q$ and R can be easily computed from robot encoders applying direct kinematics.

Quadrotor Center of Mass

The quadrotor center of mass position can be estimated by measuring the torques at the end effector due to quadrotor gravity force. Performing quasi-static motion, it is assumed that inertial forces and torques are negligible. Then, from the model in (A.1), the torques measured by the force sensor $\boldsymbol{\tau}_s$ can be computed as follows:

$$\boldsymbol{\tau}_s = -\left[\boldsymbol{f}_g\right]_\times R\,{}^b\boldsymbol{r}_{cm} \tag{A.2}$$

where the operator $[\boldsymbol{x}]_\times$ computes the skew symmetric matrix such that $[\boldsymbol{x}]_\times \boldsymbol{b}$ is the result of the cross product of $\boldsymbol{x}$ with $\boldsymbol{b}$. Notice that the measured forces $\boldsymbol{f}_s$ expressed in the absolute frame can be approximated to be equal to gravitational force $\boldsymbol{f}_g$. Then, measuring forces $\boldsymbol{f}_s(t_i)$, torques $\boldsymbol{\tau}_s(t_i)$ and quadrotor orientation $R(t_i)$ for a set of N time instants t_i, the following matrices can be defined:

$$\boldsymbol{T}_s = \begin{bmatrix} \boldsymbol{\tau}_s(t_1) \\ \boldsymbol{\tau}_s(t_2) \\ \dots \\ \boldsymbol{\tau}_s(t_N) \end{bmatrix} \qquad A_s = \begin{bmatrix} -[\boldsymbol{f}_s(t_1)]_\times R(t_1) \\ -[\boldsymbol{f}_s(t_2)]_\times R(t_2) \\ \dots \\ -[\boldsymbol{f}_s(t_N)]_\times R(t_1) \end{bmatrix} \tag{A.3}$$

such that it is possible to obtain an estimate of the quadrotor center of mass as follows:

$$^b\hat{\boldsymbol{r}}_{cm} = A_s^\dagger \boldsymbol{T}_s \tag{A.4}$$

The equation in (A.4) requires that A_s is full rank, than, as $-[\boldsymbol{f}_s(t_i)]_\times$ is not invertible, at least two configurations are needed. In the real experiment, the robot is oriented such that the propellers axes are parallel to the ground plane (i.e. the quadrotor plane is orthogonal to the ground), and a quasi-static rotation is imposed about the z quadrotor axis in the body frame. The chosen velocity profile is a sinusoid, with an amplitude of $360°$. The complete sinusoid period is performed in 40 s. Center of mass position is computed as in (A.4), and it results:

$$^b\boldsymbol{r}_{cm} = \begin{bmatrix} 0.001 & 0.021 & 0.068 \end{bmatrix}^T \text{[m]} \tag{A.5}$$

In Fig. A.10 measured torques are compared with the torques reconstructed with the estimated center of mass position: $\hat{\boldsymbol{T}}_s = A_s {}^b\hat{\boldsymbol{r}}_{cm}$. Notice that inertial torques are negligible, as the acceleration imposed is sufficiently small.

Quadrotor Moments of Inertia

In order to compute quadrotor moments of inertia, a motion with sufficiently high angular acceleration has to be performed, in order to register non-negligible inertial torques. Since the inertia matrix is assumed to be diagonal, three different experiments have been performed to estimate the three principal inertial moments. A rotation around a chosen quadrotor principal axis is imposed, and torques are registered. Two motions with sinusoid velocity profiles are performed, whose amplitude is $45°$ and angular frequency equal to $\pi\frac{\text{rad}}{\text{s}}$. Thus, applying the relation in (A.1), considering

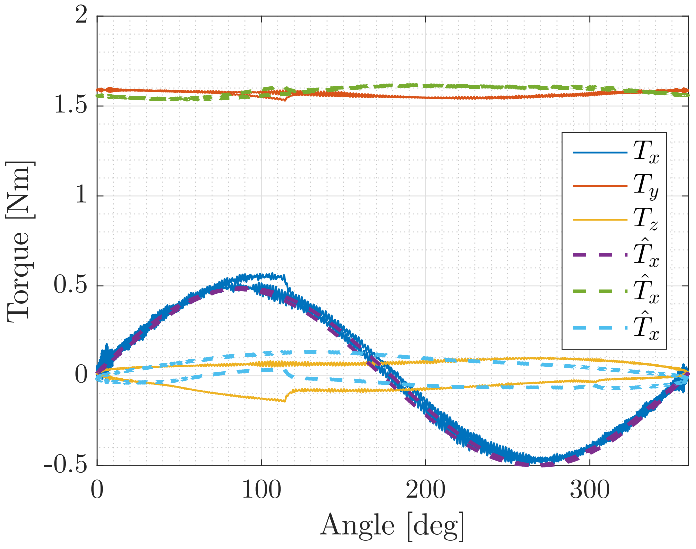


Figure A.10: *Comparison between torques measured by the sensor, represented with solid lines, and gravity torques computed with the estimated quadrotor center of mass position, represented with dashed lines.*

just one specific axis j, and employing the computed estimate of center of mass position ${}^b\hat{\boldsymbol{r}}_{cm}$, the following equation holds:

$$I_{q,j}\,{}^b\dot{\omega}_{q,j} = -{}^b\tau_{s,j} + \left[{}^b\hat{\boldsymbol{r}}_{cm} \wedge {}^b\hat{\boldsymbol{f}}_g\right]_j \tag{A.6}$$

where $[*]_j$ denotes the j component of $*$, and all the quantities have been expressed in the body frame. When a direct and accurate measure of angular acceleration ${}^b\dot{\omega}_{q,j}$ is provided, parameter $I_{q,j}$ can be directly estimated. Since a direct measure of acceleration is not available, two methods are applied.

The first approach is based on the computation of angular momentum. Thus, the two members of (A.6) are integrated. In this way, velocity measures can be employed, and an estimation of the inertia coefficients is obtained as follows:

$$\begin{aligned}
\Sigma_{T,i} &= \int_0^{t_i} \left(-{}^b\tau_{s,j} + \left[{}^b\hat{\boldsymbol{r}}_{cm} \wedge {}^b\hat{\boldsymbol{f}}_g\right]_j\right) dt \\
\left[{}^b\Omega_{q,j}\right]_i &= {}^b\omega_{q,j}\left(t_i\right) \\
\hat{I}^{\Sigma}_{q,j} &= \left({}^b\Omega_{q,j}\right)^{\dagger} \Sigma_T
\end{aligned} \tag{A.7}$$

where vectors Σ_T and ${}^b\Omega_{q,j}$ have dimension equal to the number of experiments measure N. Notice that pre-conditioning of the data are required on the function to be integrated. In fact, sensor bias has to be removed.

Another possible approach consists in computing a filtered derivative of the velocity obtaining an estimate of the angular acceleration. A derivative action with a first order low-pass filter $F_{LP}(s) = \frac{s}{\tau_{LP}s+1}$ is applied to the velocity signal, where the time constant τ_{LP} is equal to 0.03 s. Then, another estimate of the inertial coefficient is computed:

$$\begin{aligned} \boldsymbol{T}_{in,i} &= -{}^b\tau_{s,j}(t_i) + \left[{}^b\hat{\boldsymbol{r}}_{cm} \wedge {}^b\hat{\boldsymbol{f}}_g(t_i)\right]_j \\ \left[{}^b\hat{\dot{\Omega}}_{q,j}\right]_i &= {}^b\hat{\dot{\omega}}_{q,j}(t_i) \\ \hat{I}^{LP}_{q,j} &= \left({}^b\hat{\dot{\Omega}}_{q,j}\right)^\dagger \boldsymbol{T}_{in} \end{aligned} \tag{A.8}$$

where acceleration signals ${}^b\hat{\dot{\omega}}_{q,j}$ are computed with the low-pass filtered derivative mentioned above.

The two approaches are compared in Fig. A.11 for the case of the estimation of inertial coefficient I_x. In particular, measured torques T_s are

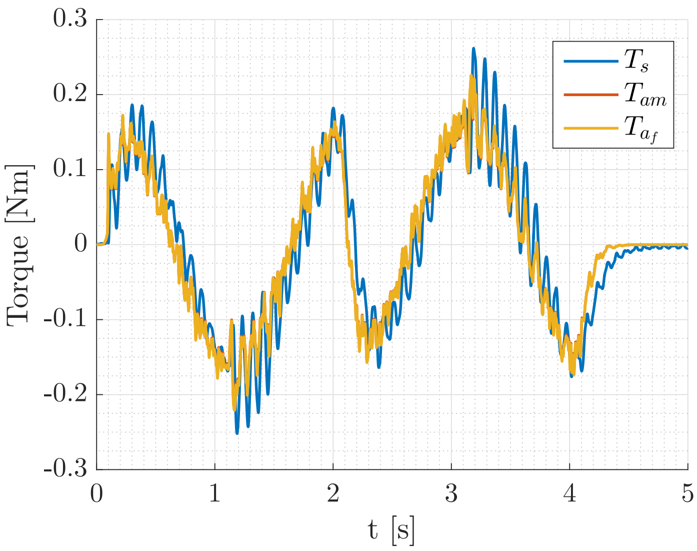


Figure A.11: *Comparison between measured torque, represented with blue line, and the torque computed with the inertial moment estimated with the first approach, reported with the red line, and computed employing the estimation obtained from the second approach, represented with the yellow line.*

compared with the torques obtained multiplying the filtered acceleration with the inertial moment estimates $\hat{I}_{q,x}^{\Sigma}$ and $\hat{I}_{q,x}^{LP}$. They are almost indistinguishable, as the two estimates are equal to $0.0433\,\mathrm{kgm}^2$ and $0.0438\,\mathrm{kgm}^2$, respectively. Notice that the difference between the two estimates is negligible if compared with the required accuracy. An estimate $\hat{I}_q$ of the matrix inertia has been obtained applying the same approach to the three quadrotor principal axes, and it results as follows:

$$\hat{I}_q = \begin{bmatrix} 0.043 & 0 & 0 \\ 0 & 0.027 & 0 \\ 0 & 0 & 0.072 \end{bmatrix} \left[\mathrm{kgm}^2\right] \tag{A.9}$$

Notice that, as expected, the sum of the principal moments $\hat{I}_{q,x}$ and $\hat{I}_{q,y}$ is almost equal to $\hat{I}_{q,z}$. This is due to the mass distribution of the quadrotor, which is mainly planar. Moreover, the principal moment $\hat{I}_{q,x}$ is higher than $\hat{I}_{q,y}$, as the quadrotor presents a X-shape of the geometry, as shown in Fig. A.13.

A.2.2 Robot Arm Inertia

The inertial parameters of robot arm links are computed from the CAD file of the structure. Detailed CAD models of the employed robot arm links are available on the website of the manufacturer company, and in Fig. A.12 a visualization of such models is reported.

A.3 Characterization of Actuation

In the following of the Chapter, a model of the actuators and the estimation of most relevant model parameters will be provided.

A.3.1 Quadrotor Propellers Parameters

Model

Quadrotor actuation is obtained assigning a PWM voltage signal to ESCs, which control propellers rotation velocity. The physical relation that models the dependence of the torques applied from propellers on a quadrotor and their rotation velocity is presented in [71]. Remind that the octacopter presented in Sec. A.1.1 is characterized by 8 propellers, and its geometry is represented in the scheme of Fig. A.13. Then, the relation between propellers angular velocities $\boldsymbol{\omega}_{m,q} = \begin{bmatrix} \omega_{m,1}^2 & \omega_{m,2}^2 & \dots & \omega_{m,8}^2 \end{bmatrix}^T$ and actuation

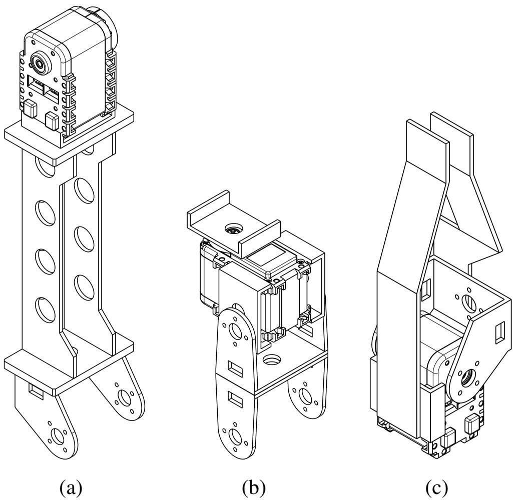


Figure A.12: *Visualization of the CAD files employed to compute an estimate of the inertia of the links composing the robotic arm.*

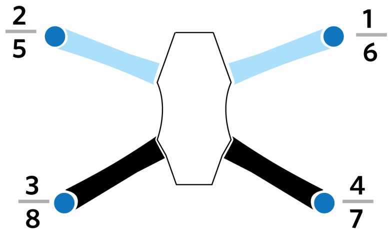


Figure A.13: *Position of the propellers and geometry of the quadrotor.*

torques $\boldsymbol{T}_m = \begin{bmatrix} T_z & \tau_\varphi & \tau_\theta & \tau_\psi \end{bmatrix}^T$ can be applied to the specific case as follows:

$$\boldsymbol{T}_m = D_m M_m \boldsymbol{\omega}_{m,q} \tag{A.10}$$

where M_m depends on shape of the drone, and D_m on the quadrotor dimension and aerodynamics parameters, as follows:

$$\begin{aligned} D_m &= \begin{bmatrix} \mu_T & 0 & 0 & 0 \\ 0 & \mu_T b & 0 & 0 \\ 0 & 0 & \mu_T b & 0 \\ 0 & 0 & 0 & \mu_D \end{bmatrix}, \\ M_m &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ s_\alpha & -s_\alpha & -s_\alpha & s_\alpha & -s_\alpha & s_\alpha & s_\alpha & -s_\alpha \\ -c_\alpha & -c_\alpha & c_\alpha & c_\alpha & -c_\alpha & -c_\alpha & c_\alpha & c_\alpha \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}, \end{aligned} \tag{A.11}$$

where μ_T and μ_D are aerodynamic thrust and drag coefficients, b is the length of the quadrotor arms, and α is the angle between the frontal symmetry axis of the quadrotor and the frontal arms. Notice that c_α and s_α are short forms for $\cos\alpha$ and $\sin\alpha$. Since the matrix M_m is assumed to be known with sufficient accuracy, it is needed to obtain an estimate $\hat{D}_m$ of matrix D_m, such that the reference propellers rotation velocities can be assigned as follows to apply the desired torques $\boldsymbol{T}_m^r$:

$$\boldsymbol{\omega}_{m,q} = M_m^\dagger \hat{D}_m^{-1} \boldsymbol{T}_m^r \tag{A.12}$$

where $M_m^\dagger$ is the right pseudo-inverse of M_m.

Experiment

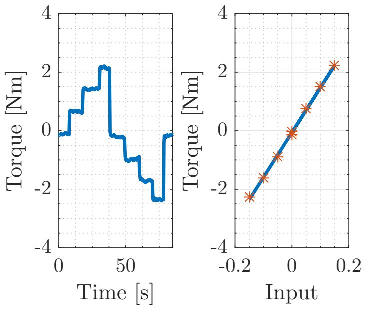
In order to obtain the estimate $\hat{D}_m$, reference rotation velocities in the following form are assigned:

$$\boldsymbol{\omega}_{m,q} = M_m^\dagger \boldsymbol{u}_T \tag{A.13}$$

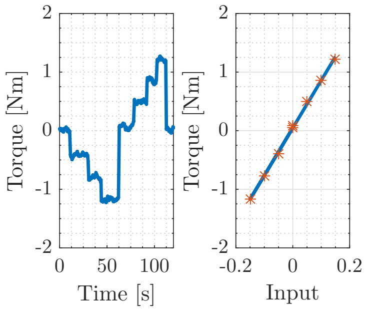
and the resulting torques $\boldsymbol{T}_m$ have to be measured. To this extent, the quadrotor is fixed to a 6 axis force-torque sensors in order to compute torques applied by the propellers on the quadrotor. Notice that, combining (A.13) with (A.10), the following relation between input and output is obtained:

$$\boldsymbol{T}_m = D_m \boldsymbol{u}_T. \tag{A.14}$$

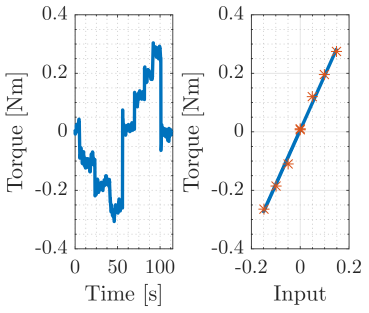
Being D_m a diagonal matrix, the four coefficients can be estimated separately. In Fig. A.14 the results of the four experiments are reported. For each quadrotor actuation degree, the behavior of the measured torque along time and in dependence of the given input $u_{T,i}$ is reported. Note that the linear coefficients can be easily identified. Thus, the estimate matrix $\hat{D}_m$ is



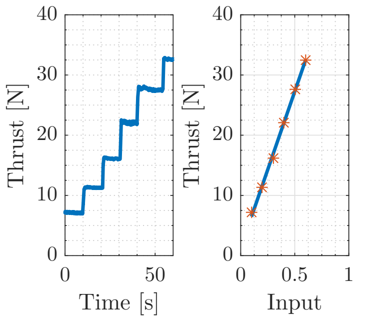
(a) Parameters estimation of roll actuation



(b) Parameters estimation of pitch actuation



(c) Parameters estimation of yaw actuation



(d) Parameters estimation of thrust actuation

Figure A.14: *Estimation of the characteristic parameters of the quadrotor actuation. For each degree of freedom of the quadrotor, two figures are reported. The figure on the left-hand side is the measured torque profile, obtained with a predefined sequence of control input. The figure on the right-hand side represents the experiments points, relating given input and measured torque, represented as red asterisks, and the estimated linear relation between the two quantities, represented with a blue line.*

obtained:

$$\hat{D}_m = \begin{bmatrix} 6.49 & 0 & 0 & 0 \\ 0 & 2.55 & 0 & 0 \\ 0 & 0 & 3.08 & 0 \\ 0 & 0 & 0 & 0.23 \end{bmatrix} \tag{A.15}$$

Note that roll and pitch torques coefficients should be equal, from the definition of D_m reported in (A.11), but they result slightly different. This can be explained by the geometry of the quadrotor, as propellers are not equidistant with each other. Aerodynamic effects due to the interaction among adjacent propellers can be the cause of this result.

As estimations of the aerodynamics parameters of the specific propellers employed in the chosen setup are available in literature, a simple verification of the results can be performed. In [18], the aerodynamics parameters of propellers APC - Slow Flyer - 11x4.7 are available. In particular, the aerodyamics modeling has been defined in [30]. The definitions of the aerodynamics coefficients C_t, C_q and C_p are reported here:

$$C_t = \frac{T}{\rho \omega^2 D^4} \quad C_q = \frac{Q}{\rho \omega^2 D^5} \quad C_p = 2\pi C_q, \tag{A.16}$$

where T is the thrust applied by the propeller, Q is the torque produced by the drag forces, ρ is the air density, and D and ω are the diameter and the angular velocity of the propeller, respectively. Experimental Tables exist for the coefficients C_t and C_p for the different propellers. Note that a direct comparison of obtained values and aerodynamics coefficient is not possible. In fact, in this Section a relation between PWM input and torques has been devised because measures of the actual angular rotation of the quadrotor propellers are not available. However, it can be shown that the ratio of the obtained coefficients can be compared, as follows:

$$\frac{\mu_D}{\mu_T} = \frac{Q}{T} = \frac{D}{2\pi}\frac{C_p}{C_t} \tag{A.17}$$

The propeller diameter D is known and it is equal to 11 inch, then 0.28 m. The behavior of coefficients C_t and C_p, for low quadrotor forward velocity (static case), is reported in Fig. A.15. The expected ratio $\frac{D}{2\pi}\frac{C_p}{C_t}$, computed employing the values of the figure, results to be equal, approximately, to 0.018 m. Instead, the ratio $\frac{\mu_D}{\mu_T}$ can be computed using the values of the estimated matrix $\hat{D}_m$. Obviously, coefficient μ_D is equal to $\hat{D}_{m,44}$, while three different values can obtained for the parameter μ_T, using the estimation of the thrust, roll and pitch coefficients:

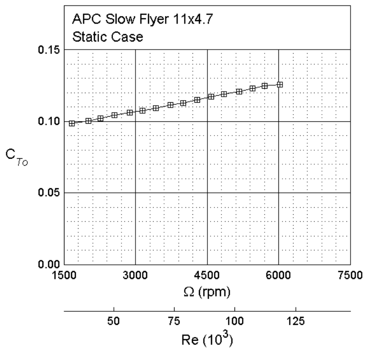
$$\mu_{T,1} = \hat{D}_{m,11}, \quad \mu_{T,2} = \frac{\hat{D}_{m,22}}{b} \quad \mu_{T,3} = \frac{\hat{D}_{m,33}}{b}, \tag{A.18}$$

Then, the ratios $\frac{\mu_D}{\mu_{T,1}}$, $\frac{\mu_D}{\mu_{T,2}}$ and $\frac{\mu_D}{\mu_{T,3}}$ are equal to 0.035 m, 0.021 m, 0.025 m. Notice that the obtained values are the same order of magnitude of value $\frac{D}{2\pi}\frac{C_p}{C_t}$, even if they are slightly higher.

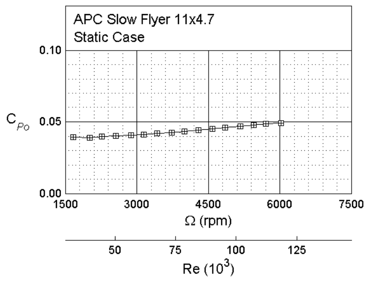
A.3.2 Robot Arm Actuation Parameters

The actuators of the robot arm are the small servos Dynamixel AX-12A. The adopted dynamic model of the single actuator is a first order linear relation between torque applied and angular shaft velocity, as follows:

$$G_m(s) = \frac{1}{J_m s + D_m} \tag{A.19}$$



(a) Coefficient C_t



(b) Coefficient C_p

Figure A.15: *Experimental characterization of parameters C_t and C_p for the propeller APC - Slow Flyer - 11x4.7 for low forward velocity. Images from UIUC Propellers Database, Applied Aerodynamics Group [18].*

.

where J_m and D_m are the motor inertia and the viscous friction coefficient. In order to estimate the two parameters, a series of step variation has been imposed to the servo torque, and the corresponding velocity has been acquired. In Fig. A.16, the Bode plot of a characteristic response of the system in (A.19) has been reported. With a linear system identification toolbox, the two coefficients have been obtained:

$$J_m = 9.8 \cdot 10^{-3} \mathrm{m\,kg}^2 \qquad D_m = 2.5 \cdot 10^{-1} \mathrm{N\,m\,s}/\mathrm{rad} \tag{A.20}$$

Notice that both parameters are referred to the link side of the servo. Due to the high gear ratio of the servo, i.e. $1 : 245$, the motor inertia of the servo is not negligible if compared with the link-side inertia of the arm.

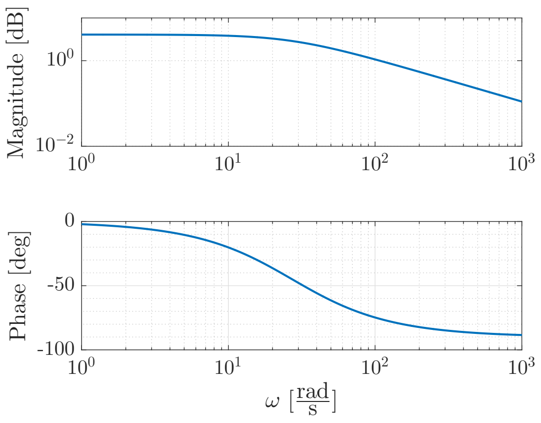


Figure A.16: *Bode plot of the transfer function relating the torque applied on the servo motor shaft, and the motor rotation velocity.*

APPENDIX $\mathcal{B}$

Proofs of the Propositions of Chapter 2

B.1 Load Transportation Case

A quadrotor equipped with a robotic arm has the main goal of interacting with the environment and manipulating specific objects. Thus, it is common that the aerial manipulator grasps an object with the tool at its end effector. If the object has a non negligible weight, the inertial properties of the overall system are affected and the position of the robot center of mass changes. Thus, it is interesting to question the effectiveness of the change of variables proposed in the previous section when an object is grasped, and whether the structure of the inertial matrices in the new variables are affected. The question arises from the fact that the transformation matrix T depends on the system center of mass, which changes in the mentioned case.

The position $\boldsymbol{x}_c^n$ and the velocity jacobian J_{co}^n of the centroid of the system composed of the robot arm and the grasped object can be easily computed:

$$\boldsymbol{x}_c^n = \frac{m_r}{m_r + m_m}\boldsymbol{x}_c^r + \frac{m_m}{m_r + m_m}\boldsymbol{x}_c^m$$

$$J_{co}^n = \frac{m_r}{m_r + m_m}J_{co}^r + \frac{m_m}{m_r + m_m}J_{co}^m$$

where m_r, $\boldsymbol{x}_c^r$, J_{co}^r and m_m, $\boldsymbol{x}_c^m$, J_{co}^m are the masses, the centroid position and the centroid velocity jacobian of the robot and the load, respectively.
If the object is grasped at the end effector, it can be assumed that the mass is concentrated in the end effector position. It means that $J_{co}^m = J_{eo}$.
The matrix J_{ec1} defined in (2.2b), necessary to the change of variables introduced in the previous section, has to be re-computed by taking into account the change of the centroid position caused by the grasped object. Then, the following relation holds:

$$\begin{aligned}
J_{ec1}^m &= T_m J_{ec1} \\
T_m &= \begin{bmatrix} I_3 & 0 \\ \frac{m_m}{m_r+m_m} I_{2,3} & \frac{m_r}{m_r+m_m} I_2 \end{bmatrix}
\end{aligned}$$

where $I_{2,3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$.
The matrix $T_{\beta\omega}$, defined in the previous section, can be computed for the system with the load:

$$T_{\beta\omega}^m = {J_{ec1,\alpha}^m}^\dagger J_{ec1,\omega}^m$$

where the weights matrix W_T has been neglected for simplicity, without any loss of generality. By substituting the definition of J_{ec1}^m and developing the pseudo-inversion operation, it results:

$$\begin{aligned}
T_{\beta\omega}^m &= J_{ec1,\alpha}^T T_m^T \left(T_m J_{ec1,\alpha} J_{ec1,\alpha}^T T_m^T \right)^{-1} T_m J_{ec1,\omega} \\
&= J_{ec1,\alpha}^T \left(J_{ec1,\alpha} J_{ec1,\alpha}^T \right)^{-1} J_{ec1,\omega} = T_{\beta\omega}
\end{aligned}$$

Thus, the matrix $T_{\beta\omega}$, and consequently the matrix T of the change of variable, are the same when the aerial manipulator is carrying a load at its end effector, then the matrices B and $J_{e\eta}$ keep the same structure and nice properties presented in (2.9).

B.2 Proof of Proposition 1

We will prove that the matrix $L_b h \in \mathbb{R}^{n-2 \times n-2}$ defined in (2.14) is not invertible, thus it has rank lower than $n-2$.
We can define the following system:

$$\boldsymbol{y} = L_b h\, \boldsymbol{v} \tag{B.1}$$

where $\boldsymbol{y} \in \mathbb{R}^{n-2}$, $\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1^T & \boldsymbol{y}_3^T \end{bmatrix}^T$, $\boldsymbol{y}_1 \in \mathbb{R}^2$, $\boldsymbol{y}_3 \in \mathbb{R}^{n-4}$, and $\boldsymbol{v} \in \mathbb{R}^{n-2}$, $\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_2^T & \boldsymbol{v}_3^T \end{bmatrix}^T$, $\boldsymbol{v}_2 \in \mathbb{R}^2$, $\boldsymbol{v}_3 \in \mathbb{R}^{n-4}$.

Given the structure of matrix $L_b h$ reported in (2.14), the system in (B.1) is equivalent to the following:

$$B\,\hat{\boldsymbol{y}} = \hat{\boldsymbol{v}}, \tag{B.2a}$$

$$\boldsymbol{y} = S_h\,\hat{\boldsymbol{y}}, \quad \hat{\boldsymbol{v}} = S_b\,\boldsymbol{v}, \tag{B.2b}$$

where $\hat{\boldsymbol{y}} \in \mathbb{R}^n$, $\hat{\boldsymbol{y}} = \begin{bmatrix}\boldsymbol{y}_1^T & \boldsymbol{y}_2^T & \boldsymbol{y}_3^T\end{bmatrix}^T$, $\boldsymbol{y}_2 \in \mathbb{R}^2$, and $\hat{\boldsymbol{v}} \in \mathbb{R}^n$, $\hat{\boldsymbol{v}} = \begin{bmatrix}0_2^T & \boldsymbol{v}_2^T & \boldsymbol{v}_3^T\end{bmatrix}^T$, $0_2 \in \mathbb{R}^2$. Taking into account the structure of matrix B, reported in (2.9), the first set of equations of (B.2) results as follows:

$$B_{tt}\,\boldsymbol{y}_1 + B_{t\beta}\,\boldsymbol{y}_3 = 0_2.$$

It means that the system in (B.1) admits a solution for $\boldsymbol{v}$ if and only if $\boldsymbol{y}$ satisfies the previous relation. As an obvious remark, the matrix $L_b h$ is full rank if and only if $\forall \boldsymbol{y} \in \mathbb{R}^{n-2}\ \exists\ \boldsymbol{v} \mid \boldsymbol{y} = L_b h\,\boldsymbol{v}$. Then, as the system does not satisfy the previous property, it can be concluded that the matrix $L_b h$ is not full rank.
In particular, the exact value of the rank of $L_b h$ matrix can be obtained. Starting from system in (B.2) and solving with respect to $\boldsymbol{y}_2$, the following equivalent system is obtained:

$$\begin{bmatrix}\boldsymbol{y}_1\\ \boldsymbol{y}_3\end{bmatrix} = \begin{bmatrix}B_{tt} & B_{t\beta}\\ B_{\beta t} & B_{\beta\beta} - B_{\beta\omega}B_{\omega\omega}^{-1}B_{\omega\beta}\end{bmatrix}^{-1}\begin{bmatrix}0 & 0\\ -B_{\beta\omega}B_{\omega\omega}^{-1} & I\end{bmatrix}\begin{bmatrix}\boldsymbol{v}_2\\ \boldsymbol{v}_3\end{bmatrix}$$

As this relation results from the problem in (B.2), which is equivalent to the system in (B.1), we obtain an expression for the matrix $L_b h$:

$$L_b h = \begin{bmatrix}B_{tt} & B_{t\beta}\\ B_{\beta t} & B_{\beta\beta} - B_{\beta\omega}B_{\omega\omega}^{-1}B_{\omega\beta}\end{bmatrix}^{-1}\begin{bmatrix}0 & 0\\ -B_{\beta\omega}B_{\omega\omega}^{-1} & I\end{bmatrix}$$

Thus, the rank of the matrix $L_b h$ is equal to $n-4$.
Notice that the rank of the matrix would be the same considering the input $\boldsymbol{v} = \boldsymbol{v}_3$.

B.3 Proof of Proposition 2

Define symmetric matrices B and $\hat{B}$, as follows:

$$B = \begin{bmatrix}B_{11} & B_{12} & B_{13}\\ B_{12}^T & B_{22} & B_{23}\\ B_{13}^T & B_{23}^T & B_{33}\end{bmatrix}, \quad \hat{B} = \begin{bmatrix}B_{11} & B_{13}\\ B_{13}^T & B_{33}\end{bmatrix}.$$

where $B_{11} \in \mathbb{R}^{n_1 \times n_1}$, $B_{22} \in \mathbb{R}^{n_2 \times n_2}$ and $B_{33} \in \mathbb{R}^{n_3 \times n_3}$. We want to prove that, if B is symmetric positive definite, matrix $\hat{B}$ results symmetric positive definite too. To this extent, define matrix X as follows:

$$X = \begin{bmatrix} I_{n_1} & 0_{n_1 \times n_3} \\ 0_{n_2 \times n_1} & 0_{n_2 \times n_3} \\ 0_{n_3 \times n_1} & I_{n_3} \end{bmatrix}$$

where $I_n \in \mathbb{R}^{n \times n}$ is an identity matrix, while $0_{m \times n} \in \mathbb{R}^{m \times n}$ is a null matrix. Thus, matrix $\hat{B}$ can be obtained as follows:

$$\hat{B} = X^T B\, X.$$

Being matrix X a full rank matrix and B symmetric positive definite, matrix $\hat{B}$ is symmetric positive definite.

B.4 Linearization of Euler angles derivatives

In this Appendix we will show the derivation of the linearized components of Euler angles derivative expressed in the new variables $\boldsymbol{\eta}$.
When the roll, pitch, and yaw representation is used, Euler angles derivatives can be computed with angular velocities by means of the matrix W:

$$\dot{\boldsymbol{\phi}} = W \boldsymbol{\omega}$$

where W is computed as follows:

$$W = \begin{bmatrix} 1 & {}^{s_\varphi s_\theta}\!/_{c_\theta} & {}^{c_\varphi s_\theta}\!/_{c_\theta} \\ 0 & c_\varphi & -s_\varphi \\ 0 & {}^{s_\varphi}\!/_{c_\theta} & {}^{c_\varphi}\!/_{c_\theta} \end{bmatrix}.$$

where s_x and c_x are short forms for $\sin x$ and $\cos x$, respectively. For small angles, the derivative of roll and pitch angles, results as follows:

$$\begin{bmatrix} \dot{\varphi} & \dot{\theta} \end{bmatrix}^T = \begin{bmatrix} \omega_x & \omega_y \end{bmatrix}^T + \begin{bmatrix} \theta & -\varphi \end{bmatrix}^T \omega_z$$

We can apply the change of variables introduced in (2.5), such that the dependence on $\boldsymbol{\beta}$ and $\boldsymbol{\omega}_{xy}$ is shown:

$$\begin{bmatrix} \dot{\varphi} & \dot{\theta} \end{bmatrix}^T = \boldsymbol{\omega}_{xy} + \begin{bmatrix} \theta & -\varphi \end{bmatrix}^T \left(\beta_\psi + T_{\beta_\psi \omega} \boldsymbol{\omega}_{xy} \right)$$

where $T_{\beta_\psi \omega}$ is the second row of matrix $T_{\beta\omega}$, corresponding to variable β_ψ. In the linearization, the term $\begin{bmatrix} \theta & -\varphi \end{bmatrix}^T T_{\beta_\psi \omega} \boldsymbol{\omega}_{xy}$ is neglected because it is

bilinear in the variables $\boldsymbol{\omega}_{xy}$ and $\begin{bmatrix}\varphi & \theta\end{bmatrix}$.
Then, the linearized derivatives of roll and pitch angles expressed in the new variables are:

$$\begin{bmatrix}\dot{\varphi} & \dot{\theta}\end{bmatrix}^T = \boldsymbol{\omega}_{xy} + \begin{bmatrix}\theta & -\varphi\end{bmatrix}^T \beta_{\psi}$$

Notice that in the paper the quantity $\boldsymbol{\sigma} = \begin{bmatrix}\theta & -\varphi\end{bmatrix}^T = D^T \begin{bmatrix}\varphi & \theta\end{bmatrix}^T$ is defined, such that:

$$\dot{\boldsymbol{\sigma}} = D^T \left(\boldsymbol{\omega}_{xy} + \boldsymbol{\sigma}\beta_{\psi}\right).$$

The linearized second order derivatives of roll and pitch $\begin{bmatrix}\ddot{\varphi} & \ddot{\theta}\end{bmatrix}^T$ are obtained with the same procedure, and it results:

$$\ddot{\boldsymbol{\sigma}} = D^T \left(\dot{\boldsymbol{\omega}}_{xy} + \dot{\boldsymbol{\sigma}}\beta_{\psi} + \boldsymbol{\sigma}\dot{\beta}_{\psi}\right)$$

B.5 Linearization of the rotation matrix

In this Appendix we will show the derivation of the linearized components of rotation matrix used for the computation of term in (2.41).
Notice that the first two components of term $g = T^{-T} g_0$ is equal to $g_t = \begin{bmatrix}I_2 & 0\end{bmatrix} R^T \begin{bmatrix}0 & 0 & \bar{g}_z\end{bmatrix}^T$, where I_2 is an identity matrix of dimension 2. Then the first and second components of the third columns of matrix R^T are needed.
When a roll, pitch, yaw representation is used, matrix R^T takes the following form:

$$R^T = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\varphi s_\theta c_\psi - c_\varphi s_\psi & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & s_\varphi c_\theta \\ c_\varphi s_\theta c_\psi + s_\varphi s_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & c_\varphi c_\theta \end{bmatrix}$$

For small roll and pitch displacements, the needed components are linearized as follows:

$$R^T_{xy} = \begin{bmatrix}-\theta & \varphi\end{bmatrix}^T.$$

From the last equation, the linear approximation for the gravity component results as follows: $g_t = \bar{g}_z \begin{bmatrix}-\theta & \varphi\end{bmatrix}^T = -\bar{g}_z \boldsymbol{\sigma}$.

B.6 Proof of Proposition 3

The symmetric positive definite matrices $B^{t\beta}$ and its inverse $B^{t\beta^{-1}}$ are given:

$$B^{t\beta^{-1}} = \begin{bmatrix} B_t^i & B_\beta^i \end{bmatrix}, \quad B^{t\beta} = \begin{bmatrix} B_t \\ B_\beta \end{bmatrix}$$

such that: $B_t B_t^i = I, \quad B_t^i B_t \neq I$. We define the following matrices:

$$A = I - B_\beta^i {B_\beta^i}^\dagger, \quad A_1 = B_t^i B_t$$

where ${B_\beta^i}^\dagger = \left({B_\beta^i}^T B_\beta^i\right)^{-1} {B_\beta^i}^T$. We want to demonstrate the following property:

$$AA_1 = A \tag{B.3}$$

It is easy to verify that both matrices A and A_1 are idempotent matrices. This implies that both matrices are diagonalizable, and their eigenvalues are either 0 or 1. In addition, the matrix A is symmetric, then its eigenvectors are orthogonal. If we diagonalize the matrix A, the following relation holds:

$$A = VDV^T,$$

where $V = \begin{bmatrix} V_1 & V_0 \end{bmatrix}$ and $D = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$. Columns of matrices V_1 and V_0 are the eigenvectors associated to the eigenvalues equal to 1 and 0, respectively. So, matrix A results:

$$A = V_1 V_1^T.$$

Then, we can observe that ${B_\beta^i}^T B_t^T = 0$ from the definiton of inverse, and consequently $AB_t^T = B_t^T$. Thus, the columns of B_t^T are a linear combination of the eigenvectors of A associated to eigenvalues 1. Equivalently, it is easy to verify that $AB_\beta^T = 0$, thus columns of B_β^T belong to the space whose base is formed by the columns of V_0. We define $\mathrm{im}(X)$ the image space of X, and $\oplus$ the operator performing the vector space direct sum. As $\mathrm{im}\big(B_t^T\big) \oplus \mathrm{im}\big(B_\beta^T\big)$ forms a basis of $\mathbb{R}^n$, and $\mathrm{im}(V_1) \oplus \mathrm{im}(V_0)$ forms a basis of $\mathbb{R}^n$, the columns of B_t^T form a basis of the columns space of V_1, and the reverse its true too.
From this observation, we can write the matrix V_1 as a function of B_t^T:

$$V_1 = B_t^T U. \tag{B.4}$$

where U is a square full rank matrix with suitable dimensions. If we write the left member of (B.3) as function of B_t^T, we obtain:

$$AA_1 = V_1 V_1^T A_1 = B_t^T UU^T B_t A_1.$$

Again, from the definition of inverse it results $B_t B_t^i = I$ and, consequently, $B_t A_1 = B_t$. Finally:

$$AA_1 = B_t^T UU^T B_t A_1 = B_t^T UU^T B_t = V_1 V_1^T = A.$$

This demonstrates the property in (B.3).

B.7 Proof of Proposition 4

The present Section will use the notations and results of Appendix B.6. We want to demonstrate that the matrix $B_t^{iT} A B_t^i$ is symmetric positive definite. The symmetry of the matrix is trivially verifiable.
Using the result of (B.4), we can write the matrix $B_t^{iT} A B_t^i$ as follows:

$$B_t^{iT} A B_t^i = B_t^{iT} B_t^T UU^T B_t B_t^i.$$

Then, from the definition of inverse, $B_t B_t^i = I$, thus it results:

$$B_t^{iT} A B_t^i = UU^T.$$

As shown in Appendix B.6, B_t^T forms a basis of V_1, then U is a square full rank matrix. Therefore, matrix $B_t^{iT} A B_t^i$ is symmetric positive definite.

B.8 Proof of Proposition 5

We want to obtain the values of $K_\phi > 0$ and $K_{t\beta} > 0$ for which P_1 is symmetric positive definite:

$$P_1 = \begin{bmatrix} \frac{K_{t\beta}^2}{\bar{g}_z^2}\left(2K_\phi - K_{t\beta}\right) I & \frac{K_{t\beta}^2}{\bar{g}_z} A B_t^i \\ \frac{K_{t\beta}^2}{\bar{g}_z} B_t^{iT} A & B_t^{iT} A B_t^i \left(K_\phi - K_{t\beta}\right) \end{bmatrix}.$$

It is useful to name the blocks composing the matrix P_1:

$$P_1 = \begin{bmatrix} P_A & P_B \\ P_B^T & P_C \end{bmatrix}. \tag{B.5}$$

The Schur complement condition can be applied. For this condition, given a symmetric matrix P_1 as in (B.5), if P_A is invertible, the matrix P_1 is positive definite if and only if both the following conditions are true:

i) P_A is positive definite,

ii) the Schur complement $P_1/P_A = P_C - P_B^T P_A^{-1} P_B$ is positive definite.

Trivially, the matrix P_A is invertible and positive definite if $K_\phi > \frac{1}{2}K_{t\beta}$. So, the Schur complement P_1/P_A results in the following nice form:

$$P_1/P_A = B_t^{i^T} A B_t^i \left(K_\phi - K_{t\beta} - \frac{K_{t\beta}^2}{2K_\phi - K_{t\beta}} \right)$$

Then, being $B_t^{i^T} A B_t^i$ symmetric positive definite, the Schur complement P_1/P_A is positive definite if $K_\phi > \frac{3}{2}K_{t\beta}$.
Finally, we can conclude that matrix P_1 is symmetric positive definite for $K_\phi > \frac{3}{2}K_{t\beta}$.

APPENDIX C

Experimental Activities relative to Chapter 5

The following Appendix reports the validation experiments relative to the contributions presented in Chapter 5. In particular, the validity of the damage assessment and the effectiveness of the control algorithm will be experimentally shown.

C.1 Experiments

To validate the control strategy proposed in Chapter 5, the 14-DOF dual-arm redundant manipulator prototype ABB FRIDA has been exploited. A controller coded MATLAB Simulink runs on an external PC in real-time dialog with the robot controller at a 250 Hz frequency, providing joint position and velocity reference to the internal robot controller. The Reflexxes Motion Libraries [59] have been exploited to allow real-time task trajectory generation, while the deployed hQP solver is the one proposed in [32]. Forces are measured with an ATI 6-axis force/torque sensor. The definition of layers for the hQP problem is reported in Tab. C.1, where: $R_p = 1$, $R_v = 0.1$, $\dot{s}^{\max} = 0.25$, $\ddot{s}^{\max} = 0.5$, $\boldsymbol{K}_p = \text{diag}\left(\begin{bmatrix}3 & 3 & 3 & 5 & 5 & 5\end{bmatrix}\right)$ and

$\boldsymbol{K}_d = \text{diag}\left(\begin{bmatrix}5 & 5 & 5 & 7 & 7 & 7\end{bmatrix}\right)$. In the performed experiments, the robot

	List of layers and corresponding constraints
1	$\min_{\ddot{\boldsymbol{z}}_k} R_p \left(s_{k+1} - s_{k+1}^{\text{ref}}\right)^2 + R_v \left(\dot{s}_{k+1} - \dot{s}_{k+1}^{\text{ref}}\right)^2$
	$\boldsymbol{q}_{\text{inf}} \leq \boldsymbol{q}_k \leq \boldsymbol{q}^{\text{sup}}$
	$\ddot{\boldsymbol{q}}_{\text{inf}} \leq \ddot{\boldsymbol{q}}_k \leq \ddot{\boldsymbol{q}}^{\text{sup}}$
	$0 \leq \dot{s} \leq \dot{s}^{\max}$
	$\lvert\ddot{s}\rvert \leq \ddot{s}^{\max}$
	$\boldsymbol{J}_k \ddot{\boldsymbol{q}}_k + \dot{\boldsymbol{J}}_k \dot{\boldsymbol{q}}_k = \ddot{\boldsymbol{x}}_k^{\text{ref}} + \boldsymbol{K}_d \dot{\boldsymbol{e}}_k + \boldsymbol{K}_p \boldsymbol{e}_k$
	$E_k \ \ddot{\boldsymbol{q}}_k \leq f_k$
2	$\min_{\ddot{\boldsymbol{z}}_k} -\sum_{i=1}^{N_i} \sum_{j=1}^{N_j} g_{ij,k} \ \ddot{\boldsymbol{q}}_k$
3	$\min_{\ddot{\boldsymbol{z}}_k} \left\lVert \dot{\boldsymbol{q}}_{k+1} \right\rVert^2$

Table C.1: *Definition of layers for the hQP problem*

task is defined w.r.t. the end effector motion and our damage assessment ΔT is computed only for a point located on the robot end effector. The proposed pre-impact control strategy has been validated in three different scenarios.
The first scenario analyzes a blunt inelastic impact with clamping. In the second, an impact with a stiff environment is studied. Finally, the third scenario shows the effects of the reflected mass minimization on the motion and on the impact properties. In the experiments the obstacle position is known in advance. This does not lead to any loss of generality since any method of obstacle position estimation can be integrated in our approach. In this Section the experimental results obtained in the three scenarios will be presented.

C.1.1 First Scenario: Clamped Inelastic Impact

The goal of the first scenario is to demonstrate that the dissipated energy presented in (5.3) provides a suitable description of the inelastic impact. The experiment consists in causing an impact between a robot and a clamped breakable element imposing given values of ΔT at the instant of impact. If the provided damage index is a good description of the physical phenomenon, the element will break for comparable values of ΔT just before the impact occurs. This behavior should not depend from the specific experiment condition, e.g. the robot configuration.

Organization of the Experiment

In the experiment i the robot end effector will perform a motion along z axis keeping bounded the ΔT under a specific threshold $\Delta T_{th,i}$. Along the trajectory the robot end effector will collide with a breakable element, a plate of polystyrene foam (dimensions: $350mm \times 100mm \times 10mm$) leaned on a support. An analysis of the residual torques based on [29] is implemented, performing an evasive post-collision strategy after the impact detection. This control strategy will be active in all the three scenarios. The experiments are repeated for three different values of $\Delta T_{th,i}$ and two different robot joint configurations, for a total of six experiments. Comparable deformations of the polystyrene plate are expected for the same values of $\Delta T_{th,i}$. The experiments in which the polystyrene foam has been broken have been repeated with a new plate, such that the fracture does not depend from the permanent deformations given by the previous impacts.

Results

The experiments are performed with $\Delta T_{th,i}$ respectively equal to $0.01\,\mathrm{J}$, $0.02\,\mathrm{J}$ and $0.03\,\mathrm{J}$ while the two end effector positions are $0.390\,\mathrm{m}$ far from each other. For the experiments with clamping, the parameter m_e is chosen equal to a value sufficiently high ($m_e = 20\,\mathrm{kg}$). As reported in Sec. 5.3, this choice correctly models a clamped impact. In addition, it can be a safety oriented choice for unclamped impacts, as it maximizes the value to be bounded. With the first $\Delta T_{th,1} = 0.01\,\mathrm{J}$, in both configurations the plate of polystyrene has an elastic behavior and the robot performs the evasive post-impact strategy causing no damage. Differently, with $\Delta T_{th,2} = 0.02\,\mathrm{J}$ the plate of polystyrene shows a small inelastic deformation in both configurations. Finally the two plates are broken in the third experiment with $\Delta T_{th,3} = 0.03\,\mathrm{J}$. It is remarkable that in the two configurations the qualitative behavior has been the same for equal $\Delta T_{th,i}$. The pictures of the impacts for the second and third ΔT_{th} in the two configurations have been reported in Fig. C.1. In the attached video the behaviors of the six experiments are showed and compared. As a conclusion, the dissipated energy chosen as danger index can describe a clamped inelastic impact since there is an actual correlation between the imposed ΔT and the effects of the impact, independently from the configuration.

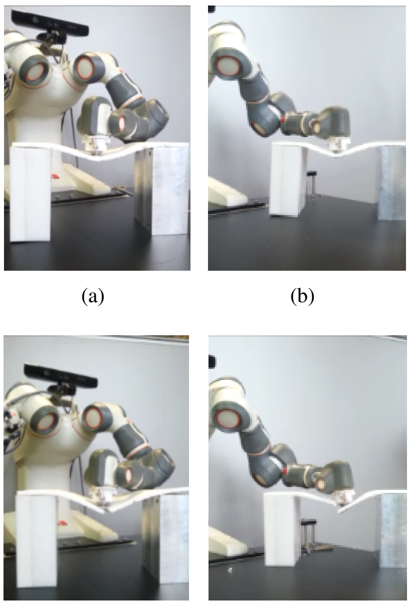

(a) (b) (c) (d)

Figure C.1: *Inelastic Impact with clamping. (a) $\Delta T_{th} = 0.02$, central position (b) $\Delta T_{th} = 0.02$, right position (c) $\Delta T_{th} = 0.03$, central position (d) $\Delta T_{th} = 0.03$, right position*

C.1.2 Second Scenario: Impact with a Stiff Environment

The second experiment aims at quantitatively describing the effect of bounding the threshold of ΔT under a certain value, by characterizing the force in a clamped impact. The use of a force sensor requires to perform an elastic impact to preserve the integrity of the sensor itself. Nevertheless, modeling the environment as an equivalent stiff spring, a functional relation between the ΔT just before the impact and the maximum force F_M exchanged with a stiff clamped environment can be obtained: $F_M \simeq \sqrt{K_{\text{env}} \Delta T}$, where K_{env} is the equivalent stiffness of the system at the end effector. Omitting the details, the relation has been obtained by linearizing the dynamical equation of the system around the state of the robot at the instant just before the impact, with the hypothesis that the control torques of the robot during the impact are negligible with respect to the inertial forces. Although it is largely used in literature [106], this approximation is quite stringent. Therefore, in the present work the relation is not intended to completely describe the obtained results, while it is used to justify the shape of the relationship between ΔT and F_M.

Organization of the Experiment and Results

During the experiment the robot end effector performs a descending vertical motion towards the force sensor and collides with it. The experiment is repeated 11 times, bounding the ΔT of the robot under different values of $\Delta T_{th,i}$. The plot in Fig. C.2 shows on the x axis the values of the ΔT_i of the robot just before the impact i, while the y axis reports the maximum force $F_{M,i}$ measured by the sensor during the same impact. A square root curve that fits the first 11 data is calculated with a linear regression and it is superimposed in the plot. Notice that, the last two samples, not considered in the fitting, show stronger influence of the post-collision strategy in limiting the maximum impact force. The high correlation coefficient $R^2 = 0.9984$ of the fitting curve clearly supports the functional relation presented in (C.1.2). The force profile in time, as measured by the force

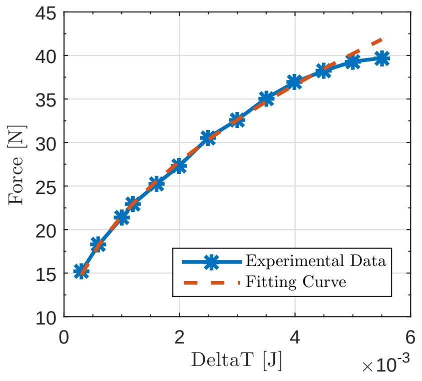


Figure C.2: *Relation between the ΔT index and maximum force measured in the impact, fitted with a square root curve.*

sensor, is shown in Fig. C.3. Only four profiles have been chosen for clarity. Plots have been synchronized by the instant in which the controller switches to the post-impact strategy, having detected a non null force acting on the robot (black dashed line). Therefore the effect of the robot control torques on the measured force is different in the two conditions: before the detection of the contact, the control torques contribute to increase the collision force, while after that instant the evasive control strategy contributes to decrease the force. The quantitative definition of these contributions are outside the goals of the present work.

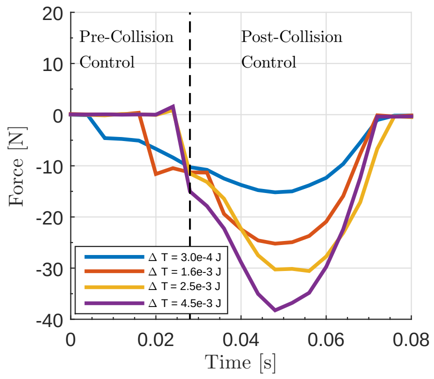


Figure C.3: *Profile of the force measured during impact in four experiments with different imposed ΔT.*

C.1.3 Third Scenario: Decreasing the Robot Reflected Mass

The last experiment aims at demonstrating the effectiveness of the manipulator reflected mass minimization implemented in the second level hierarchy of the control algorithm as explained in Sec. 5.4.2. Decreasing the reflected mass in the direction of a potential impact allows to increase the velocity in that direction, maintaining the same potential damage index. In the experiment it will be proven that the algorithm succeeds in decreasing the reflected mass keeping the same damage index, thus increasing the velocity while yielding comparable impact forces.

Organization of the Experiment

In the experiment, the robot end effector will perform a motion in the z axis ending in a collision with the force sensor exactly as in the previous experiment. The ΔT is kept bounded under a threshold of 0.005 J. The motion will be performed two times: in the first case the reflected mass minimization algorithm will not be active, while in the second case it will be active, resulting in a different null space motion and, eventually, in a different task velocity execution.

Results

Fig. C.4 shows the profile of the reflected mass before the impact. Notice that the algorithm succeeds in limiting the reflected mass of the robot in the direction of the impact (solid blue line) that can be compared to the profile of the reflected mass when no mass minimization algorithm is active (dashed cyan line). As a consequence, the velocity can be maintained

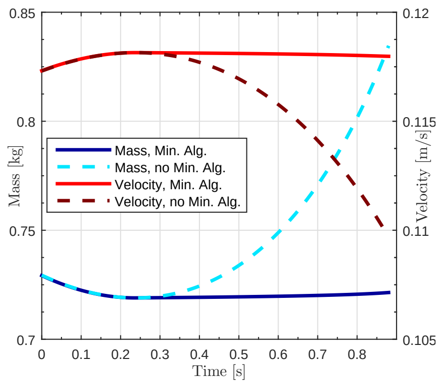


Figure C.4: *Mass and Velocity profile before the impact in the experiments with and without the reflected mass minimization algorithm.*

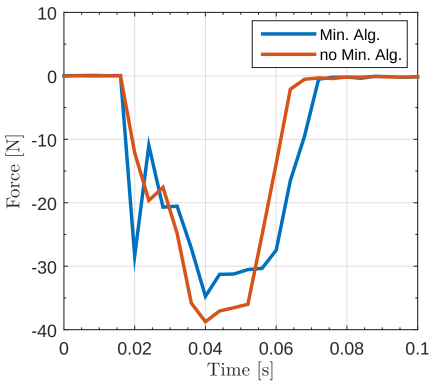


Figure C.5: *Profile of the force measured during impact in two experiments with and without the reflected mass minimization algorithm.*

constant (solid red line), while in the other case (dashed dark red line) the velocity has to decrease to keep the ΔT constant at the given threshold. In Fig. C.5 the force impact profile is shown. When the reflected mass minimization algorithm is active, the force reaches a slightly lower maximum force (approx. 12%), but it shows a higher slope at the first instants of the impact and a non homogeneous profile.

C.1.4 Summary of the Experimental Results

The results of the three experiments scenarios allow to draw the following conclusions

i) The index ΔT correctly describes the fracture energy in an inelastic impact.

ii) The pre-impact control strategy implemented succeeds in bounding the

ΔT under a desired threshold.

iii) Our damage assessment has direct functional dependence with the force exchanged in an elastic impact with stiff environment.

iv) The control strategy succeeds in minimizing the reflected mass of the end effector in the direction of a potential impact, allowing increased task efficiency.

The obtained results can be trivially extended to the case where more points of the robot and of the impact object are considered, combined with the weighting strategy proposed in Sec. 5.4. In addition, a detection strategy of the position of the object should be adopted, and it can be easily integrated in the control architecture.

List of Figures

List of Tables

Acronyms

C	
CoM	center of mass. 54, 60, 65–68, 70, 72, 73, 75, 77–79, 165, 166
D	
DoFs	degrees of freedom. 17, 19, 35, 37, 52, 53, 55, 59, 61, 64, 65, 74–77, 79, 165, 171
E	
ENU	East-North-Up. 39, 105
ESC	Electronic Speed Control. 131, 132, 139
F	
FAA	Federal Aviation Administration. 2
G	
GPS	Global Positioning System. 85, 86, 127
H	
hQP	hierarchical Quadratic Programming. 114, 155
I	
IMU	Inertial Measurement Unit. 85, 88

M	
Mavlink	Micro Aerial Vehicle Communication Protocol. 130
MPC	Model Predictive Control. 70
N	
NED	East-North-Up. 89, 100
NMPC	Non Linear Model Predictive Control. 53, 72, 75, 76, 78–80, 82, 165, 166
P	
PWM	Pulse Width Modulation. 127, 139, 143
Q	
QCQP	Quadratically Constrained Quadratic Program. 94, 98, 99
QPA	Quadratic Programming Approach. 80, 82
R	
ROS	Robot Operating System. 63, 64, 130, 131
S	
sID	standard inverse dynamics control. 37–43, 164
U	
UAM	Unmanned Aerial Manipulator. 4, 5, 52–55, 61–64, 66, 70, 71, 74, 76, 79, 80, 85, 87, 88, 109, 110, 165
UAV	Unmanned Aerial Vehicle. 2–5, 13–15, 36, 43, 52, 85, 86, 88
UWB	Ultra Wide Band. 86
V	
VTOL	Vertical Take-Off and Landing. 3, 4

Bibliography

[1] Horizon 2020 european project aeroarms: Aerial robotics system integrating multiple arms and advanced manipulation capabilities for inspection and maintenance. `https://aeroarms-project.eu/`, 2015. Accessed: 2016-10-08.

[2] Fp7 european project airobots: Innovative aerial service robotics for remote inspection by contact. `http://airobots.dei.unibo.it/`, 2010. Accessed: 2016-10-08.

[3] Fp7 european project arcas: Aerial robotics cooperative assembly system. `http://www.arcas-project.eu/`, 2011. Accessed: 2016-10-08.

[4] Fp7 european project aware. `http://cordis.europa.eu/project/rcn/91247_en.html`, 2009. Accessed: 2016-10-08.

[5] Fp7 european project sherpa: Smart collaboration between humans ans ground-aerial robots for improving rescuing activities in alpine environments. `http://www.sherpa-project.eu`, 2013. Accessed: 2016-10-08.

[6] Institut de robòtica i informàtica industrial (iri), csic-upc. `http://www.iri.upc.edu/`, 2016. Accessed: 2016-10-08.

[7] Mechatronics and robotics laboratory for innovation (merlin) of the politecnico di milano. `http://merlin.dei.polimi.it/`, 2016. Accessed: 2016-10-08.

[8] A. Albers, S. Trautmann, T. Howard, Trong Anh Nguyen, M. Frietsch, and C. Sauter. Semi-autonomous flying robot for physical interaction with environment. In *2010 IEEE Conference on Robotics, Automation and Mechatronics*, pages 441–446, June 2010.

[9] G. Antonelli, F. Arrichiello, and S. Chiaverini. The null-space-based behavioral control for autonomous robotic systems. *Intelligent Service Robotics*, 1(1):27–39, 2008.

[10] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, volume 1, pages 323–329, Oct 1998.

[11] K. Baizid, G. Giglio, F. Pierri, M. A. Trujillo, G. Antonelli, F. Caccavale, A. Viguria, S. Chiaverini, and A. Ollero. Behavioral control of unmanned aerial vehicle manipulator systems. *Autonomous Robots*, pages 1–18, 2016.

[12] T. Bartelds, A. Capra, S. Hamaza, S. Stramigioli, and M. Fumagalli. Compliant aerial manipulators: Toward a new generation of aerial robotic workers. *IEEE Robotics and Automation Letters*, 1(1):477–483, Jan 2016.

[13] S. Bellens, J. De Schutter, and H. Bruyninckx. A hybrid pose / wrench control framework for quadrotor helicopters. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 2269–2274, May 2012.

[14] C. D. Bellicoso, L. R. Buonocore, V. Lippiello, and B. Siciliano. Design, modeling and control of a 5-dof light-weight robot arm for aerial manipulation. In *Control and Automation (MED), 2015 23th Mediterranean Conference on*, pages 853–858, June 2015.

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

[16] M. Bernard and K. Kondak. Generic slung load transportation system using small size helicopters. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3258–3264, May 2009.

[17] M. Bernard, K. Kondak, I. Maza, and A. Ollero. Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, 28(6):914–931, 2011.

[18] J.B. Brandt, R.W. Deters, G.K. Ananda, and M.S. Selig. Propeller database of university of illinois at urbana-champaign, retrieved from `http://m-selig.ae.illinois.edu/props/propdb.html`. In *UIUC Propeller Database*, 2016.

[19] D. Brescianini, M. Hehn, and R. D'Andrea. Quadrocopter pole acrobatics. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3472–3479, Nov 2013.

[20] P. J. Bristeau, P. Martin, E. Salaün, and N. Petit. The role of propeller aerodynamics in the model of a quadrotor uav. In *Control Conference (ECC), 2009 European*, pages 683–688, Aug 2009.

[21] F. Caccavale, G. Giglio, G. Muscio, and F. Pierri. Adaptive control for uavs equipped with a robotic arm. *IFAC Proceedings Volumes*, 47(3):11049–11054, 2014.

[22] R.T.N. Chen. *A Survey of Nonuniform Inflow Models for Rotorcraft Flight Dynamics and Control Applications*. NASA (National Aeronautics and Space Administration), 1989.

[23] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, Jun 1997.

[24] G. Chmaj, T. Buratowski, T. Uhl, K. Seweryn, and M. Banaszkiewicz. The dynamics influence of the attached manipulator on unmanned aerial vehicle. In *Aerospace Robotics: Selected Papers from I Conference on Robotics in Aeronautics and Astronautics*, pages 109–119. Springer, Berlin, Heidelberg, 2013.

[25] P. I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[26] T. W. Danko, K. P. Chaney, and P. Y. Oh. A parallel manipulator for mobile manipulating uavs. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, May 2015.

[27] T. W. Danko and P. Y. Oh. A hyper-redundant manipulator for mobile manipulating unmanned aerial vehicles. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 974–981, May 2013.

[28] B. Datta. *Numerical Methods for Linear Control Systems*. Elsevier Academic Press, 2004.

[29] A. De Luca and R. Mattone. Sensorless robot collision detection and hybrid force/motion control. In *IEEE International Conference on Robotics and Automation, ICRA*, 2005.

[30] R.W. Deters, G.K. Ananda, and M.S. Selig. Reynolds number effects on the performance of small-scale propellers. In *Proceedings of 32nd AIAA Applied Aerodynamics Conference, American*, 2014.

[31] C. E. Doyle, J. J. Bird, T. A. Isom, C. J. Johnson, J. C. Kallman, J. A. Simpson, R. J. King, J. J. Abbott, and M. A. Minor. Avian-inspired passive perching mechanism for robotic rotorcraft. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4975–4980, Sept 2011.

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

[33] 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*, 18(8):816–830, 2008.

[34] 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*, 6(4):327–363, 2014.

[35] F. Forte, R. Naldi, A. Macchelli, and L. Marconi. Impedance control of an aerial manipulator. In *2012 American Control Conference (ACC)*, pages 3839–3844, June 2012.

[36] F. Forte, R. Naldi, A. Macchelli, and L. Marconi. On the control of an aerial manipulator interacting with the environment. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 4487–4492, May 2014.

[37] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor mav. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4557–4564, Oct 2012.

[38] P. J. From, I. Schjølberg, J. T. Gravdahl, K. Y. Pettersen, and T. I. Fossen. On the boundedness and skew-symmetric properties of the inertia and coriolis matrices for vehicle-manipulator systems. *IFAC Proceedings Volumes*, 43(16):193 – 198, 2010.

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

[40] S. Garrido-Jurado, R. Munoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marin-Jimenez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.

[41] M. Geisert and N. Mansard. Trajectory generation for quadrotor based systems using numerical optimal control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2958–2964, May 2016.

[42] G. Giglio and F. Pierri. Selective compliance control for an unmanned aerial vehicle with a robotic arm. In *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, pages 1190–1195, June 2014.

[43] A. Godil, R. Bostelman, K. Saidi, W. Shackleford, G. Cheok, M. Shneier, and T. Hong. 3d ground-truth systems for object/human recognition and tracking. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 719–726, June 2013.

[44] M. Hehn and R. D'Andrea. Quadrocopter trajectory generation and control. *18th IFAC World Congress*, 44(1):1485–1491, 2011.

[45] G. Heredia, A.E. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J.A. Acosta, and A. Ollero. Control of a multirotor outdoor aerial manipulator. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3417–3422, Sept 2014.

[46] B. Houska, H. J. Ferreau, and M. Diehl. Acado toolkit—an open-source framework for automatic control and dynamic optimization. *Opt. Cont. App. and Methods*, 32(3):298–312, 2011.

[47] F. Huber, K. Kondak, K. Krieger, D. Sommer, M. Schwarzbach, M. Laiacker, I. Kossyk, S. Parusel, S. Haddadin, and A. Albu-Schaffer. First analysis and experiments in aerial manipulation using fully actuated redundant robot arm. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3452–3457, Nov 2013.

[48] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, New York, third edition, 1995.

[49] A.E. Jimenez-Cano, G. Heredia, M. Bejar, K. Kondak, and A. Ollero. Modelling and control of an aerial manipulator consisting of an autonomous helicopter equipped with a multi-link robotic arm. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 230(10):1860–1870, 2016.

[50] A.E. Jimenez-Cano, J. Martin, G. Heredia, A. Ollero, and R. Cano. Control of an aerial robot with multi-link arm for assembly tasks. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 4916–4921, May 2013.

[51] A. Q. L. Keemink, M. Fumagalli, S. Stramigioli, and R. Carloni. Mechanical design of a manipulation system for unmanned aerial vehicles. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3147–3152, May 2012.

[52] M. Kemper and S. Fatikow. Impact of center of gravity in quadrotor helicopter controller design. In *Proc. of Mechatronics 2006*, pages 157–162. Elsevier, 09 2006.

[53] S. Kim, S. Choi, and H.J. Kim. Aerial manipulation using a quadrotor with a two dof robotic arm. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 4990–4995, Nov 2013.

[54] S. Kim, H. Seo, S. Choi, and H. J. Kim. Vision-guided aerial manipulation using a multirotor with a robotic arm. *IEEE/ASME Transactions on Mechatronics*, 21(4):1912–1923, Aug 2016.

[55] K. Kondak, F. Huber, M. Schwarzbach, M. Laiacker, D. Sommer, M. Bejar, and A. Ollero. Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 2107–2112, May 2014.

[56] K. Kondak, K. Krieger, A. Albu-Schaeffer, 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*, 10(145):1–9, 2013.

[57] K. Kondak, A. Ollero, I. Maza, K. Krieger, A. Albu-Schaeffer, M. Schwarzbach, and M. Laiacker. *Unmanned Aerial Systems Physically Interacting with the Environment: Load Transportation, Deployment, and Aerial Manipulation*, pages 2755–2785. Springer Netherlands, Dordrecht, 2015.

[58] C. M. Korpela, T. W. Danko, and P. Y. Oh. Designing a system for mobile manipulation from an unmanned aerial vehicle. In *2011 IEEE Conference on Technologies for Practical Robot Applications*, pages 109–114, April 2011.

[59] T. Kroeger and F.M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, 26:94–111, 2010.

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

[61] D. Lee. Passive decomposition and control of nonholonomic mechanical systems. *Robotics, IEEE Transactions on*, 26(6):978–992, Dec 2010.

[62] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control Systems*, 34(1):28–41, Feb 2014.

[63] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[64] H. Lim, J. Park, D. Lee, and H. J. Kim. Build your own quadrotor: Open-source projects on unmanned aerial vehicles. *IEEE Robotics Automation Magazine*, 19(3):33–45, Sept 2012.

[65] Q. Lindsey, D. Mellinger, and V. Kumar. Construction with quadrotor teams. *Autonomous Robots*, 33(3):323–336, 2012.

[66] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. Á. Trujillo, Y. Rodríguez Esteves, and A. Viguria. Hybrid visual servoing with hierarchical task composition for aerial manipulation. *IEEE Robotics and Automation Letters*, 1(1):259–266, Jan 2016.

[67] V. Lippiello and F. Ruggiero. Cartesian impedance control of a uav with a robotic arm. *IFAC Proceedings Volumes*, 45(22):704 – 709, 2012.

[68] V. Lippiello and F. Ruggiero. Exploiting redundancy in cartesian impedance control of uavs equipped with a robotic arm. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3768–3773, Oct 2012.

[69] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, Nov 2007.

[70] R. Mahony, T. Hamel, and J. M. Pflimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, June 2008.

[71] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *Robotics Automation Magazine, IEEE*, 19(3):20–32, Sept 2012.

[72] M. Mammarella, G. Campa, M. R. Napolitano, M. L. Fravolini, Y. Gu, and M. G. Perhinschi. Machine vision/gps integration using ekf for the uav aerial refueling problem. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(6):791–801, Nov 2008.

[73] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000.

[74] R. Mebarki, V. Lippiello, and B. Siciliano. Image-based control for dynamically cross-coupled aerial manipulation. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 4827–4833, Sept 2014.

[75] L. Meier, D. Honegger, and M. Pollefeys. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015.

[76] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2992–2997, May 2011.

[77] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[78] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 2668–2673, Sept 2011.

[79] A.Y. Mersha, S. Stramigioli, and R. Carloni. Variable impedance control for aerial interaction. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3435–3440, Sept 2014.

[80] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, April 2007.

[81] M. Orsag, C. Korpela, S. Bogdan, and P. Oh. Lyapunov based model reference adaptive control for aerial manipulation. In *Unmanned Aircraft Systems (ICUAS), International Conference on*, pages 966–973, May 2013.

[82] M. Orsag, C. M. Korpela, S. Bogdan, and P. Y. Oh. Hybrid adaptive control for aerial manipulation. *Journal of intelligent & robotic systems*, 73(1-4):693–707, 2014.

[83] I. Palunko, P. Cruz, and R. Fierro. Agile load transportation : Safe and efficient load manipulation with aerial robots. *IEEE Robotics Automation Magazine*, 19(3):69–79, Sept 2012.

[84] I. Palunko and R. Fierro. Adaptive control of a quadrotor with dynamic changes in the center of gravity. *IFAC Proceedings Volumes*, 44(1):2626 – 2631, 2011. 18th IFAC World Congress.

[85] C. Papachristos, K. Alexis, and A. Tzes. Efficient force exertion for aerial robotic manipulation: Exploiting the thrust-vectoring authority of a tri-tiltrotor uav. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 4500–4505, May 2014.

[86] 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*, 368:69120, 2010.

[87] P.E.I. Pounds, D.R. Bersak, and A.M. Dollar. Grasping from the air: Hovering capture and load stability. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 2491–2498, May 2011.

[88] P.E.I. Pounds and A. M. Dollar. Uav rotorcraft in compliant contact: Stability analysis and simulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2660–2667, Sept 2011.

[89] P.E.I. Pounds and A. M. Dollar. Stability of helicopters in compliant contact under pd-pid control. *IEEE Transactions on Robotics*, 30(6):1472–1486, Dec 2014.

[90] P.E.I. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691 – 699, 2010. Special Issue on Aerial Robotics.

[91] B. Povse, D. Koritnik, Tadej Bajd, and M. Munih. Correlation between impact-energy density and pain intensity during robot-man collision. In *IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 179–183, 2010.

[92] A. Del Prete, F. Romano, L. Natale, G. Metta, G. Sandini, and F. Nori. Prioritized optimal control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2540–2545, May 2014.

[93] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer Berlin Heidelberg, 2007.

[94] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[95] 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*, pages 1–6, Oct 2014.

[96] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, S. Weiss, and L. Meier. Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics Automation Magazine*, 21(3):26–40, 2014.

[97] I. Schjølberg, J. T. Gravdahl, K. Y. Pettersen, T. I. Fossen, et al. On the boundedness property of the inertia matrix and skew-symmetric property of the coriolis matrix for vehicle-manipulator systems. *Journal of Dynamic Systems, Measurement, and Control*, 134(4):1–3, 2012.

[98] J.L.J. Scholten, M. Fumagalli, S. Stramigioli, and R. Carloni. Interaction control of an uav endowed with a manipulator. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 4910–4915, May 2013.

[99] M. Sharifi and H. Sayyaadi. Nonlinear robust adaptive cartesian impedance control of uavs equipped with a robot manipulator. *Advanced Robotics*, 29(3):171–186, 2015.

[100] S. Shen, N. Michael, and V. Kumar. Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015*, pages 5303–5310, 2015.

[101] R. Spica, A. Franchi, G. Oriolo, H.H. Bulthoff, and P. R. Giordano. Aerial grasping of a moving target with a quadrotor uav. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 4985–4992. IEEE, 2012.

[102] 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*, pages 4888–4895, 2013.

[103] A. Suarez, G. Heredia, and A. Ollero. Lightweight compliant arm for aerial manipulation. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1627–1632, Sept 2015.

[104] J. Thomas, G. Loianno, J. Polin, K. Sreenath, and V. Kumar. Toward autonomous avian-inspired grasping for micro aerial vehicles. *Bioinspiration & Biomimetics*, 9(2):1–15, Jun. 2014.

[105] A. Torre, D. Mengoli, R. Naldi, F. Forte, A. Macchelli, and L. Marconi. A prototype of aerial manipulator. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2653–2654, Oct 2012.

[106] I. D. Walker. Impact configurations and measures for kinematically redundant and multiple armed robot systems. *IEEE Transaction on Robotics and Automation*, 10:670–1683, 1994.

[107] M. Wolf and S. McQuitty. Understanding the do-it-yourself consumer: Diy motivations and outcomes. *AMS Review*, 1(3):154–170, 2011.

[108] H. Yang and D. Lee. Dynamics and control of quadrotor with robotic manipulator. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 5544–5549, May 2014.

[109] A. M. Zanchettin and P. Rocco. Near time-optimal and sensor-based motion planning for robotic manipulators. In *IEEE Conference on Decision and Control, CDC*, 2013.

[110] 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*, pages 2748–2753, May 2015.

[111] 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.

[112] 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*, pages 1829–1834, Aug 2012.