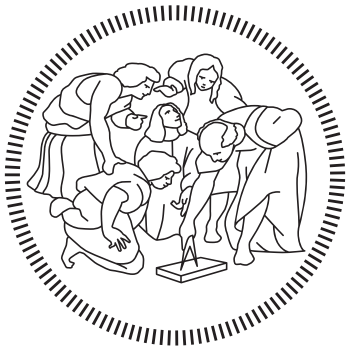
Politecnico di Milano
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Aeronautical Engineering



UAV autonomous landing on moving aerial vehicle

Advisor: Prof. Marco LOVERA
Co-Advisor: Dr. Simone PANZA
Eng. Mattia GIURATO
Dr. Davide INVERNIZZI

Thesis by:
Giovanni GOZZINI Matr. 874252

Academic Year 2018–2019

Ai miei genitori
grazie ai quali sono diventato
la persona che sono.

Acknowledgments

Giunto al termine di questi cinque anni da studente universitario, ci sono alcune persone che desidero ringraziare personalmente.
Il primo ringraziamento va al professor Marco Lovera che mi ha dato la possibilità di svolgere la tesi su un argomento interessante e ha sempre messo a disposizione le sue competenze. Un grazie di cuore va a Simone che, con le sue conoscenze tecniche e le sue doti umane, mi ha seguito durante tutta l'attività sperimentale, tra schianti e atterraggi di successo, mostrando grande dedizione, infinita pazienza e altruismo. Un grosso grazie va anche a Mattia e a Davide che hanno messo a disposizione le loro abilità e capacità con consigli e suggerimenti.
A seguire ringrazio Davide e Daniele con cui ho trascorso questi ultimi mesi in Lab mentre lavoravamo alla tesi. Grazie mille a Manu, Gian, Marco, Francio, Tullio e Umberto compagni di corso e non che mi hanno accompagnato e aiutato in questi anni di studio al Politecnico.
Volevo poi ringraziare gli amici con i quali ho condiviso le mie avventure, ognuno mi è stato vicino a modo suo. Grazie mille Tito, Castre, Laura, Fiamma, Vale e Benni. Un pensiero va anche alle persone con cui condivido la mia passione per la musica, in special modo a Simo e Stefano che negli ultimi anni mi hanno sopportato anche nei momenti difficili.
Un grazie speciale a Gretha che mi ha sostenuto e supportato con pazienza in questi ultimi mesi.
Un pensiero particolare va a Marc, un compagno di studi ma soprattutto un amico staordinario, con cui ho condiviso praticamente ogni giorno di questi ultimi anni, a cui devo tutto e a cui sarò sempre grato.
Infine un ringraziamento enorme alla mia famiglia. Ai miei nonni che hanno sempre creduto in me, a mio fratello Lorenzo e a mia sorella Elisa che mi sono stati sempre vicini, a mamma Martina e a papà Alberto che mi hanno aiutato e confortato nei momenti difficili e con mille sacrifici mi hanno permesso di realizzare ogni mio obiettivo.

Abstract

Nowadays, the unmanned aerial vehicles (UAVs) represent a research field that is continuously increasing and expanding. In particular, the interaction between more aircraft during flight is getting more attention; especially for what concerns formation flight and air-to-air refuelling. When surveillance, reconnaissance and search-and-rescue missions, both in the military and the civil fields, are considered, small-scale UAVs suffer of low autonomy problems. For this reason, to extend mission endurance a drone can be used as carrier for lighter and smaller drones (followers) that can take-off from and land on it. In real applications to guarantee enough endurance during missions the carrier might be powered by hybrid or conventional propulsion and not by pure electric propulsive system. To increase the efficiency the carrier could also be a fixed-wing or a tiltrotor UAV; in the latter case the more efficient aeroplane mode could be employed for loiter while the rotorcraft mode could be used to simplify the landing operation.
The purpose of this thesis is to investigate the problem of automatic landing of a small UAV on a multi rotor carrier drone in indoor environment, considering two different conditions: the first one with carrier moving along a circular trajectory with constant speed, simulating a loiter condition, the second one with carrier oscillating vertically.
Guidance laws for autonomous landing with carrier moving in horizontal plane and oscillating in vertical direction are implemented and simulated. Eventually, the guidance laws are successfully tested and validated through experimental activity in Flying Arena for Rotorcraft Technologies (FlyART) of Aerospace Systems and Control Laboratory (ASCL) of Politecnico di Milano.

Sommario

Oggigiorno, i velivoli a pilotaggio remoto rappresentano un campo di ricerca in continua crescita e espansione. In particolare, l'interazione tra più velivoli durante il volo sta ricevendo sempre maggiore attenzione, specialmente per quanto riguarda il volo in formazione e il rifornimento aria-aria. Considerando le missioni di sorveglianza, ricognizione e ricerca e salvataggio, sia nel campo militare che in quello civile, gli UAV di piccole dimensioni soffrono di problemi di bassa autonomia oraria. Per questa ragione, per estendere l'autonomia oraria, un drone può essere usato come "cargo" per droni più piccoli e più leggeri ("inseguitori") che possono decollare e atterrare su di esso. In applicazioni reali per garantire abbastanza autonomia durante le missioni il drone cargo potrebbe essere alimentato tramite propulsione ibrida o convenzionale e non da un sistema propulsivo puramente elettrico. Per aumentare l'efficienza il cargo potrebbe anche essere un UAV ad ala fissa o un convertiplano; in quest'ultimo caso la modalità aeroplano, più efficiente, potrebbe essere utilizzata per la fase di loiter mentre la modalità elicottero potrebbe essere utilizzata per semplificare l'operazione di atterraggio.
Lo scopo di questa tesi è di investigare il problema dell'atterraggio automatico di un piccolo UAV su un drone cargo multirotore in ambiente chiuso, considerando due diverse condizioni: la prima con il cargo in movimento a velocità costante lungo una traiettoria circolare, simulando la condizione di loiter, la seconda con il cargo che oscilla verticalmente.
Le leggi di guida sono implementate e simulate per l'atterraggio autonomo con il drone cargo in movimento nel piano orizzontale, nel primo caso, e oscillante in direzione verticale nel secondo. Infine, le leggi di guida sono state testate e validate con successo attraverso l'attività sperimentale condotta nell'arena FlyART del laboratorio di controllo e sistemi aerospaziali (ASCL) del Politecnico di Milano.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV) is an aircraft able to fly autonomously or piloted from the ground, anyhow without a pilot aboard. Usually called drones, in recent years this type of vehicles have received increasing attention both in civil and military fields thanks to their wide range of application, *e.g.*, aerial inspection, policing and surveillance, search and rescue, precision agriculture, photography, product delivery and entertainment.
While there is a great variety of types of UAVs ([5]), in this thesis we focus on the category of multirotor Vertical Take-Off and Landing (VTOL) vehicles provided with at least four motors, of small/medium size and remotely controlled. Thanks to their versatility in technical operation, they are widespread in commercial and research fields.
When surveillance, reconnaissance and search-and-rescue missions are considered, these small-scale UAVs suffer of low endurance, being usually powered by batteries. For this reason, to extend mission endurance a drone can be used as carrier for lighter and smaller drones (followers) that can take-off from and land on it. This procedure requires the interaction between the two UAVs. Recent research activities deals with the interaction between more aircraft during flight, especially thinking about formation flight and air-to-air refuelling, studying the possibility to remotely command many drones together, following the same path or performing many tasks at the same time.
In this thesis the objective is to develop and validate a new set of guidance, navigation and control laws enabling air-to-air UAV landing. In particular the thesis focuses on the design of guidance laws aimed at providing a small multirotor with a reference descent trajectory. This kind of manoeuvre is as risky as dangerous. The wake of the propellers of a multirotor generates an unsteady flow field around it, such that when two UAVs fly close they perturb each other. During landing operation the follower drone is always above the carrier, which constantly flies in a perturbed regime as stronger as the vehicles are closer.
Obvious considerations regarding the size and weights of the two UAVs must be taken into account, *e.g.*, it is necessary that the carrier drone must be larger and

heavier than the follower, in order to be able to stand its weight after the touch down. Moreover the on-board controller must be able to reject at least part of the aerodynamic disturbances.
The hazardous nature of this manoeuvre makes the design of the trajectory complex, being the problem three-dimensional and the carrier in movement. To overcome these issues a decoupled approach is adopted: first a horizontal synchronization between follower and target must be achieved, then the reference landing path is computed and sent to the follower. While the two drones are not in the same horizontal position, the landing is stopped until the synchronization is obtained again.

State of the art

In the literature there are examples of autonomous landing of UAVs on platforms attached to ground vehicles or ships. On the other hand, only one reference ([6]) investigates the more difficult air-to-air landing, but under the hypothesis that target is not moving. In this section, an overview of some recent studies is provided.

Hu, Lu and Mishra [7], [8], [9] investigated the problem of fast, safe and precise landing of Vertical Take-Off and Landing (VTOL) UAV onto a vertically oscillating platform. The control architecture to achieve these goals consists of three different modules:

- a trajectory generation module, in which a time-optimal reference trajectory for the quadrotor is generated, through a bang-zero-bang algorithm, such that it converges from the initial height to the platform height with the lowest possible velocity (*i.e.*, achieving a smooth landing);
- a tracking control module, in which an Adaptive Robust Controller (ARC) is designed to robustly adapt the non-linear ground effect to enable the quadrotor accurately track a given reference trajectory;
- a motion estimation module, in which the positions and velocities of UAV and platform are estimated online from only measurement of relative distance and acceleration (IMU) using an Unscented Kalman Filter (UKF).

Borowczyk et al. [10] investigated the problem of automated landing of a micro aerial vehicle (MAV) on a ground vehicle moving at relatively high speed, using only commercially available and relatively low-cost sensors. The guidance and control system switches between a Proportional Navigation (PN) law for the long range approach phase and a Proportional-Integral-Derivative (PID) law for close

range and landing phase. The authors used the AprilTag visual fiducial system placed on the landing pad and a camera mounted on the MAV to obtain relative position. The estimation of the relative position, velocity and acceleration between the moving aerial vehicle and the ground landing pad, required by the guidance and control system, is performed by a Kalman Filter.

F. Alarcon et al. [11] developed a precise landing system that allows rotary-wing UAVs to approach and land safely on moving platforms, without using GNSS at any stage of the landing manoeuvrer, with a centimetre level accuracy and high level of robustness. This system implements a novel concept where the relative position and velocity between the aerial vehicle and the landing platform are calculated from the angles of a cable that physically connects them. The use of a cable also increases the precision in the control of the UAV altitude, facilitates centring the UAV right on top of the expected landing position and increases the stability of the UAV just after contacting the landing platform. Results show that the developed system allowed landing with centimetre accuracy by using only local sensors and that the helicopter could follow the landing platform in multiple trajectories at different velocities.

Kim, Jung et al. [12] proposed a vision-based target following and landing system for a quadrotor vehicle on a moving platform. Algorithms developed for detecting the target are based on the color appearing on the image, the target geolocation consists on locating the pad on the navigation frame and is obtained using outputs of vision sensors. A Kalman filter is used to estimate position and velocity of the target in the navigation frame. This system has been validated by flight tests in outdoor environment and can be used for following or landing on specific platforms such as ground carrier vehicle or ship-boards with visual markers.

Giuri, Marini Cossetti et al. [13], [6] implemented and validated through experimental activity two non-linear time-optimal guidance laws to obtain an air-to-air automatic landing of a small quad-copter on a bigger hexacopter used as a carrier. They used a bang-bang control logic (maximum deceleration and acceleration) with relative acceleration as control variable to solve a time-optimal path generation problem with saturations in the acceleration set-point. With this type of control law, velocity might reach undesired high speeds, so an additional constraint is imposed to keep the velocity bounded, resulting in a bang-zero-bang control law.

Thesis structure

The thesis is organized as follows:

- in Chapter 2 modelling and simulation of multirotor UAV flight is presented;

in particular reference frames, rotational formalism, flight dynamics and notation used in the thesis are described. Results and limits of application of the work done by Giuri and Marini Cossetti [13] in their master thesis are reported. Finally the general architecture of guidance, navigation and control problem is presented.

- In Chapter 3 the procedure for tuning position and attitude controller of the two drones is described. In particular the identification of vertical, roll and pitch dynamics of the carrier and lateral and roll dynamics of the racer are used to make position control more aggressive and to improve the performance of the trajectory tracking system.

- In Chapter 4 the in-plane synchronization and landing procedure is analysed; specifically, a mechanism to generate the follower trajectory is presented. Simulation results obtained in MATLAB are also reported.

- In Chapter 5 the case of landing with the target oscillating in the vertical plane is discussed and simulated in the MATLAB environment.

- In Chapter 6, after a brief description of the experimental system setup (hardware and software), a preliminary study about the landing performance degradation in presence of noisy and reduced frequency position measurements is conducted. Finally, flight test results for the problems of landing with target moving in-plane and oscillating vertically are presented and compared with simulation results.

Chapter 2

Problem formulation

This chapter is divided in three main parts: in the first one modelling and simulation of multirotor UAVs and notation used are described; in the second part the results of the previous work [13] are recalled, paying attention especially to limitations encountered and finally, in the last part, the overall control architecture is described.

2.1 Modelling and simulation of multirotor UAVs

In this section a brief introduction about modelling and simulation of aircraft dynamics is presented, for a general treatment of the subject see [14].

2.1.1 Reference frames

In order to describe the motion of a flying vehicle, at least two reference frames are needed, defined as follows.

Earth axes

Considering that the UAVs fly indoor and close to Earth's surface, the hypothesis of flat and non-rotating Earth is assumed. The Earth fixed frame is defined as $\mathcal{F}_E = \{O_E,\ e_{1E},\ e_{2E},\ e_{3E}\}$, where the first element is the origin while the others are three unit vectors. The origin is arbitrary, it could be the intersection of the equator, the prime meridian and mean sea level. The standard convention has e_{1E} pointing North, e_{2E} pointing to the East and e_{3E} aligned with the direction of gravity, pointing downward. This type of reference frame is often referred to as NED, meaning North-East-Down; these three axes are mutually perpendicular and, when referred in N, E, D order, form a right-handed coordinate system.

Body axes

To write the equations of motion of an aircraft it is easier to use a different frame instead of considering a reference attached to the Earth. A moving reference frame centred in the centre of gravity of the vehicle can be used. In this thesis, the body reference frame $\mathcal{F}_B = \{O_B,\ e_{1B},\ e_{2B},\ e_{3B}\}$ refers to a right-handed system of coordinates, rigidly attached to the UAVs centre of gravity ($O_B \equiv CG$), and changing orientation with it. The unit vector e_{1B} lies in the plane of symmetry of the vehicle and points forward, e_{2B} is normal to the plane of symmetry pointing rightward and e_{3B} points downward (frd convention).

2.1.2 Rotation formalism

Euler angles

The Euler angles (ϕ, θ, ψ) are three independent angular quantities able to describe the 3D orientation of an object, using two sets of reference frames: an inertial one, Earth-fixed, and a body one rigidly attached to the object. The position of the axes is therefore obtained by an ordered sequence of three rotations so as to bring the two sets to coincide.
The Euler angles define the transformation of the components of a generic vector between two sets of axes. The rotation matrix R_A^B represents a rotation from system A to system B. Thus, a vector v_A in A coordinate system can be resolved to B coordinate system obtaining v_B through the matrix operation:

$$v_B = R_A^B v_A. \tag{2.1}$$

To understand how these rotation matrices work, it is possible to consider rotations about one axis at time. The rotation about the x-axis does not change the component of the vector directed along the same axis, but it does change the y and z components. The rotation matrix that describes this transformation is

$$R_X(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}. \tag{2.2}$$

In a similar fashion the following matrices perform rotations about the y-axis and the z-axis, respectively:

$$R_Y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{2.3}$$

$$R_Z(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

The rotation angle is measured about (around) the axis of the rotation with the sign convention given by the right-hand rule. The location of 0° is fixed arbitrarily, but once it is chosen it must not vary. The rotation matrices are orthonormal, which means the columns in the matrices, if considered as vectors, have unit magnitude (normal) and are mutually orthogonal. For this reason inverse and transpose matrices coincide, in formula:

$$\begin{aligned} R_X^{-1}(\phi) &= R_X^T(\phi), \\ R_Y^{-1}(\theta) &= R_Y^T(\theta), \\ R_Z^{-1}(\psi) &= R_Z^T(\psi). \end{aligned} \tag{2.5}$$

Rotations can be used in cascade. For example, a rotation about the z-axis in system 1 can be followed by a rotation about the y-axis in system 2 and then about the x-axis in system 3. In fact, any number of rotations can be put together in any order. When all is said and done, the resulting cascade can be reduced to a rotation about just three axis. In flight dynamics, one encounters a specific order of rotation using the Euler angles ψ, θ and ϕ, which represent rotations about z-axis, the new intermediate y-axis and the newer x-axis respectively. The matrix that performs this specific action is called an Euler rotation matrix and has the following definition:

$$T_E^B(\phi, \theta, \psi) = R_X(\phi) R_Y(\theta) R_Z(\psi), \tag{2.6}$$

where the subscript B and the apex E stand for Body and Earth, respectively. The matrix T_E^B resolves an Earth-based vector to body axes. Expanding the matrix products, it results:

$$T_E^B(\phi, \theta, \psi) = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi - S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix}, \tag{2.7}$$

where a shorthand notation, which is $C_\alpha = \cos(\alpha)$ and $S_\alpha = \sin(\alpha)$, has been adopted. In the following, the transformation related to velocity vector is mathematically explained:

$$r_e = \begin{bmatrix} N \\ E \\ D \end{bmatrix}, \tag{2.8}$$

$$v_e = \begin{bmatrix} \dot{N} \\ \dot{E} \\ \dot{D} \end{bmatrix} = \dot{r_e}, \tag{2.9}$$

$$v_b = T_E^B(\phi, \theta, \psi) v_e = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{2.10}$$

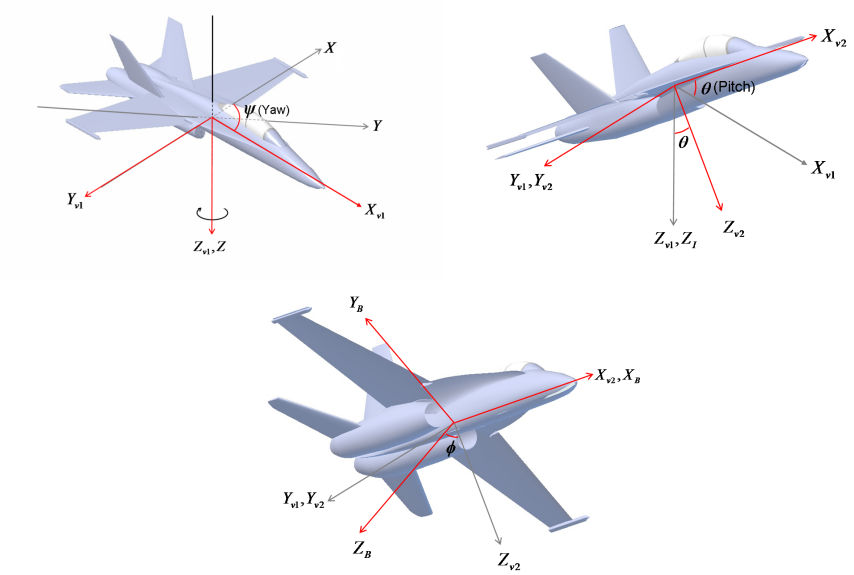


Figure 2.1: Euler angles (from [1]).

where r_e is the position vector of the aircraft centre of gravity in inertial (Earth) axes (the elements are the North, East and Down positions), v_e is the velocity of the aircraft with respect to the Earth and v_b is the inertial linear velocity of the aircraft, resolved to body-axes. The vector of angular position of the aircraft body-axes with respect to the Earth can also be defined, the elements are roll, pitch and yaw angles:

$$\alpha_e = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}. \tag{2.11}$$

Time derivatives of Euler angles

The attitude of an aircraft changes with time when the aircraft manoeuvres. The Euler rates are function of the Euler angles and body-axis angular rate. Euler angles rotate Euler rates into body-axis angular rates. However, the transformation is different from the transformation for linear rates discussed earlier. The process is complicated by the fact that the Euler angles themselves are involved in the transformation from Euler to body-axis rate. The procedure is shown below. The Euler rates are defined as

$$\omega_e = \dot{\alpha}_e = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}, \tag{2.12}$$

and the components of the body angular velocity as

$$\omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.13}$$

To get the body-axis rates from the Earth-axis rates, consider the Euler rates individually, resolve them individually to intermediate axes and then finally to body axes. Define the Euler rate elemental vectors:

$$\omega_{\dot{\phi}} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}, \ \omega_{\dot{\theta}} = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix}, \ \omega_{\dot{\psi}} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}. \tag{2.14}$$

Rotate $\omega_{\dot{\psi}}$ through the angle θ about the y-axis, and add the result to the $\omega_{\dot{\theta}}$ vector. Rotate the sum about the x-axis through the angle ϕ, and add the result to the $\omega_{\dot{\phi}}$ vector. The result is the vector of body-axis angular rates:

$$\omega_b = \omega_{\dot{\phi}} + R_X(\phi)\left[\omega_{\dot{\theta}} + R_Y(\theta)\omega_{\dot{\psi}}\right]. \tag{2.15}$$

After some expansion and rearrangement,

$$\omega_b = E(\phi,\theta)\omega_e = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{2.16}$$

To get the Earth-axis rates in terms of body-axis rates, the transformation matrix E, that doesn't depend on the angle ψ, must be inverted:

$$E^{-1}(\phi,\theta) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix}. \tag{2.17}$$

Unlike the T_E^B matrix, the inverse of E is not the transpose. Worse yet, the inverse is singular at pitch angles of $\pm 90°$. This singularity is called gimbal lock and can be avoided using quaternions.

Quaternions

A quaternion is a four-dimensional representation of a sphere introduced by W. R. Hamilton (1805-1865) that can be used to represent the orientation of a rigid body or a coordinate frame in three-dimensional space (see [14]).

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}, \quad ||q|| = 1. \tag{2.18}$$

The quaternion elements generate the following coordinate transformation:

$$T_E^B = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{2.19}$$

The Euler angles are related to quaternions through these equations:

$$\phi = \arctan\left(\frac{T_E^B(2,3)}{T_E^B(3,3)}\right), \tag{2.20}$$

$$\theta = \arcsin\left(-T_E^B(1,3)\right), \tag{2.21}$$

$$\psi = \arctan\left(\frac{T_E^B(1,2)}{T_E^B(1,1)}\right). \tag{2.22}$$

The quaternion conjugate, denoted by $(\cdot)^*$, can be used to swap the relative frames described by an orientation. Orientation of frame B relative to frame A can be represented by the quaternion q_{AB}, and its conjugate q_{AB}^* describes the orientation of frame A relative to frame B (q_{BA}). Having said this, the following equation can be written:

$$q_{AB}^* = q_{BA} = \begin{bmatrix} q_0 \\ -q_1 \\ -q_2 \\ -q_3 \end{bmatrix}. \tag{2.23}$$

To define compound orientation, the quaternion product ($\otimes$) that can be determined using the Hamilton rule and defined as equation (2.24) can be used. Remember that a quaternion product is not commutative.

$$q_{AC} = q_{BC} \otimes q_{AB} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \otimes \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_1b_1 - a_2b_2 - a_3b_3 - a_4b_4 \\ a_1b_2 + a_2b_1 + a_3b_4 - a_4b_3 \\ a_1b_3 - a_2b_4 + a_3b_1 + a_4b_2 \\ a_1b_4 + a_2b_3 - a_3b_2 + a_4b_1 \end{bmatrix} \tag{2.24}$$

A three dimensional vector can be rotated by a quaternion using the following relationship:

$$v_B = q_{AB} \otimes v_A \otimes q_{AB}^*\ , \tag{2.25}$$

where v_A and v_B are the same vector described in frame A and frame B respectively and each vector contains a 0 insterted as the first element to make them four-elements vector.
The quaternion derivative describing the rate of change of orientation of the Earth frame relative to the body frame can be calculated by the equation

$$\dot{q}_{BE} = \frac{1}{2} q_{BE} \otimes \omega \quad \text{where } \omega = \begin{bmatrix} 0 \\ \omega_b \end{bmatrix}. \tag{2.26}$$

In expanded form:

$$\begin{bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix}. \tag{2.27}$$

2.1.3 Flight dynamics equations

Kinematics

Let S be the position relative to the origin of the centre of gravity of a rigid body that is rotating with angular rate ω_b relative to the Earth frame, that is:

$$S = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \ \omega_b = \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.28}$$

So

$$\frac{\partial S}{\partial t} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.29}$$

The following expressions using dot notation, instead of partial derivative with respect to time, give the total velocity and acceleration in the Earth frame:

$$\frac{dS}{dt} = \frac{\partial S}{\partial t} + \omega_b \times S = \dot{S} + \omega_b \times S, \tag{2.30}$$

$$\frac{d^2S}{dt^2} = \frac{\partial}{\partial t}\left(\frac{\partial S}{\partial t}\right) + \omega_b \times \frac{dS}{dt} = \ddot{S} + \dot{\omega_b} \times S + 2\omega_b \times \dot{S} + \omega_b \times (\omega_b \times S). \tag{2.31}$$

The obtained expressions represent respectively the total linear velocity and acceleration of an object with respect to inertial space as it moves in a frame that rotates with respect to the inertial one. Equation (2.30) is also known as Poisson's formula and can be applied to whatever quantity so as to express it in a non-inertial reference system.

The dynamic equilibrium of the aircraft can be expressed by two vectorial equations:

$$F^{(a)} + F^{(v)} + F^{(i)} = 0, \tag{2.32}$$

$$M^{(a)} + M^{(v)} + M^{(i)} = 0. \tag{2.33}$$

Obviously, for an aircraft in flight, reaction forces $F^{(v)}$ and moments $M^{(v)}$ are null, while in the applied forces $F^{(a)}$ and moments $M^{(a)}$ can be included aerodynamic

and gravitational terms. Inertial forces and moments can be defined in inertial frame as:

$$F^{(i)} = -\frac{dQ}{dt}, \tag{2.34}$$

$$M^{(i)} = -\frac{dK}{dt} - v_P \times Q, \tag{2.35}$$

where $Q = mV_b$ is the momentum and $K = I_n\omega_b$ is the angular momentum, m and I_n refer respectively to mass and inertial tensor and v_P is the linear speed of point P, used as a reference point to compute all moments. In case $P \equiv CG$ then $v_P \times Q = 0$, hence equation (2.32) and equation (2.33) reduce to

$$F^{(a)} = \frac{dQ}{dt}, \tag{2.36}$$

$$M^{(a)} = \frac{dK}{dt}. \tag{2.37}$$

Linear equation of motion

Starting from (2.36) and considering that the momentum is the product of mass and velocity, the following equation can be written:

$$F = \frac{d\left(mV_b\right)}{dt}. \tag{2.38}$$

Applying the Poisson's equation (2.30), obtained in the previous paragraph, and the chain rule to the preceding expression and expanding the derivative we obtain:

$$F = \left(\frac{dm}{dt}\right) V_b + m\left(\frac{\partial V_b}{\partial t} + \omega_b \times V_b\right) = \dot{m}V_b + m\dot{V_b} + \omega_b \times \left(mV_b\right). \tag{2.39}$$

In the case of an UAV, the mass does not change with time, so the first term in equation (2.39) can be neglected. The obtained equation represents the rate of change of linear momentum as a result of applied forces. Those forces are aerodynamics, gravity and others that can be lumped into a category called externally applied loads F_{ext}. The equation of linear motion can be written as:

$$m\dot{V_b} + \omega_b \times \left(mV_b\right) = F_{ext}. \tag{2.40}$$

Angular equation of motion

The definition of the inertia matrix can be introduced

$$I_n = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}, \tag{2.41}$$

where the terms are defined as:

$$I_{xx} = \int_m \left(y^2 + z^2\right) dm, \qquad I_{yy} = \int_m \left(x^2 + z^2\right) dm, \qquad I_{zz} = \int_m \left(x^2 + y^2\right) dm,$$
$$I_{xy} = \int_m (xy)\, dm, \qquad I_{xz} = \int_m (xz)\, dm, \qquad I_{yz} = \int_m (yz)\, dm.$$

For our purposes, usually small scale UAVs show a symmetry both geometrical and in the mass distribution with respect to the same plane. If the body frame is coincident with the symmetry axes of the aircraft body, it follows that equation (2.41) will be diagonal because $I_{xy} = I_{xz} = I_{yz} = 0$.

Starting from equation (2.37), applying the Poisson's formula and definition of angular momentum, the equilibrium of moments can be written as:

$$M^{(a)} = \dot{K} + \omega_b \times K = I_n \dot{\omega_b} + \omega_b \times I_n \omega_b. \tag{2.42}$$

The moments applied to the aircraft on the body axes are L, M and N (roll, pitch and yaw) and can be collected in a vector for all external moments:

$$M_{ext} = \begin{bmatrix} L \\ M \\ N \end{bmatrix}. \tag{2.43}$$

In this way we obtain the angular equation of motion:

$$I_n \dot{\omega_b} + \omega_b \times I_n \omega_b = M_{ext}. \tag{2.44}$$

External forces and moments

The forces and moments acting on the rigid body can be described with these equations:

$$F_{ext} = F_g + F_{props}, \tag{2.45}$$
$$M_{ext} = M_{damp} + M_{props}, \tag{2.46}$$

where F_g is the gravity force, F_{props} and M_{props} are respectively the forces and moments generated by the UAV propellers and M_{damp} is the aerodynamic damping caused by the rotating propellers moving through the air.
The actual geometry of the UAV must be described in order to define the forces and moments that have to be considered and the application point. In Figure 2.2 and Figure 2.3 the configuration, the label of each propeller and its rotation direction are shown for both the quadcopter (in x configuration) and octocopter (in + configuration).

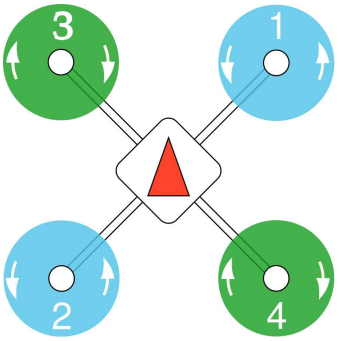


Figure 2.2: Quadcopter configuration (from [2]).

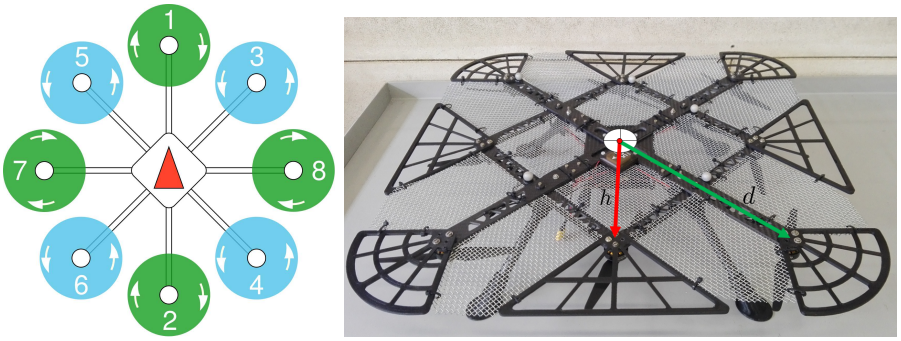


Figure 2.3: Octocopter configuration (from [2]).

The basic principles of rotor aerodynamics state that a rotor produces thrust and torque depending on many aerodynamic quantities. For small scale UAVs, thrust and torque originated from each propeller are modelled in a very simple way, neglecting all aerodynamic effects that can influence rotor blades behaviour during flight.
The basic principle of motion for a small scale UAV is based just on control forces and moments that arise by simply varying each rotor-motor angular speed. For this reason two simple static relationships between each propeller angular speed Ω_i and the corresponding generated thrust and torque can be written:

$$T_i = K_T \Omega_i^2, \quad K_T = C_T \rho A R^2, \tag{2.47}$$
$$Q_i = K_Q \Omega_i^2, \quad K_Q = C_Q \rho A R^3, \tag{2.48}$$

where C_T and C_Q are the thrust and torque coefficients, ρ is the air density, A and R are respectively the area of the propeller disk and its radius, and Ω_i is the rotational speed of the i-th propeller.
For the quadcopter the distance between the centre of gravity and the j-th propeller is b, while for the octocopter there are two different distances: h is the distance from propellers 1, 2, 7, 8 and d is the distance from propellers 3, 4, 5, 6 to the centre of the UAV (Figure 2.3).
Now it is possible to write the equations of the forces and the moments produced

by the propeller for the quadcopter with body frame in the centre defined with x-axis pointing forward (halfway between motors 1 and 3 for quadcopter), y-axis pointing to the right and z-axis downward:

$$F_{4props} = -\begin{bmatrix} 0 \\ 0 \\ K_T\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2\right) \end{bmatrix}, \tag{2.49}$$

$$M_{4props} = \begin{bmatrix} K_T\frac{b}{\sqrt{2}}\left(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2\right) \\ K_T\frac{b}{\sqrt{2}}\left(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2\right) \\ K_Q\left(\Omega_1^2 + \Omega_2^2 - \Omega_3^2 - \Omega_4^2\right) \end{bmatrix}, \tag{2.50}$$

and octocopter (x-axis pointing to motor 1 and same definitions as the quadcopter for y-axis and z-axis)

$$F_{8props} = -\begin{bmatrix} 0 \\ 0 \\ K_T\left(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2 + \Omega_5^2 + \Omega_6^2 + \Omega_7^2 + \Omega_8^2\right) \end{bmatrix}, \tag{2.51}$$

$$M_{8props} = \begin{bmatrix} K_T\left[h\left(\Omega_7^2 - \Omega_8^2\right) + \frac{d}{\sqrt{2}}\left(-\Omega_3^2 - \Omega_4^2 + \Omega_5^2 + \Omega_6^2\right)\right] \\ K_T\left[h\left(\Omega_1^2 - \Omega_2^2\right) + \frac{d}{\sqrt{2}}\left(\Omega_3^2 - \Omega_4^2 + \Omega_5^2 - \Omega_6^2\right)\right] \\ K_Q\left(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2 + \Omega_5^2 + \Omega_6^2 - \Omega_7^2 - \Omega_8^2\right) \end{bmatrix}. \tag{2.52}$$

These forces and moments can be rearranged in order to realize the so called "mixer" matrix χ of the motors, that relates the required thrust and moments around each axis to the rotational speed of the propellers (that are the control inputs of the UAV)

$$\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ -K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} \\ K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} & K_T\frac{b}{\sqrt{2}} & -K_T\frac{b}{\sqrt{2}} \\ K_Q & K_Q & -K_Q & -K_Q \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \chi_4 \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}. \tag{2.53}$$

It follows that the required rotor-motor angular speeds come from:

$$\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \chi_4^{-1} \begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix}. \tag{2.54}$$

For the octocopter the mixer matrix χ_8 is:

$$\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T & K_T & K_T & K_T & K_T \\ 0 & 0 & -K_T\frac{d}{\sqrt{2}} & -K_T\frac{d}{\sqrt{2}} & K_T\frac{d}{\sqrt{2}} & K_T\frac{d}{\sqrt{2}} & K_T h & -K_T h \\ K_T h & -K_T h & K_T\frac{d}{\sqrt{2}} & -K_T\frac{d}{\sqrt{2}} & K_T\frac{d}{\sqrt{2}} & -K_T\frac{d}{\sqrt{2}} & 0 & 0 \\ -K_Q & -K_Q & K_Q & K_Q & K_Q & K_Q & -K_Q & -K_Q \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \\ \Omega_7^2 \\ \Omega_8^2 \end{bmatrix} = \chi_8 \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \\ \Omega_7^2 \\ \Omega_8^2 \end{bmatrix}, \tag{2.55}$$

where being χ_8 rectangular, its inverse matrix is not defined in a traditional sense, but the relation between the motors angular rates and thrust and moments is defined through the pseudo inverse matrix $\chi_8^\dagger$:

$$\begin{bmatrix} T \\ L \\ M \\ N \end{bmatrix} = \chi_8^\dagger \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \\ \Omega_7^2 \\ \Omega_8^2 \end{bmatrix} = \chi_8^T \left(\chi_8 \chi_8^T\right)^{-1} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \\ \Omega_5^2 \\ \Omega_6^2 \\ \Omega_7^2 \\ \Omega_8^2 \end{bmatrix}. \tag{2.56}$$

Moreover, besides the control actions generated by the quadrotor, another external force is represented by gravity. Gravity force is directed downward along the D direction of the Earth frame, so it must be projected on the body axes:

$$F_g = T_E^B(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}. \tag{2.57}$$

The last forces that must be considered are the aerodynamic damping caused by the rotating propellers moving through the air (the aerodynamic drag caused by the structure is neglected because the structure is very thin). Assuming that the UAVs have to fly in near hover condition, the drag produced by linear translations can be neglected. Only the aerodynamic damping proportional to $\omega_b = [\ p\ q\ r\ ]^T$ is considered. Another important approximation is to consider the moments relative to an axis only proportional to the rotational speed around that axis (decoupled moments):

$$M_{damp} = \begin{bmatrix} \frac{\partial L}{\partial p} & 0 & 0 \\ 0 & \frac{\partial M}{\partial q} & 0 \\ 0 & 0 & \frac{\partial N}{\partial r} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.58}$$

The derivatives in the previous matrix are called stability derivatives and they have an analytical form, which is taken from helicopter dynamics background

[15]. Because of the geometry of the UAV, in quadcopter case, $\frac{\partial L}{\partial p} = \frac{\partial M}{\partial q}$, the derivatives can be written as follows:

$$\frac{\partial L}{\partial p} = -4\rho A R^2 \Omega^2 \frac{\partial C_T}{\partial p} \frac{b}{\sqrt{2}}, \qquad \frac{\partial C_T}{\partial p} = \frac{C_{L_\alpha}}{8} \frac{\sigma}{R\Omega} \frac{b}{\sqrt{2}}, \tag{2.59}$$

where σ is the solidity ratio defined as the ratio between the total blade area and the disk area A, C_{L_α} is the slope of the thrust coefficient that, because the propeller's blade is thin and the actual aerofoil is not known, can be assumed equal to the one of the flat plate (2π). Due to the slow rotational speed about the z-axis with respect to the other two axes, its damping moment can be neglected, which means $\frac{\partial N}{\partial r} = 0$.

2.1.4 Notation

In this section the sign conventions and notation used in the rest of the work are presented. In particular the origin of the NED reference frame is located on the ground at the centre of the cage in which the experiments are conducted, the North axis points forward, the East axis points rightward and the Down axis downward. Using this convention when the UAVs are flying, the Down position will be negative. The same sign convention are applied for velocity and acceleration: in particular when the UAV is moving up from the floor it has negative Down velocity $\dot{D}$, while when it descends it has a positive velocity, the downward acceleration is positive, while the upward acceleration is negative.
Moreover all relative quantities are defined as target drone measurements minus follower drones measurements, so relative position and velocity for example along the Down axis, are defined in this way:

$$D_r = D_t - D_f, \tag{2.60}$$

$$\dot{D}_r = \dot{D}_t - \dot{D}_f, \tag{2.61}$$

where the subscript $(\cdot)_r$ refers to relative quantities, $(\cdot)_t$ to target and $(\cdot)_f$ to follower.
We will refer to estimated quantities with the hat symbol $\hat{(\cdot)}$. Finally, the set-points given to a drone are identified by the $(\cdot)^o$ apex.

2.2 Previous work

The objective of this work is to perform autonomous landing of a small UAV on a larger one which is moving vertically with an oscillating motion or in the plane along a circular trajectory, simulating a loiter condition.
Each of the two problems presents its own critical aspects. Both of them are clearly three-dimensional problems but, for the sake of simplicity, are decoupled:

the first part deals with trajectories synchronization in the horizontal plane, while the second concerns the descent of the follower drone on the target one.
Therefore the landing objective is:

$$\lim_{t \to t_f} \left(\begin{bmatrix} N_t(t) \\ E_t(t) \\ D_t(t) \end{bmatrix} - \begin{bmatrix} N_f(t) \\ E_f(t) \\ D_f(t) \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ \epsilon_D \end{bmatrix}, \tag{2.62}$$

where t_f is the time at which the procedure is terminated and ϵ_D is a positive defined tolerance.
As described in Chapter 1, the starting point is the work in Giuri and Marini Cossetti's master thesis [13]. In that work the problem of landing a small quadcopter on another UAV while the latter is hovering has been analysed. The proposed control structure consists of: a motion monitoring module, a trajectory generation module and a tracking control module.
In the previous work, the relative navigation problem has not been studied; the absolute positions of the drones were known exactly by means of a motion capture system. The motion monitoring module only manages the feedback position measurements and checks the safety of the whole procedure (the in-plane relative distance must be smaller than a certain value so as to perform landing).
The state measurements and the safety variables feed the trajectory generation module where the desired set-point is generated, such that the follower converges from the initial height to the target. These two modules run on the ground control station.
The tracking control module consists of the built-in controllers of the adopted flight control units, capable of tracking a given position and yaw reference with good performance in the case of a hovering target.
The proposed architecture presents some limitations: first of all the in-plane synchronization was performed using the absolute position measurement of the two drones, while in a more general non-collaborative case (when the follower knows only the relative distance from the target) these measurements are not available, moreover to perform a sinusoidal landing an estimate of the vertical motion of the target is needed as suggested by [7]. Finally the synchronization while the target is moving represents a challenging problem because the wakes of the follower's rotors disturb the target to the point that the latter is no longer able to follow its trajectory. For this reason the tracking control module has to be improved.

2.3 Overall control architecture

In this thesis a control architecture capable of performing an autonomous landing of an UAV on another one, while the second is flying, is proposed in two different conditions.
Two problems are investigated separately. The first one deals with the target

drone that is moving in the plane following a circular trajectory at constant speed. The second one deals with the target that is moving out of the plane, following a sinusoid in the vertical direction.
The proposed structure consists of three main modules (as shown in Figure 2.4), as in the previous works [7], [8], [9] and [13], but with some improvements to make the landing possible also in the two conditions described before.

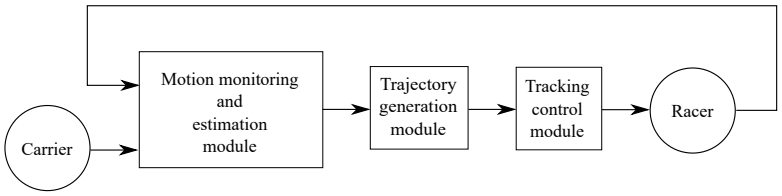


Figure 2.4: Overall control architecture: three modules.

2.3.1 Tracking control module

The tracking control module is the same for the two problems and consists in the built-in controllers running on the Flight Control Units (FCUs) of follower and target. They are capable of tracking a given position and a yaw reference (control variables) with good performance. The UAVs feature a nested control loops architecture: an outer position control loop generates the thrust set-point (f_c) and the attitude set-point (q^o) for the inner attitude control loop.
The position control loop is based on an inner Proportional-Integral-Derivative (PID) loop for the linear velocity and an outer Proportional (P) loop for the position itself. The attitude controller instead generates the required moments set-point (τ_c) to achieve attitude tracking, this controller is also based on an inner PID loop for the angular velocity and a P loop for the angular attitude angle.
Finally a control allocator (Mixer block) is defined according to the multirotor frame/configuration in order to convert the force/moments set-points into propeller angular velocity commands. A module for position, velocity and attitude estimation running on the FCU provides then the state estimates which will be used as a feedback for the control loops described above.
The design of an ad hoc controller for this task is beyond the goal of this thesis, but, as previously said, while the racer is flying over the carrier, aerodynamic interaction between lander and carrier leads to very significant perturbations in the carrier motion, so the gains of position control loop of both drones have been retuned making control more aggressive to achieve better performances. This has been done through model identification and simulation. The detailed procedure will be presented in Chapter 3.

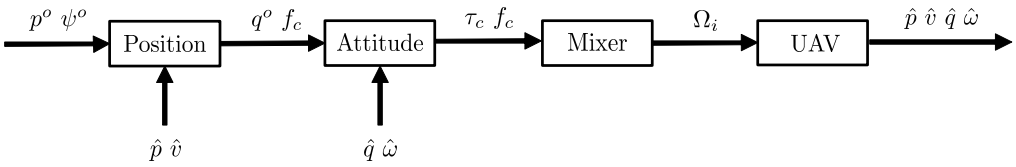


Figure 2.5: Tracking control module.

2.3.2 Motion monitoring and estimation module

The motion monitoring module manages the feedback position measurements and ensures the safety of the whole procedure. In particular in the case of landing with carrier that is moving in the plane the motion capture system is able to give position and velocity measurements used to check the safety of the whole process, while in the case of the sinusoidal landing an online estimation of the target motion is necessary. The latter problem will be analysed in Section 5.2.
The most important safety constraint is the in-plane synchronization. This is obtained with the simple and the augmented control position described in Section 4.2.2 and Section 4.2.1. Only when the follower is in the neighbourhood of the target position, the landing manoeuvrer can start. This constraint must be verified for the entire duration of the landing otherwise the descent is stopped until repositioning and synchronization are re-obtained. Moreover, since the target may voluntarily vary its altitude or experience vertical oscillations caused by the aerodynamic perturbations originating from the follower rotor's wake, continuous feedback of the relative vertical distance as well is used to design real-time reference landing trajectory.

2.3.3 Trajectory generation module

The trajectory generation module has to generate the follower trajectory so as to perform first of all the synchronization in the plane and finally the landing on the target. To do this, it uses informations given by the motion monitoring and estimation module.
The trajectory for the in-plane synchronization is generated in two different ways: the first one is related to a collaborative case, in which the follower knows the exact position of the target and synchronization is performed giving to the follower the target position as set-point (simple position control, Section 4.2.1); the second one is related to a non-collaborative case where only the in-plane relative distance is known and starting from it, the follower acceleration set-point is computed through an augmented position control presented in Section 4.2.2. This

acceleration is integrated twice to obtain the set-point position. This can be interpreted as a feedback loop on the follower position, where the control variable is the follower acceleration set-point; this control loop is at a higher level with respect to the position controller implemented in the tracking control module.
The trajectory for the vertical approach (landing) is computed using the bang-zero-bang algorithm [16], with different features (bisection method and check on the estimate quality) in the case of vertical oscillating target. Also in this case the acceleration computed through the algorithm is integrated twice to obtain the position set-point for the racer, starting from measurements of the motion monitoring and estimation module.
The detailed description of trajectory generation process is presented in Section 4.3 and Section 5.3.

Chapter 3

Position and attitude controller tuning

While the follower is flying over the target, aerodynamic interactions lead to very significant perturbations in the target's motion, making the landing with moving target dangerous. For this reason a more aggressive controller is necessary to achieve better performance in terms of trajectory tracking. The gains of the position and attitude control laws have been tuned through model identification and simulation. In this chapter the procedure for tuning the control laws for the two drones is explained. In the first part the procedure for the model dynamics identification is explained, in the second part the tuning of vertical position, attitude and horizontal position controllers of the target are presented. Finally, in the third part, the tuning of the in-plane position controller for the follower is shown.

3.1 Identification

Dynamic models of the two drones are needed so as to tune the controllers. To do this, model identification is applied. In particular black-box models of the target vertical dynamics, the target pitch dynamics and the roll dynamics of the two drones have been identified with the Predictor Based Subspace Identification PBSID algorithm [17] using closed-loop experimental data.
A Pseudo Random Binary Sequence (PRBS) is given as input in different experiments, for: target thrust, target roll, target pitch and follower roll moments. Through PBSID the following four models are obtained:

- model from the thrust input T to the vertical velocity output $\dot{D}$ of the target, that is of order two and contains a time delay of four sampling periods,
- model from the roll moment input L to the roll rate output p of the target, that is of order two with a time delay of four sampling periods,

- model from the pitch moment input M to the pitch rate output q of the target, that is of order four with a time delay of five sampling periods,
- model from the roll moment input L to the roll rate output p of the follower, that is of order four and contains a time delay of four sampling periods.

Figure 3.1 shows the coherence of the non-parametric frequency response function (FRF) in both open and closed-loop case for the target roll attitude identification experiment as an example: it indicates over which frequency range the experimental data is valid (in this example the frequency validity range is 5-50 rad/s).

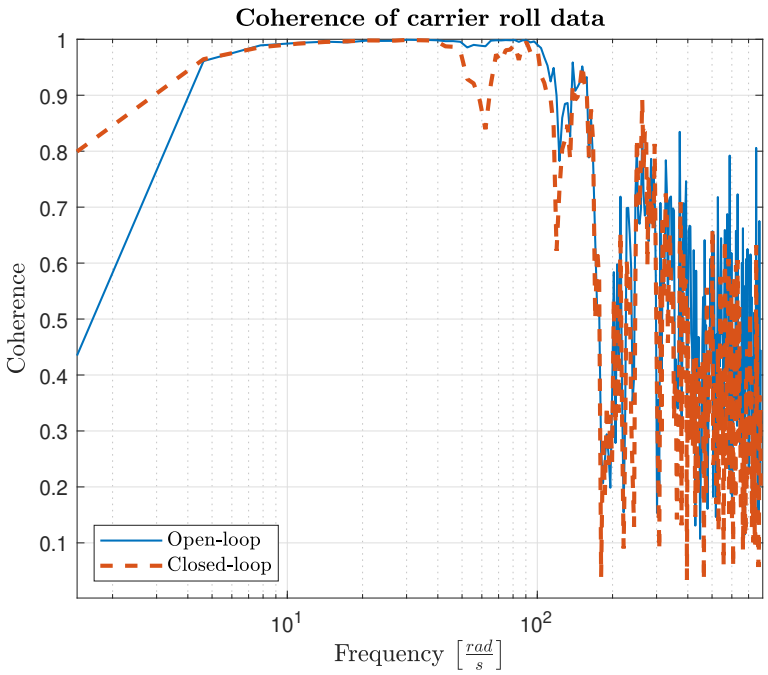


Figure 3.1: Coherence function of open-loop and closed-loop non-parametric frequency response function of target roll dynamics.

Figure 3.2 shows the measured roll rate response against the simulated closed-loop roll rate response obtained with the identified model; the close match to the measured data is clear. This is also evident from values of the percentage of the Variance Accounted For ([17]) reported in Table 3.1 calculated with the a dataset used only for validation. In all cases the VAF is calculated with filtered measured and simulated output. In particular a band-pass filter is used to eliminate bias, the bandwidth for each identification is: 10-100 rad/s for the follower roll dynamics, 5-50 rad/s for the target roll and pitch dynamics and 0.5-20 rad/s for the target position dynamics.

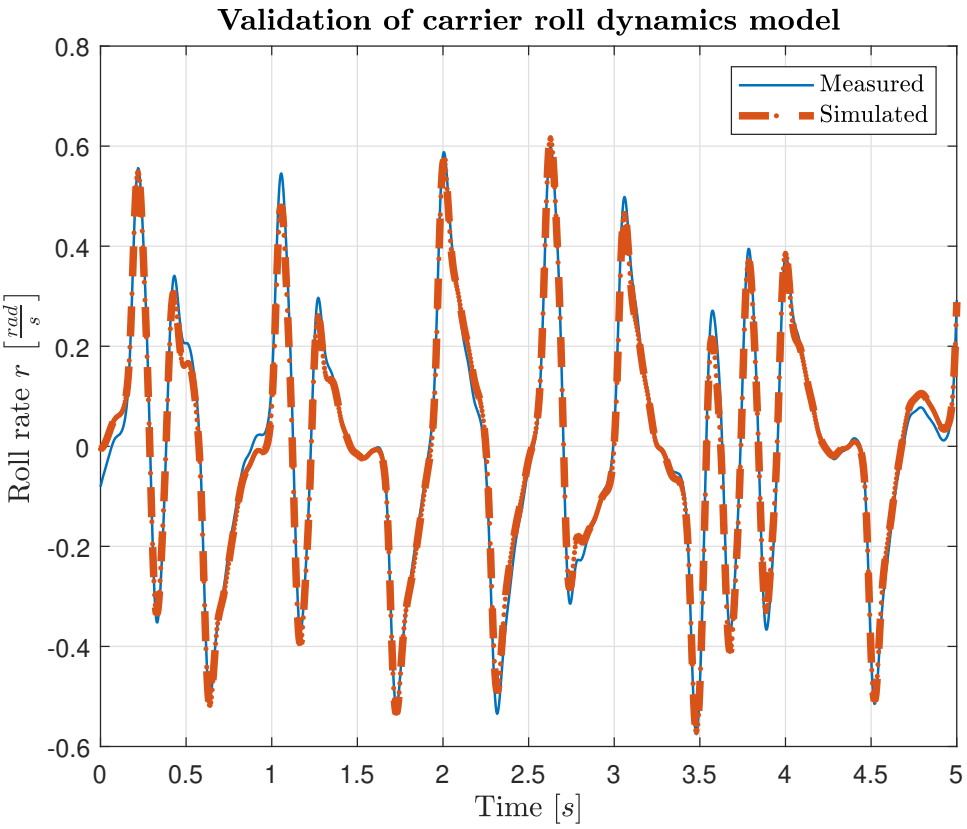


Figure 3.2: Validation of closed-loop identified model of target roll dynamics.

Model	Validation VAF [%]
Target vertical dynamics	95.7
Target roll dynamics	98.2
Target pitch dynamics	97.45
Follower roll dynamics	87.2

Table 3.1: VAF of the identified models.

3.2 Carrier tuning

In this section the procedure to tune the controllers of the carrier drone is described. In particular starting from experiment, it has been noticed that, when the racer flies over carrier, the aerodynamic perturbations make the carrier unable to follow its trajectory: it is not able to maintain the set-points for vertical and in-plane position.

3.2.1 Vertical position control

In order to characterize the amount of disturbance introduced by the aerodynamic interaction of the follower flying over the carrier, an experiment with carrier in hovering is performed in two situations: undisturbed and disturbed by the inflow of follower above it. The Power Spectral Density (PSD) of the normalized thrust of the carrier is computed in both conditions. Normalized thrust is defined as the set-point command given to rotors normalized with respect to the maximum value, so it has a value between 0 and 1 [2]. The idea is to quantify the amount of control action induced by the disturbance with respect to the baseline undisturbed case. This information is used as a requirement in control law design of the vertical position loop of carrier to better reject the disturbance.
Ten seconds time windows for each condition have been selected avoiding transient phases. From the trimmed signals the mean value is removed, because the interest is on variation with respect to the hover condition. In Figure 3.3 the normalized thrust variation in the two conditions is shown.

The PSD is computed using the MATLAB function pwelch. Figure 3.4 shows the results. It can be noticed that the low frequency content of PSD increases in the case with the disturbance, a peak at 0.4 Hz (2.5 rad/s) is present. This frequency value is used as a requirement for position control law tuning, *i.e.*, it represents a lower limit on the bandwidth of the vertical velocity loop.

At this point the model of the vertical position dynamics is used to retune the gains of the vertical position controller. As stated in Section 2.3.1, the controller is given by an outer P loop on the vertical position and an inner PID loop on the vertical velocity. Taking into account the result obtained studying the thrust PSD in disturbed and undisturbed conditions, since the inner loop is directly affected by the thrust disturbance caused by the follower, only its gains are retuned with a loop shaping approach: to increase the bandwidth with respect to the baseline solution (Table 3.2), a higher cut-off frequency of 10 rad/s is chosen as requirement; then since the open loop transfer function presents two poles, one at high frequency and the other one at low frequency, the two zeros of the regulator are chosen one at high frequency $z_1 = 30$ rad/s to cancel the open loop dominant pole, thus obtaining a slope of -20 dB/dec in the neighbourhood of the desired

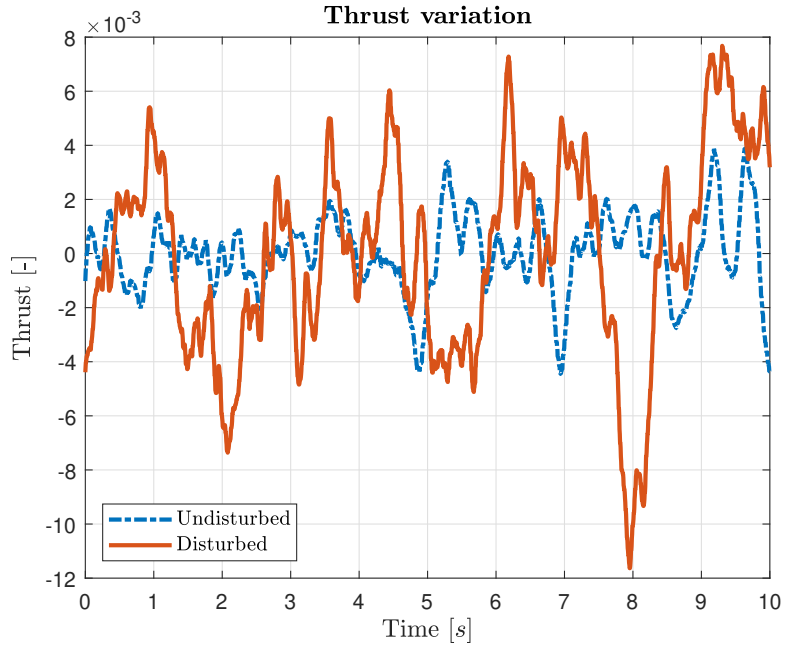


Figure 3.3: Carrier normalized thrust variation with time in undisturbed and disturbed condition.

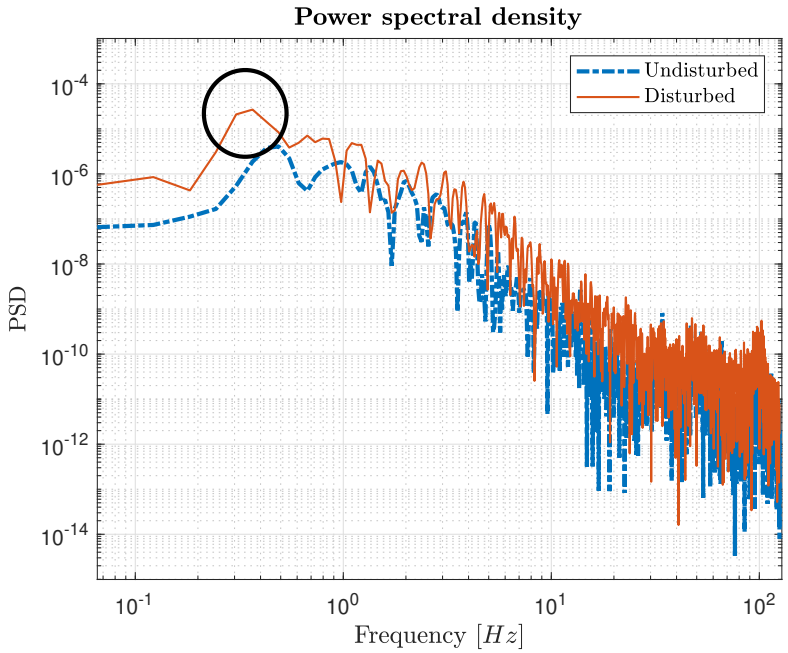


Figure 3.4: PSD of the carrier normalized thrust variation in undisturbed and disturbed condition.

bandwidth, and one at low frequency $z_1 = 0.1\ rad/s$ to increase the phase. The gains are computed with the following equations:

$$k_p = k\,(z_1 + z_2)\,, \tag{3.1}$$

$$k_i = k\ z_1\ z_2, \tag{3.2}$$

$$k_d = k, \tag{3.3}$$

where the controller gain k in this case is selected in order to obtained the desired cut-off frequency.
In Table 3.2 the baseline and retuned gains values are reported, with the values of the inner loop phase margin and cut-off frequency. It must be noted that with the new values we obtain greater phase margin and cut-off frequency, then a more aggressive control able to better reject disturbance. The gain values obtained are then tested in flight.

Values	Outer loop	Inner loop			Phase margin	Cut-off frequency
	k_p	k_p	k_i	k_d		
Baseline	1	0.2	0.02	0	70.3 deg	6.46 rad/s
Retuned	1	0.3	0.03	0.01	75.5 deg	9.96 rad/s

Table 3.2: Carrier vertical position loop gains.

3.2.2 Attitude control

During the experiment with target in hovering in undisturbed condition, the pitch and roll control actions show a significant amount of high frequency content. This is probably due to a too large derivative gain in the angular rate loop: the noise on the gyroscope angular rate measurements passes through the derivative action and it results in a noisy control action. To reduce this control action the derivative gain k_d of the inner pitch attitude control has been reduced (Table 3.3), in this way the pitch control moment generated is less sensitive to noise on pitch rate even if a slight decrease in performance is observed in terms of phase margin and bandwidth.
It must be noted that, as the thrust, also the moments are normalized quantities, the only difference is that they may assume value between -1 and 1.

In fact with the new value of the derivative gain the phase margin and the cut-off frequency (*i.e.*, the bandwidth) are slightly reduced (as can be seen in Table 3.4), but the Root Mean Square (RMS) of control action is significantly reduced as shown in Figure 3.5 and Table 3.5.

In Table 3.4, Table 3.5 and Figure 3.5 the results obtained adopting the same strategy for roll control action are shown.

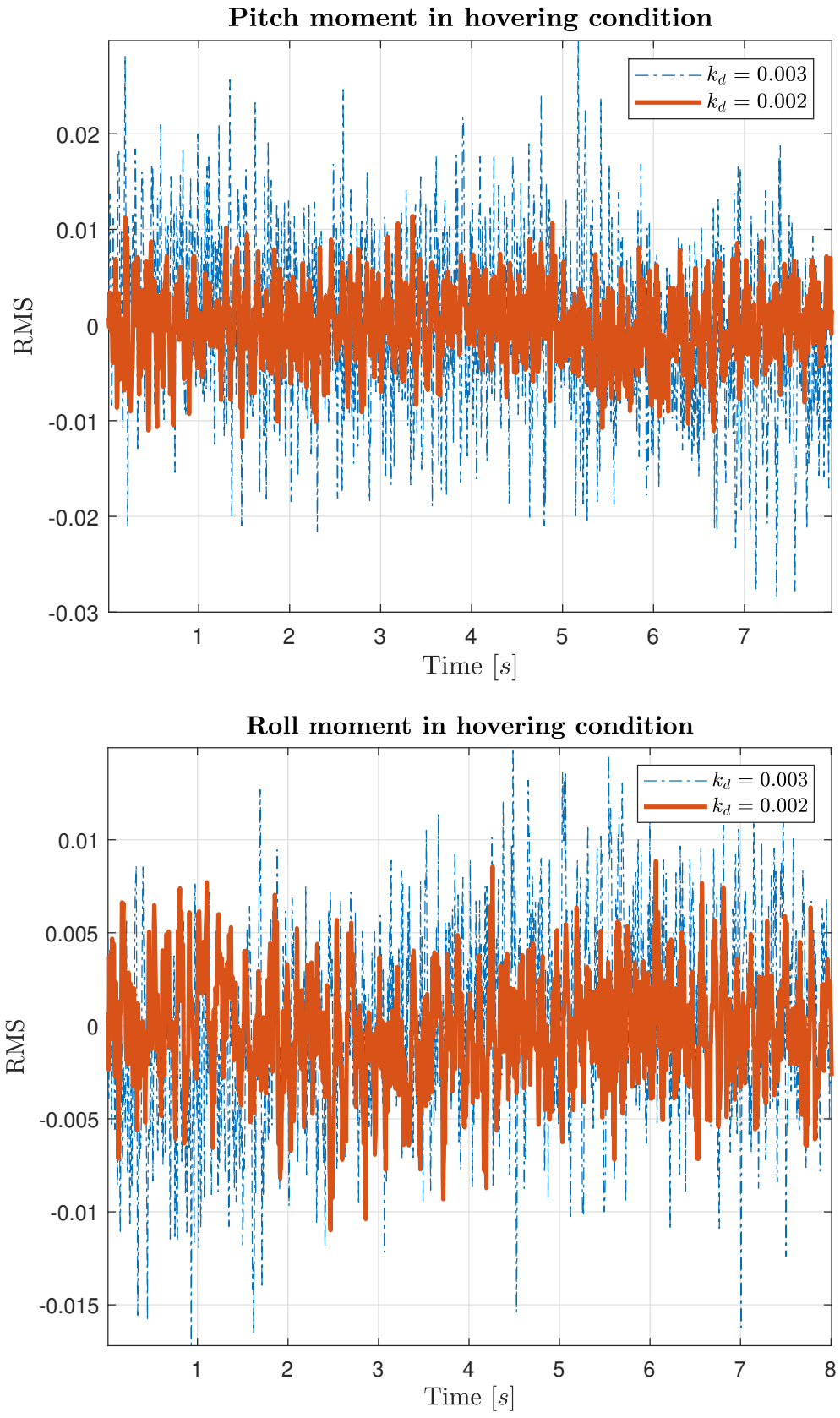


Figure 3.5: Pitch and roll normalized moment.

Values	Outer loop	Inner loop		
	k_p	k_p	k_i	k_d
Baseline	6.5	0.15	0.05	0.003
Retuned	6.5	0.15	0.05	0.002

Table 3.3: Target pitch and roll attitude loop gains.

Gain values	Phase margin $[deg]$		Cut-off frequency $[rad/s]$	
	Baseline	Retuned	Baseline	Retuned
Pitch	49.3	44.9	48.5	42.2
Roll	49.6	47.9	53.4	44.4

Table 3.4: Phase margin and cut-off frequency value of inner attitude loop of the target for pitch and roll.

3.2.3 In-plane position control

After tuning the vertical position control, we also tune the in-plane position control in order to let the target better follow its trajectory also when it is disturbed by the follower. In particular new values, that make the control more aggressive, of the proportional gain of the outer P loop for the position and the proportional, integral and derivative gains of the inner PID loop for the v_x and v_y body-axis velocities are obtained. Starting from a baseline solution for the values of the in-plane position and velocity controller gains reported in Table 3.6, the model of the position and velocity closed loop system is obtained. In the system the identified model of the pitch attitude for the velocity v_x and of the roll attitude for the velocity v_y is closed in loop using the retuned values of the attitude controller reported in Table 3.3.

The in-plane velocity loop was tuned by means of loop shaping approach: as a requirement, the bandwidth of the inner (velocity) loop is imposed to be $\omega_{bw_i} = 4.5\ rad/s$ on both axes because a symmetric behaviour is expected, then, for the same reason, to tune the inner loop PID on both velocities two zeros are used: the first one at high frequency, $z_1 = 10\ rad/s$, to cancel the dominant pole of the attitude dynamics, the second one at low frequency, $z_2 = 0.1\ rad/s$, to introduce phase lead at low frequency and to decrease the slope of the closed loop transfer

Gain values	RMS Baseline	RMS Retuned
Pitch	$8.7\ 10^{-3}$	$3.8\ 10^{-3}$
Roll	$5\ 10^{-3}$	$3\ 10^{-3}$

Table 3.5: Phase margin and cut-off frequency value of inner attitude loop of the target for pitch and roll.

function magnitude from $-40\ dB/dec$ to $-20\ dB/dec$. The gains are computed with the following equations:

$$k_p = \frac{\omega_{bw_i}}{z_1\ g}\ \gamma_t\left(z_1 + z_2\right), \tag{3.4}$$

$$k_i = \frac{\omega_{bw_i}}{z_1\ g}\ \gamma_t\ z_1\ z_2, \tag{3.5}$$

$$k_d = \frac{\omega_{bw_i}}{z_1\ g}\ \gamma_t, \tag{3.6}$$

where $g = 9.81\ m/s^2$ is the acceleration of gravity and $\gamma_t = 0.32$ is a scaling factor of the thrust defined as the inverse of the hovering thrust because PX4 velocity controller computes a thrust that is normalized, so this scaling factor is in cascade downstream the velocity controller (see [18]).
Once the inner loop has been tuned, the outer P loop is simply tuned increasing the cut-off frequency up to a value that is one third of the ω_{bw_i} to ensure frequency separation; so $k_p = \omega_{bw_o} = 1.5\ rad/s$.

Values	Outer loop	Inner loop		
	k_p	k_p	k_i	k_d
Baseline	0.95	0.09	0.02	0.01
Retuned x and y	1.5	0.15	0.015	0.015

Table 3.6: Carrier in-plane position and velocity loop gains.

In Table 3.7 we notice that with retuned gains values, slightly lower phase margin of the inner (velocity loop) transfer function with respect to the baseline case, together with a significant increase in the cut-off frequency are obtained. This reduces the settling time to step response making control action more aggressive. The outer (position loop) transfer function with the retuned gain has greater phase margin and cut-off frequency with respect to the baseline case (Table 3.7), this could lead to the conclusion that we have an improvement both in stability properties and settling time. This is not true, in fact this result is influenced also by the inner loop new parameters: using the new values of inner (velocity) loop and the default value for the outer (position) loop we obtain phase margins of 71.9 deg in x and 72 deg in y greater with respect to the new inner-new outer case but lower cut-off frequencies 0.966 rad/s in x and 0.965 rad/s in y, this means high settling time and so slow response.
In Figure 3.6 the faster simulated step response given with retuned gains values can be seen for the y case, analogous results are obtained in x.

3.2.4 Position control performance evaluation

After tuning the vertical position, the attitude and the in-plane position controller of the carrier the performance in undisturbed and disturbed conditions can be

Gain values	Phase margin $[deg]$		Cut-off frequency $[rad/s]$	
	Inner	Outer	Inner	Outer
Baseline x	77.4	62.4	2.7	1
Retuned x	74.7	70.9	4.38	1.49
Baseline y	78.3	62.5	2.72	0.998
Retuned y	75.7	71.1	4.44	1.49

Table 3.7: Phase margin and cut-off frequency value of inner and outer loop position transfer functions for carrier.

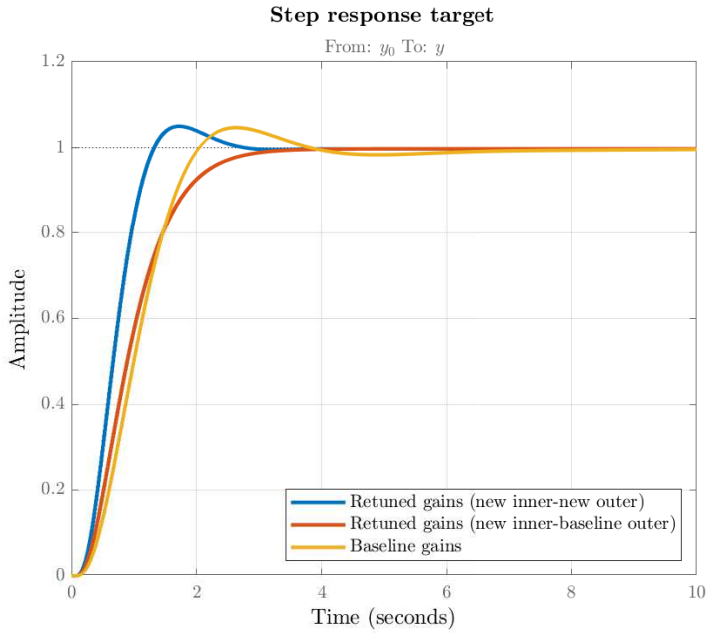


Figure 3.6: Comparison of the simulation to step response of in-plane position dynamics of the target in y.

Condition	RMS $[m]$	Peak error $[m]$
Undisturbed	0.0121	0.0318
Disturbed	0.0396	0.1020

Table 3.8: Root mean square and peak of the in-plane position error with respect to the set-point in disturbed and undisturbed conditions.

evaluated. In particular, in Figure 3.7 the position of the carrier and the error with respect to the set-point position in the two conditions, for an experiment in which an oscillating set-point in the vertical direction was given to the carrier, are presented. It is clear also from the values of the root mean square and of the peak of the error with respect to the position set-point (Table 3.8) that disturbance affects the controller performance despite the fact that we improve it with respect to the baseline case.

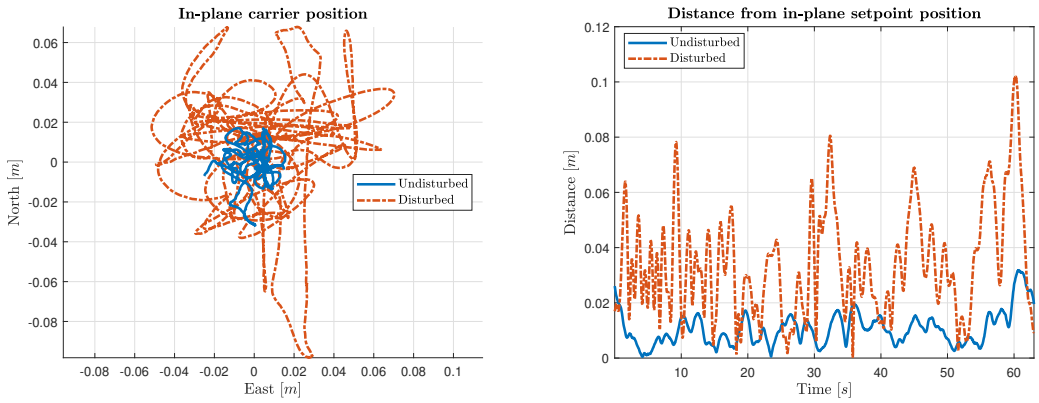


Figure 3.7: Carrier in-plane position and error with respect to set-point in disturbed and undisturbed conditions.

3.3 Racer tuning

Also for the racer we proceed to retune the in-plane position control to improve synchronization with the in-plane target position. In particular new value of the proportional gain of the outer P loop for the position and the proportional, integral and derivative gains of the inner PID loop for the velocity are obtained. The in-plane symmetric behaviour is considered, hence the procedure is implemented for the y loop and the same values of the gains are used also for the x loop. Starting from baseline values for the in-plane position and velocity controller gains reported in Table 3.10 the baseline position and velocity closed loop system is obtained. In the system the identified model of the roll attitude is closed using the baseline values of the attitude controller reported in Table 3.9. As a requirement, the

	Outer loop	Inner loop		
	k_p	k_p	k_i	k_d
Roll	6.5	0.05	0.05	0.001
Pitch	6	0.07	0.05	0.001

Table 3.9: Racer baseline attitude loop gains.

bandwidth of the inner (velocity) loop is imposed to be of $\omega_{bw_i} = 3\ rad/s$, then to tune the inner loop PID on velocity two zeros are used: the first at high frequency, $z_1 = 20\ rad/s$, to cancel the pole of the attitude dynamics, the second at low frequency, $z_2 = 0.1\ rad/s$, to introduce phase lead at low frequency and to decrease the slope from $-40\ dB/dec$ to $-20\ dB/dec$. The gain are computed with the same equations used for the target but with new values:

$$k_p = \frac{\omega_{bw_i}}{z_1\ g}\ \gamma_f\ (z_1 + z_2)\,, \tag{3.7}$$

$$k_i = \frac{\omega_{bw_i}}{z_1\ g}\ \gamma_f\ z_1\ z_2, \tag{3.8}$$

$$k_d = \frac{\omega_{bw_i}}{z_1\ g}\ \gamma_f, \tag{3.9}$$

where $g = 9.81\ m/s^2$ is the acceleration of gravity and $\gamma_f = 0.3$ is a scaling factor of the thrust defined as the inverse of the hovering thrust because of normalization (see [18]).

Values	Outer loop	Inner loop		
	k_p	k_p	k_i	k_d
Baseline	0.95	0.06	0.02	0.01
Retuned	1.5	0.09	0.009	0.005

Table 3.10: Racer in-plane position and velocity loop gains.

In Table 3.11 we notice that with retuned gains higher phase margins of the inner and outer transfer functions with respect to the baseline case are obtained, together with an increase in the cut-off frequencies. This reduces the settling time to step response making control action more aggressive with a slight loss in robustness properties (the overshoot is greater) as can be seen in Figure 3.8.

3.4 Conclusions

In Tables 3.12 and 3.13 all control gains retuned are listed, the ones that are not present have the default values of [18].

Values	Phase margin $[deg]$		Cut-off frequency $[rad/s]$	
	Inner	Outer	Inner	Outer
Baseline	64.5	57.5	1.94	1.06
Retuned	71.2	61.5	2.89	1.48

Table 3.11: Phase margin and cut-off frequency value of inner and outer loop position transfer functions for racer.

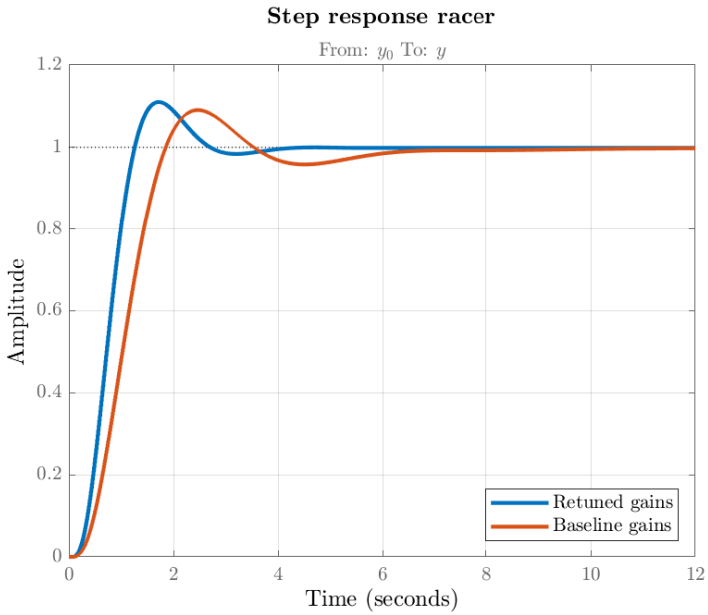


Figure 3.8: Comparison of the simulated step response of in-plane position dynamic of the racer in y direction.

	Outer k_p	Inner k_p	Inner k_i	Inner k_d
In-plane position x and y	1.5	0.15	0.015	0.015
Vertical position z	1	0.3	0.03	0.01
Pitch and roll attitude	6.5	0.15	0.05	0.002

Table 3.12: Carrier retuned control gains.

	Outer k_p	Inner k_p	Inner k_i	Inner k_d
In-plane position x and y	1.5	0.09	0.009	0.005

Table 3.13: Racer retuned control gains.

Chapter 4

In-plane landing

In this chapter the procedure for the landing with target moving on a circular trajectory is described. The chapter is divided in four parts: in the first one the problem is presented in general terms, in the second one two approaches to obtain the in-plane synchronization are formulated, in the third one the algorithm for the landing is presented and in the last part simulation results obtained in MATLAB and Simulink environments are finally shown.

4.1 Problem description

The objective of this work is to perform autonomous landing of the follower on the target while both are flying, in particular while the target is moving along a circular trajectory with constant velocity. This is done in order to simulate the possibility of landing a small multirotor UAV on a fixed-wing drone that has greater endurance than the multirotor target and might be more suitable for outdoor applications. Being fixed-wing UAV unable to hover, loiter at low velocity is the easiest condition so as to land a small drone on it; the problem with the target moving on a circular trajectory is the most similar to loiter and for this reason it is investigated.
The problem is clearly three-dimensional but, for the sake of simplicity, it is decoupled: the first part deals with synchronization of the trajectories of the two UAVs in the horizontal plane, while the second one concerns the descent of the follower drone on the target.
In-plane synchronization is the first safety constraint. Only when the follower is in the same in-plane position as the target, or in its neighbourhood, the landing manoeuvre can start. This constraint must be verified for the entire duration of the descent, otherwise the landing procedure is paused until the constraint is verified again. The safety constraints do not concern just in-plane synchronization. As a matter of fact, continuous feedback of the vertical relative position is used to design the real-time reference landing trajectory. The main reason of this approach

is that the target may vary its altitude or experience vertical oscillations caused by the aerodynamic perturbations, originating from the follower's rotors wake.
Obviously the follower must be more reactive and faster than the target to perform synchronization, otherwise the former is not able to follow and to land on the latter.
In the problem, limitations on performance of the follower are also considered. In particular constraints on maximum horizontal and vertical velocities, together with maximum in-plane acceleration, are taken into account during the whole procedure.
The two parts of the landing procedure, namely in-plane synchronization and landing algorithm, are described in detail respectively in Section 4.2 and Section 4.3.

4.2 In-plane synchronization

The objective of the in-plane synchronization procedure is that the follower, that starts in a different position with respect to the target, must be able to reach and continuously be in the neighbourhood of the target in-plane position, expressed into North and East coordinates.
Defining the in-plane position error in NED frame as

$$H_{tf}(t) = \begin{bmatrix} N_t(t) \\ E_t(t) \end{bmatrix} - \begin{bmatrix} N_f(t) \\ E_f(t) \end{bmatrix} = \begin{bmatrix} N_r(t) \\ E_r(t) \end{bmatrix} = \begin{bmatrix} e_N(t) \\ e_E(t) \end{bmatrix}, \tag{4.1}$$

the safety objective can be expressed as

$$H_{tf}(t) = \sqrt{e_N^2(t) + e_E^2(t)} \leq \epsilon_{NE}(D_r), \tag{4.2}$$

where D_r is the relative vertical distance computed as the difference between the target and the follower Down positions. In equation (4.2) ϵ_{NE} represents the horizontal tolerance on the in-plane position error; when the error is within this tolerance, landing over the target is considered safe. This horizontal tolerance is defined as a function of the relative vertical distance, in order to restrict the safety area while the UAVs are approaching, in this way:

$$\epsilon_{NE}(D_r) = m_s D_r + q_s, \tag{4.3}$$

where the suitable coefficients are selected to be $m_s = 0.067$ and $q_s = 0.2\ m$ in order to obtain a circular safety area with radius of $0.2\ m$ at zero vertical relative distance and with radius of $0.3\ m$ at $1.5\ m$ of relative vertical distance. In this way the follower can land as close as possible to the centre of gravity of the target so as not to create high moments on it. A truncated cone defines the safety area for landing as can be seen in Figure 4.1.

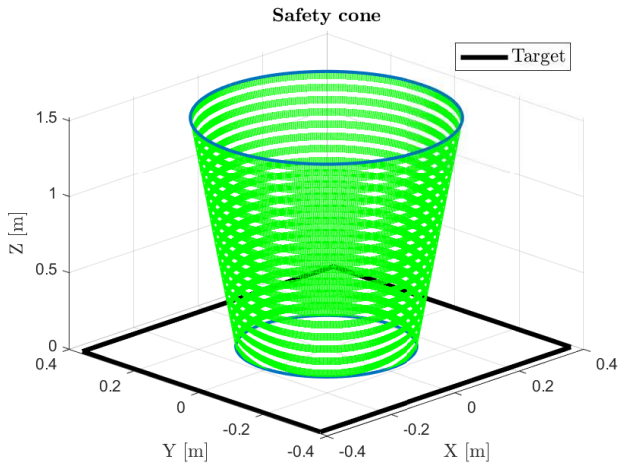


Figure 4.1: Safety cone.

The problem of in-plane synchronization has been studied in two different conditions: collaborative and non-collaborative cases.
In the collaborative case it is supposed that the target drone is able to communicate its position and velocity to the follower, hence simplifying the synchronization that can be performed with a simple position control architecture in which the target position is directly used as set-point for the follower trajectory tracking control module.
In the non-collaborative case the follower does not know the absolute position of the target, but starting only from relative position and velocity measurements is able to compute the follower position set-point to obtain synchronization, through a feed-back control law.

4.2.1 Simple position controller

Supposing to know the exact absolute position of the target (collaborative case), the control law for synchronization uses the current North and East positions of the target as the position set-point for the tracking control module of the follower as outlined in Figure 4.2 and in equation (4.4) and equation (4.5):

$$N_f^o(t) = N_t(t), \tag{4.4}$$

$$E_f^o(t) = E_t(t). \tag{4.5}$$

It is clear that in this way there is no control on the velocity and acceleration limits of the follower. In the worst cases, when the relative distance is large, the follower tracking control module receives an input that can cause dangerously high

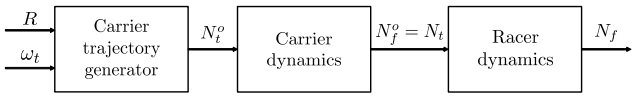


Figure 4.2: Collaborative case control scheme.

in-plane acceleration and velocity, obtained with high roll and pitch angles. In this case a low pass bandwidth filter with time constant of 0.1 s on the set-point position is implemented so as to filter possible noise on target position measurements.

4.2.2 Augmented position controller

In the non-collaborative case the target does not provide the follower with information about its state, hence the follower has to estimate or measure the target motion, in particular all the information needed by the augmented position control architecture and by the landing algorithm, *i.e.*, relative position and velocity. This problem can be solved through a visual based approach, *i.e.*, [10], [12] and [19]. Methods to obtain the measurements used as inputs in the algorithm are beyond the thesis objective. In this work Motion Capture (Mo-Cap) measurements are employed to get all quantities used in the algorithm. In particular, a centralized architecture is used, where a Ground Control Station (GCS) is able to communicate with all the UAVs. The Mo-Cap software running on the GCS measures the absolute position and velocity of all the UAVs; then, the relative position and velocity are computed and fed to the in-plane synchronization algorithm, which in turn sends the position set-point to the follower.
The augmented position controller employed in the non-collaborative case consists in a feedback architecture ([10]) that uses only information about the relative in-plane position $H_{tf}(t)$, defined in equation (4.1) (in-plane position error), and the relative in-plane velocity in the NED frame, namely the in-plane velocity error, computed as:

$$\dot{H}_{tf}(t) = v_t(t) - v_f(t) = \begin{bmatrix} \dot{N}_t(t) \\ \dot{E}_t(t) \end{bmatrix} - \begin{bmatrix} \dot{N}_f(t) \\ \dot{E}_f(t) \end{bmatrix} = \begin{bmatrix} \dot{N}_r(t) \\ \dot{E}_r(t) \end{bmatrix} = \begin{bmatrix} \dot{e}_N(t) \\ \dot{e}_E(t) \end{bmatrix}. \tag{4.6}$$

The in-plane acceleration set-point a_f, directed along the line of sight between the two UAVs, is then computed with the following Proportional-Integral-Derivative control law:

$$a_f(t) = \begin{bmatrix} a_N(t) \\ a_E(t) \end{bmatrix} = k_p H_{tf}(t) + k_i \int_{t_0}^{t} H_{tf}(\tau) d\tau + k_d \dot{H}_{tf}(t). \tag{4.7}$$

Considering that the trajectory tracking control module requires a position set-point, the acceleration set-point must be integrated twice. Before doing this, the

limitations in follower performance must be defined. In this case saturations on the in-plane acceleration set-point and in-plane velocity of the follower can be taken into account. In particular when the in-plane acceleration set-point computed through equation (4.7) is greater than a maximum value (a_{max}), instead of using the acceleration value computed by equation, the acceleration set-point vector is scaled by a factor $1/a_{max}$ so as to reduce its module while maintaining the same direction.
When the in-plane velocity reaches the maximum value v_{max}, an input acceleration that decelerates the follower is used.
These switching conditions can be summarized as follows:

$$u_f(t) = \begin{bmatrix} u_N(t) \\ u_E(t) \end{bmatrix} = \begin{cases} a_f(t) = \begin{bmatrix} a_N(t) \\ a_E(t) \end{bmatrix} & \text{for } \mid a_f(t) \mid \leq a_{max} \text{ and } \mid v_f(t) \mid < v_{max} \\ a_{max} \begin{bmatrix} \cos\gamma(t) \\ \sin\gamma(t) \end{bmatrix} & \text{for } \mid a_f(t) \mid > a_{max} \text{ and } \mid v_f(t) \mid < v_{max} \\ -K_{brake} \begin{bmatrix} \dot{N}_f(t) \\ \dot{E}_f(t) \end{bmatrix} & \text{for } \mid v_f(t) \mid \geq v_{max} \ , \end{cases} \tag{4.8}$$

where

$$\gamma(t) = \arctan \frac{a_E(t)}{a_N(t)} \tag{4.9}$$

and K_{brake} $[s^{-1}]$ is the brake constant that defines the forces to be applied when the acceleration overcomes the limit.
In the equations a_f is defined as the acceleration set-point computed through the PID law while u_f is the acceleration set-point really used after saturation and control of the switching conditions in equation (4.8).
As already said the input saturated acceleration $u_f(t)$ is then integrated twice and the position set-point for the trajectory tracking control module of the follower is obtained:

$$\begin{aligned} \dot{N}_f(t) &= \dot{N}_f(t_0) + \int_{t_0}^{t} u_N(\tau) d\tau, \\ \dot{E}_f(t) &= \dot{E}_f(t_0) + \int_{t_0}^{t} u_E(\tau) d\tau, \\ N_f^o(t) &= N_f(t_0) + \int_{t_0}^{t} \dot{N}_f(\tau) d\tau, \\ E_f^o(t) &= E_f(t_0) + \int_{t_0}^{t} \dot{E}_f(\tau) d\tau. \end{aligned} \tag{4.10}$$

In discrete time position and velocity are computed as:

$$\begin{aligned}
\dot{N}_f(k) &= \dot{N}_f(k-1) + u_N(k)\, T_{int}, \\
\dot{E}_f(k) &= \dot{E}_f(k-1) + u_E(k)\, T_{int}, \\
N_f^o(k) &= N_f(k-1) + \dot{N}_f(k)\, T_{int}, \\
E_f^o(k) &= E_f(k-1) + \dot{E}_f(k)\, T_{int}.
\end{aligned} \tag{4.11}$$

The integration initial conditions are the initial position and velocity of the follower when the chase begins, while T_{int} is the integration time step.
The position set-point just computed is sent to the follower at a specific rate, T_{sp}, namely set-point rate.
The augmented position control block diagram is shown in Figure 4.3 for the longitudinal direction, while the same applies to the lateral direction.

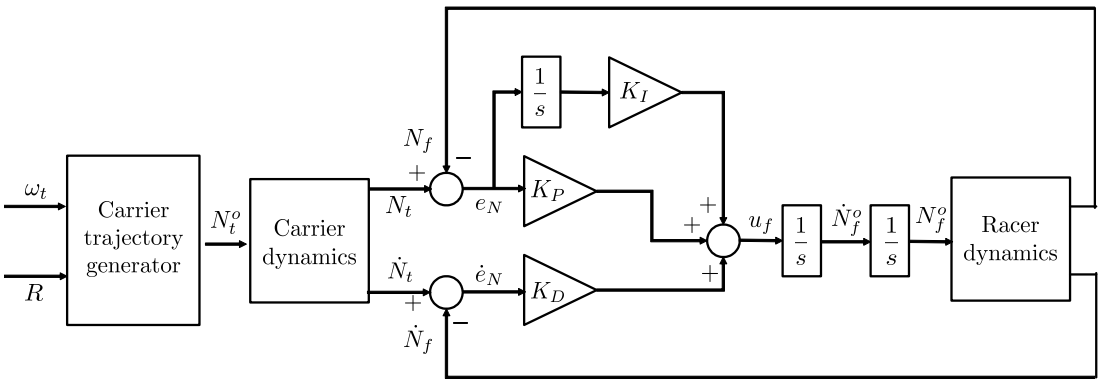


Figure 4.3: Augmented position control block diagram for in-plane synchronization in longitudinal direction.

4.3 Landing algorithm

Once the in-plane synchronization is obtained, the landing algorithm can start. The objective in equation (2.62) simplifies in:

$$\lim_{t \to t_f} (D_t(t) - D_f(t)) = \epsilon_D. \tag{4.12}$$

The landing trajectory is computed starting from an acceleration command $u_D(t)$ in Down direction, integrated twice because UAVs have their own position controllers onboard which can receive a position set-point, and aim to minimize position tracking errors.
To generate the acceleration command the three-states bang-zero-bang algorithm described in [7] is used.
The problem reduces to design a control input to minimize the total time to land the follower on the target, under the following constraints:

- the absolute ([7]) input acceleration of the follower should be bounded by a given limit

$$a_{D_{min}} \leq u_D(t) \leq a_{D_{max}}, \tag{4.13}$$

- the absolute descending velocity ([7]) of the follower should also be bounded

$$\mid v_f(t) \mid \leq v_{D_{max}}. \tag{4.14}$$

The reference trajectory D^o_{ref}, like the whole procedure, is discretized and updated at sampling time T_{int}. Assuming the total step is N_{stop}, the cost function is defined as in equation (4.15):

$$\min J = \min N_{stop}\ T_{int}. \tag{4.15}$$

Optimization constraints in discrete time are:

- initial and final state constraints: the initial states should be the same as the initial follower states and the final states should be the same as the final target states; in the following equations l is the sample index at which the descent begins, calculated from the start of the entire algorithm (including the initial synchronization):

$$\begin{aligned} D^o_{ref}(l) &= D_f(l), \\ \dot{D}^o_{ref}(l) &= \dot{D}_f(l), \\ D^o_{ref}(l + N_{stop}) &= D_t(l + N_{stop}), \\ \dot{D}^o_{ref}(l + N_{stop}) &= \dot{D}_t(l + N_{stop}). \end{aligned} \tag{4.16}$$

- System kinematic constraints: the kinematic model of the double-integrator is:

$$\begin{aligned} &\dot{D}^o_{ref}(k+1) = \dot{D}^o_{ref}(k) + u_D(k)T_{int} \\ &D^o_{ref}(k+1) = D^o_{ref}(k) + \dot{D}^o_{ref}(k+1)T_{int} \quad \forall\ k = l, ..., N_{stop}. \end{aligned} \tag{4.17}$$

- System input constraint: the acceleration input is bounded between the maximum downward and the maximum upward accelerations, respectively positive and negative according to the NED sign convention. These values are selected in order not to exceed the maximum reachable acceleration:

$$a_{D_{min}} \leq u_D(k) \leq a_{D_{max}} \quad \forall\ k = l, ..., N_{stop}. \tag{4.18}$$

- Landing velocity constraints: we bound the follower Down velocity during landing:

$$\mid \dot{D}_f(k) \mid \leq v_{D_{max}} \quad \forall\ k = l, ..., N_{stop}. \tag{4.19}$$

According to Pontryagin's theorem [16], the optimal solution is a bang-bang type control law. The novelty is that, to avoid abrupt changes in the angular speed set-point of the rotors, bang-zero-bang control law, involving three states instead of the classical two, is used.
Starting from the initial conditions, the time τ needed to reach zero relative velocity, applying the maximum deceleration $a_{D_{min}}$ (upward acceleration), is computed solving the linear equation

$$\tau(k) = -\frac{\dot{D}_r(k)}{a_{D_{min}}}. \tag{4.20}$$

At this step there are two possibilities: in case $\tau(k)$ is negative, this means that the two UAVs are moving away; being the solution unphysical, $\tau(k)$ is set to zero. In case $\tau(k)$ is positive we proceed evaluating the relative distance at time $t + \tau$, where the shorthand notation $\hat{D}_r^\tau(k)$ will be used instead of $D_r(t(k) + \tau(k))$ that is defined as:

$$\hat{D}_r^\tau(k) = \hat{D}_t^\tau(k) - \hat{D}_f^\tau(k), \tag{4.21}$$

$$\hat{D}_t^\tau(k) = D_t(k), \tag{4.22}$$

$$\hat{D}_f^\tau(k) = D_f(k) + \dot{D}_f(k)\tau + a_{D_{min}}\tau^2(k). \tag{4.23}$$

Based on the value of the relative distance after time $\tau(k)$, the control action is chosen:

- if the future relative position is lower or equal to the threshold on relative distance ϵ_D, it means that the follower "hits" the target, so the maximum upward acceleration (hence deceleration) $a_{D_{min}}$ must be applied to reduce the touch down velocity;
- if the future relative position is greater than ϵ_D there are two possible situations:
  - if the follower descent velocity limit $v_{D_{max}}$ has already been reached then zero acceleration is applied, in order to maintain the maximum descent velocity that minimizes the total landing time,
  - else the maximum downward acceleration must be applied to increase as much as possible the descent velocity and reduce the time to land.

It must be noticed that the case in which $\tau(k)$ is negative, to put it to zero corresponds to apply the maximum downward acceleration because the future vertical relative distance $\hat{D}_r^\tau(k)$ coincides with the actual one that must be positive otherwise the follower would be already landed. The procedure is summarized in the following equation (4.24):

$$u_D(k) = \begin{cases} a_{D_{min}} & \text{for} \quad \hat{D}_r^\tau(k) \leq \epsilon_D \\ 0 & \text{for} \quad \hat{D}_r^\tau(k) > \epsilon_D \text{ and } \mid \dot{D}_f(k) \mid \geq v_{D_{max}} \\ a_{D_{max}} & \text{for} \quad \hat{D}_r^\tau(k) > \epsilon_D \text{ and } \mid \dot{D}_f(k) \mid < v_{D_{max}}. \end{cases} \tag{4.24}$$

Parameter	Measurement unit	Value
Integration time step T_{int}	$[s]$	0.01
a_{max}	$[m/s^2]$	1
v_{max}	$[m/s]$	2
$v_{D_{max}}$	$[m/s]$	0.2
$a_{D_{min}}$	$[m/s^2]$	-0.1
$a_{D_{max}}$	$[m/s^2]$	0.1
$v_{D_{max}}$	$[m/s]$	0.2
ϵ_D	$[m]$	0.02
Target height h_t	$[m]$	-1.3
Follower initial height h_f	$[m]$	-2.8

Table 4.1: Parameters used in the simulations.

4.4 Simulation results

To test the synchronization and landing algorithms, before real flight experiments, simulations in MATLAB and Simulink are conducted. These simulations have also the objective of tuning the gains of the PID for the acceleration controller of non-collaborative synchronization, which, given relative position and velocity, computes an acceleration directed along the line of sight (LOS).
The UAVs have been modelled as point masses.
The simulations, that include the augmented position controller (Figure 4.3), are performed: the models of the closed loop dynamics in-plane position of target and follower constructed starting from the identified models of Section 3.1, are used. The simulation requires the target set-point as an input.
The follower and the target vertical dynamics are simplified with lumped mass model. This has been done because we are more interested in the synchronization phase than on the landing algorithm that instead has been already validated ([6]).
Based on models of the closed-loop position dynamics of the follower obtained in Chapter 3, it has been possible to carry out model-based tuning of the augmented position control architecture gains by means of the loop shaping approach. In Table 4.1 all the parameters used in simulation are reported.
In Table 4.2 the most interesting PID gains obtained for the longitudinal acceleration controller are reported. The two gain-sets selected for real flight test are A and B because, while the bandwidth is similar, higher phase margin of the follower closed-loop position dynamics with respect to the other two cases are obtained (Table 4.2). The bandwidth of the follower position controller (*i.e.*, the simple position controller architecture) imposes a limitation on the upper bandwidth achievable in the augmented position controller. In particular, the augmentation loop introduces a double integrator and the gains of the PID can be interpreted as the elastic and damping coefficients of a mass-spring-damper system. The bandwidth of such a system could be increased without limit while maintaining

stability, in absence of the position dynamics in the augmentation loop. This results in a trade-off between achievable bandwidth and robustness margins. The gains in Table 4.2 have been tuned so as to get the largest achievable bandwidth (which is fundamental to get good tracking performance) without impairing stability of the augmented closed-loop system (*i.e.*, avoiding too many oscillations), and result in a low phase margin. Moreover, it has been decided to keep the integral gain to zero, since its contribution has been considered not fundamental to trajectory tracking, and it would instead further slow down the response. The target and follower dynamic models of the in-plane positions are used. The

Gain-set	k_p	k_i	k_d	Phase margin $[deg]$	Cut-off frequency $[rad/s]$
A	0.3	0	1	30.4	1.09
B	0.4	0	1	24.8	1.12
C	0.5	0	1	19.4	1.15
D	0.5	0	0.9	19.4	1.07

Table 4.2: PID gains for the follower longitudinal acceleration controller.

simulation inputs are the target set-points for a circular trajectory defined as

$$\begin{aligned} N_t^o(k) &= R\,\cos(\omega_t\,k\,t_s) \\ E_t^o(k) &= R\,\sin(\omega_t\,k\,t_s + \pi), \end{aligned} \tag{4.25}$$

where $R = 1.5\ m$ is the radius of the circular trajectory, $\omega_t = 0.1\ rad/s$ is the angular frequency of the signal.
In Figure 4.4 it can be seen, that with gain-set A, after the initial transient, the follower is able to follow the target with good precision. The good performance of

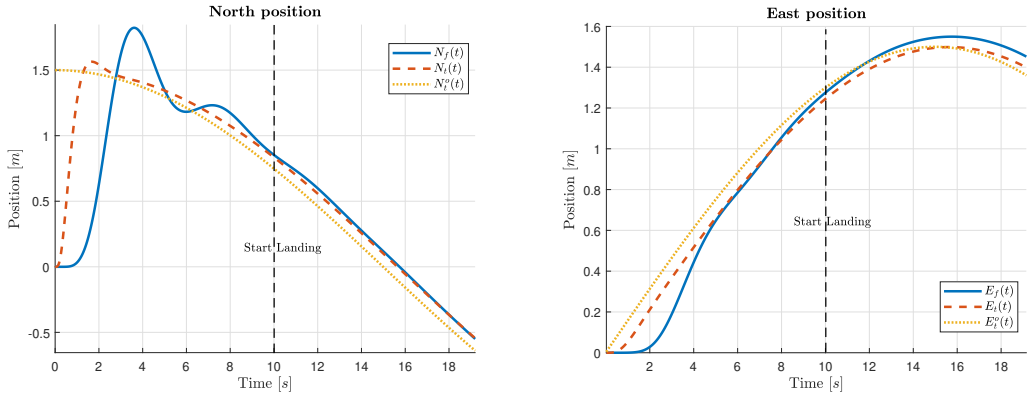


Figure 4.4: North and East positions simulated with gain-set A (Table 4.2).

the acceleration controller are confirmed also in Figure 4.5 where, except for the initial phase during which the target goes from the initial position at the centre of the circular trajectory to the starting position of the circular trajectory, the

horizontal relative position is lower then the threshold and during landing the trajectory lies always in the safety cone.

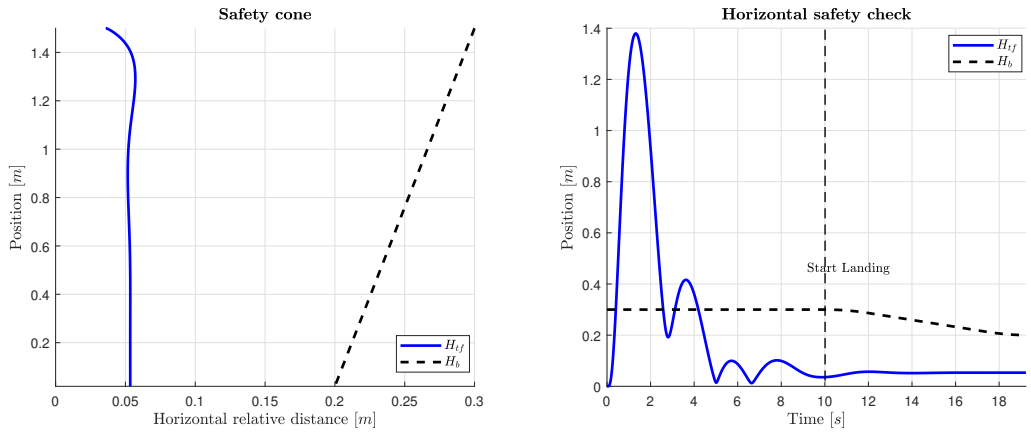


Figure 4.5: Safety cone during landing and horizontal safety check during the entire simulation with gain-set A (Table 4.2).

In Figure 4.6 the accelerations computed by the augmented loop PID controller are presented. The red line a_{sat} is the saturated acceleration effectively used to obtain the in-plane trajectory. Finally in Figure 4.7 the trajectory generated by the bang-zero-bang landing algorithm is shown.

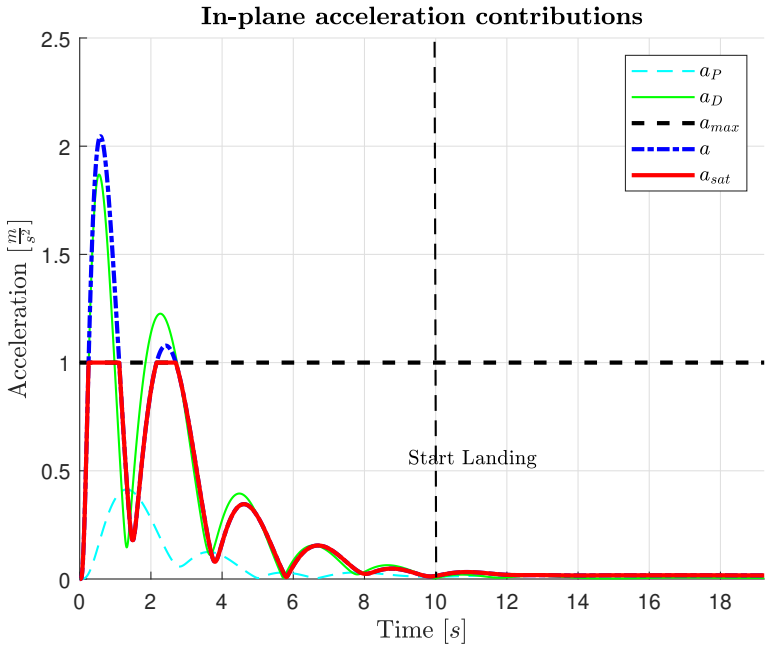


Figure 4.6: Accelerations for in-plane synchronization with gain-set A (Table 4.2).

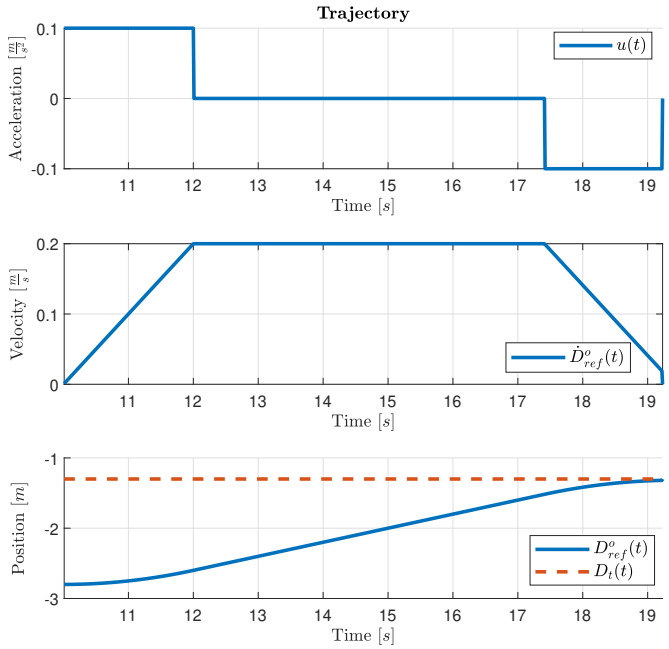


Figure 4.7: Landing trajectory computed during simulation with gain-set A (Table 4.2).

Chapter 5

Landing on oscillating target

In this chapter the procedure for landing with the target oscillating vertically is described. The chapter is divided in four parts: in the first one the problem is presented in general terms, in the second the motion estimation module necessary to perform the landing is described, in the third the landing algorithm is presented, highlighting the main differences with respect to the one in the previous chapter, and in the last part simulation results are finally shown.

5.1 Problem description

The problem of landing the follower on the target while the latter is oscillating vertically is quite different from the one in which it flies at constant altitude (*i.e.*, in-plane) and presents some critical issues.
The target is oscillating vertically as described by equation (5.1):

$$D_t(t) = h_t + \sum_{i=1}^{p} A_i \sin\left(2\pi f_i t + \beta_i\right), \tag{5.1}$$

where f_i is the i-th frequency, A_i is the i-th amplitude of oscillation, β_i is the i-th phase, p is the total number of sinusoidal components and h_t is the mean height around which the target is oscillating. In the tests conducted only one frequency component is considered in order to simplify the problem (*i.e.*, $p = 1$).
The problem, for the sake of simplicity, can be decoupled as in Chapter 4 in trajectory synchronization in the horizontal plane and vertical approach.

The objective of the synchronization in this problem is that the follower, that starts in a different position with respect to the target, must be able to reach and continuously be in the neighbourhood of the target North and East positions. Recalling the definition in equation (4.1), we want to verify that, after a transition time, the following relation is verified for all the time of vertical approach and

landing:

$$H_{tf}(t) = \sqrt{e_N^2(t) + e_E^2(t)} \leq \epsilon_N E(D_r); \tag{5.2}$$

where $\epsilon_N E$ has the same function of Section 4.2. The same safety cone of the Figure 4.1 is defined.
In this problem to keep the follower inside the safety area, *i.e.*, perform in-plane synchronization, the target in-plane position is given as set-point to the tracking control module of the follower, as described in Section 4.2.1. This has been done for the sake of simplicity, but the synchronization can be also performed with the augmented position controller described in Section 4.2.2.
Once the synchronization is obtained and maintained, the landing algorithm can start. The bang-zero-bang algorithm described in Section 4.3 is used with some modifications with respect to the in-plane problem because in this case the target is moving vertically. A motion estimation module is needed so as to apply the algorithm, as described later in Section 5.3.

5.2 Motion estimation module

In the sinusoidal landing case the trajectory generation module needs also a prediction of the target motion. This is obtained estimating online the amplitude and the phase of the oscillation and the height around which the target oscillates. To do this the Recursive Least Squares (RLS) algorithm is used ([20]) under the hypothesis to know the frequency of the oscillation a priori. Supposing to be in a partially collaborative case in which only the target position is known, the signal is represented exactly by the target position and is decomposed in a Fourier basis as in equation (5.3):

$$y(k) = [g(k)\cos(2\pi f T_s k) + h(k)\sin(2\pi f T_s k)] + z_0(k), \tag{5.3}$$

where T_s is the sampling period and for the frequency component f_i the amplitude is $A(k) = \sqrt{g(k)^2 + h(k)^2}$ and the phase is $\beta(k) = \arctan\left(g(k)/h(k)\right)$.
Some quantities must be defined at discrete time k: $y(k)$ is the signal that needs to be estimated, $\theta(k)$ is the vector of the real parameters containing the quantities that have to be estimated, the estimated parameter vector $\hat{\theta}(k)$, contains all the estimated parameters, *i.e.*, the coefficients of the Fourier decomposition and the mean height, and $\Phi(k)$ is the regression vector (equations (5.4)).

$$\begin{cases} \hat{\theta}(k) & = [\hat{g}(k)\ \hat{h}(k)\ \hat{z_0}(k)]^T \\ \Phi(k) & = [\cos(2\pi f T_s k)\ \ \sin(2\pi f T_s k)\ \ 1]^T. \end{cases} \tag{5.4}$$

Starting from these quantities the signal estimate can be computed through

$$\hat{y}(k) = \hat{\theta}^T(k)\Phi(k). \tag{5.5}$$

and the error is the difference between real and estimated signal $\varepsilon_k = y(k) - \hat{y}(k)$. The recursive least squares algorithm is implemented at the same rate of the set-point integration used in the landing algorithm, *i.e.*,$T_s = T_{int}$, in this way:

$$\begin{cases} \varepsilon_k & = y_k - \hat{\theta}_{k-1}^T \Phi_k \\ G_k & = \frac{1}{\lambda_0}\left(G_{k-1} - \dfrac{G_{k-1}\Phi_k^T \Phi_k G_{k-1}}{\lambda_0 + \Phi_k^T G_{k-1}\Phi_k}\right) \\ \hat{\theta}_k & = \hat{\theta}_{k-1} + G_k \Phi_k \varepsilon_k, \end{cases} \tag{5.6}$$

where the subscripts are used as a shorthand notation for $(\cdot)(k)$ and indicate the step, ε_k is the a priori error, G_k is the adaptation gain and λ_0 is the exponential forgetting factor, typically chosen between 0.98 and 0.995 (0.98 in our case).

With the estimation procedure, starting from an initial guess, the amplitudes $\hat{A}$, the phases $\hat{\beta}$ and the mean height $\hat{z_0}$ are obtained at each time step, so that future values of the position and velocity can be computed starting from actual parameter values using equation (5.7) and equation (5.8):

$$\hat{D}_t^{\Delta t}(k) = \hat{z_0}(k) + \hat{A}(k)\sin\left[\omega(kT_s + \Delta t) + \hat{\beta}(k)\right], \tag{5.7}$$

$$\dot{\hat{D}}_t^{\Delta t}(k) = \hat{A}(k)\omega\cos\left[\omega(kT_s + \Delta t) + \hat{\beta}(k)\right], \tag{5.8}$$

where $\omega = 2\pi f$, and the velocity is obtained by analytical differentiation of the estimated position. The notation $\hat{D}_t^{\Delta t}(k)$ means that the vertical position of the target is predicted on a time horizon Δt based on the parameter estimate at time k.
The accuracy of the RLS method is strongly dependent on the initial guess.

5.3 Landing algorithm

In this section, the landing algorithm for the case of oscillating target is described. Once the follower is in the safety cone, the vertical descent can start in order to reach the objective described in equation (4.12). Also in the sinusoidal case the landing trajectory is computed starting from an acceleration command $u_D(t)$ that, integrated twice, yields a position set-point used by the trajectory tracking module onboard the follower.
The acceleration command is generated through the three-states bang-zero-bang algorithm [7].
Starting from initial conditions, the maximum deceleration $a_{D_{min}}$ is applied (upward acceleration), the time instant $\tau(k)$ at which the estimated relative velocity becomes zero must be identified solving an equation that in this case is no longer linear in the unknown τ, as equation (4.20), and that requires the estimation

of the future motion of the target obtained with RLS described in the previous Section 5.2:

$$\hat{\dot{D}}_r^\tau(k) = \hat{\dot{D}}_t^\tau(k) - \hat{\dot{D}}_f^\tau(k) = 0, \tag{5.9}$$

with:

$$\hat{\dot{D}}_f^\tau(k) = \dot{D}_f(k) + a_{D_{min}}\tau(k), \tag{5.10}$$

$$\hat{\dot{D}}_t^\tau(k) = \hat{A}(k)\omega\cos\left[\omega(kT_s + \tau(k)) + \hat{\beta}(k)\right], \tag{5.11}$$

where $\tau(k)$ is the unknown variable.
To solve equation (5.9) the bisection algorithm [21] is applied because thanks to its simplicity it does not increase too much the computational time. The same two possibilities presented in Section 4.3 may occur: in the case $\tau(k)$ is negative, *i.e.*, the two UAVs are moving away, being the solution unphysical, $\tau(k)$ is set to zero, while in the case $\tau(k)$ is positive we proceed evaluating the relative distance at instant $t + \tau$, $\hat{D}_r(t(k) + \tau(k))$, using for the target the estimated parameters:

$$\hat{D}_r^\tau(k) = \hat{D}_t^\tau(k) - \hat{D}_f^\tau(k), \tag{5.12}$$

$$\hat{D}_t^\tau(k) = z_0(k) + \hat{A}(k)\sin\left[\omega(kT_s + \tau(k)) + \hat{\beta}(k)\right], \tag{5.13}$$

$$\hat{D}_f^\tau(k) = D_f(k) + \dot{D}_f(k)\tau(k) + \frac{1}{2}a_{D_{min}}\tau^2(k). \tag{5.14}$$

Based on the value of $\hat{D}_r^\tau(k)$, the control action is chosen in the same way described in Section 4.3. The procedure is summarized in this equation:

$$u_D(k) = \begin{cases} a_{D_{min}} & \text{for} \quad \hat{D}_r^\tau(k) \leq \epsilon_D \\ 0 & \text{for} \quad \hat{D}_r^\tau(k) > \epsilon_D \text{ and } \mid \dot{D}_f(k) \mid \geq v_{D_{max}} \\ a_{D_{max}} & \text{for} \quad \hat{D}_r^\tau(k) > \epsilon_D \text{ and } \mid \dot{D}_f(k) \mid < v_{D_{max}}. \end{cases} \tag{5.15}$$

5.4 Simulation results

To test the modifications done to the landing algorithm, simulations in MATLAB and Simulink environment are performed.
In simulation the two drones start in the same North and East position, synchronization is performed, only so as to maintain the follower inside the safety cone during landing, with the simple in-plane position controller described in Section 4.2.1. Gaussian noise with zero mean and variance $10^{-2}\,m^2$ is added to the target in-plane position measurement, that is constant at $[N_t, E_t]^T = [0, 0]^T$, to simulate the disturbance given by follower's rotor wake. This is used as set-point for the follower in-plane dynamic model identified in Section 3.1. The controller is able to maintain the follower inside the safety cone during the entire landing procedure.
So as to consider the time necessary to the initial in-plane synchronization, the

landing procedure starts after 12 s.
The vertical motion of the target is described by equation (5.1) with the parameters of Table 5.1.
The vertical dynamics of the follower has been modelled as a simple lumped mass.

Parameter	Measurement unit	Value
h_t	$[m]$	-1.3
p	$[-]$	1
A	$[m]$	0.1
$2\pi f$	$[rad/s]$	0.5
β	$[rad]$	0
$v_{D_{max}}$	$[m/s]$	0.3
ϵ_D	$[m]$	0.02
T_s	$[s]$	0.02
$a_{D_{min}}$	$[m/s^2]$	-0.1
$a_{D_{max}}$	$[m/s^2]$	0.1

Table 5.1: Parameters used for simulation.

In Figure 5.1 the input acceleration $u(t)$ is shown together with all quantities that determine it: when the follower vertical velocity $\dot{D}_f$ reaches the limits, the input acceleration becomes zero; the maximum upward acceleration is selected when the relative position estimate over a time horizon τ, $\hat{D}_r^\tau(t)$, is less or equal then the threshold ϵ_D (Figure 5.2) while the landing procedure ends when the relative position D_r is under the threshold ϵ_D.

In Figure 5.3 the follower and the target positions, velocities and accelerations (for the follower not the real acceleration but the input that then will be integrated) from the beginning of the landing to the end of the simulation are shown.

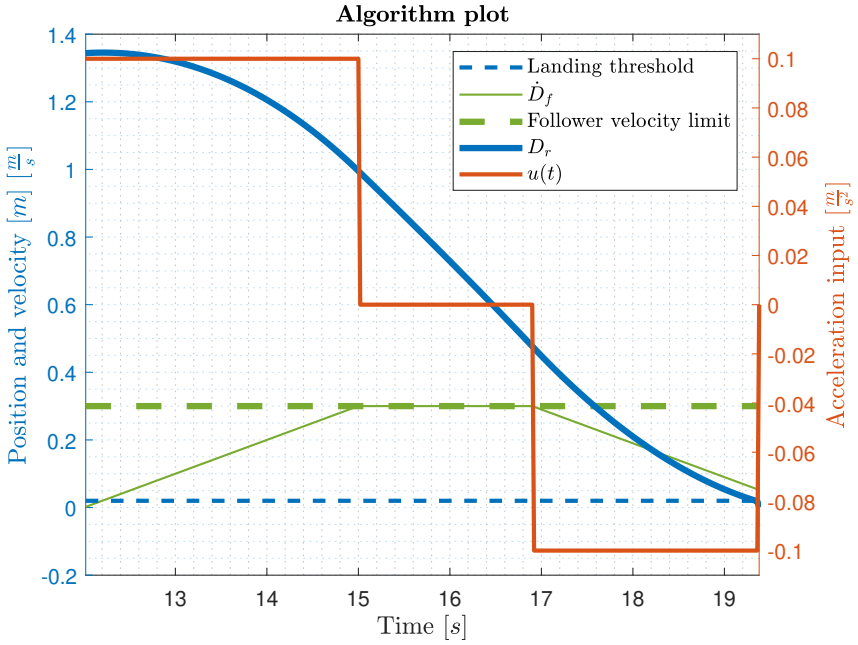


Figure 5.1: Quantities determined by the landing algorithm during simulation.

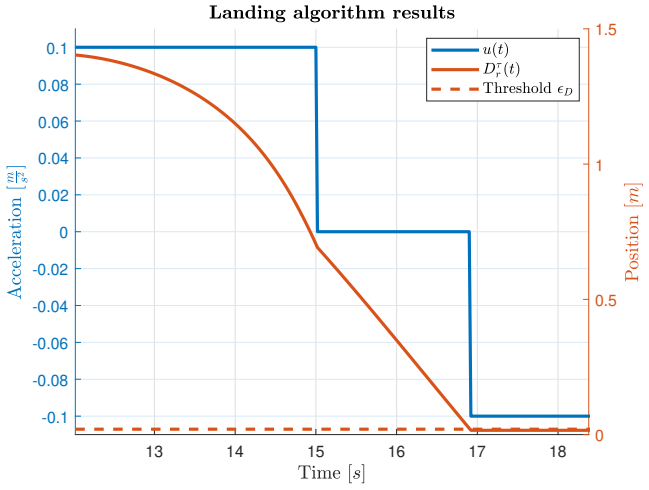


Figure 5.2: Landing algorithm results of the simulation: input acceleration with relative position computed after time τ.

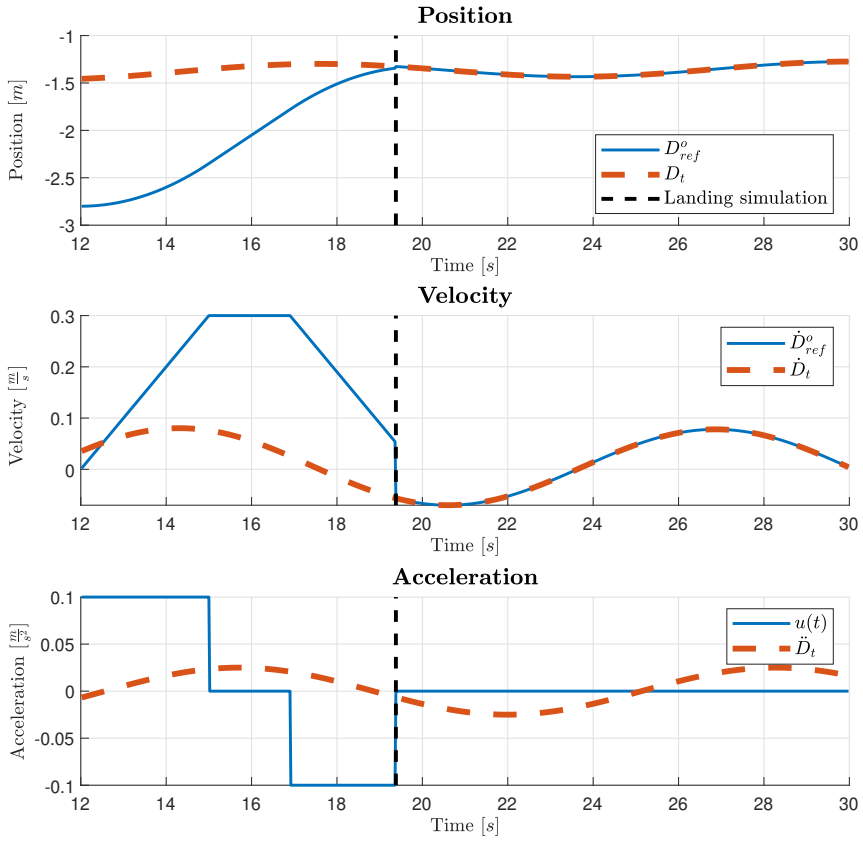


Figure 5.3: Position, velocity and acceleration of the drones during simulation.

Chapter 6

Experimental results

In this chapter an overview of the experimental setup (hardware and software) is presented. Then, the results of the preliminary study about landing performance degradation in presence of noisy and reduced frequency position measurements are shown. The in-plane and the sinusoidal landings algorithm are tested, in different conditions, in flight experiments whose results are finally presented.

6.1 System architecture

The aim of this section is to describe the hardware and software components of the system architecture used for flight sessions.
The landing algorithms have been tested inside the Flying Arena for Rotorcraft Technologies (FlyART) of Politecnico di Milano which is an indoor facility with a flight volume of 6 x 12 x 4 m, equipped with a motion capture system (Optitrack). In the following the Flight Control Unit (FCU) and companion computer used by both drones are described. Hence the characteristics of the two drones, of the motion capture system (Mo-Cap) and of the Ground Control Station (GCS) are presented.

6.1.1 Flight Control Unit

Each UAV has a FCU represented in Figure 6.1 that is the part dedicated to control simultaneously the revolutions per minute (RPM) of each motor to stabilize the drone in the air. In both drones the FCU used is the electronic board Pixhawk Mini [3] (designed by 3DR in collaboration with HobbyKing) which employs sensors, such as 3-axes accelerometer, 3-axes gyroscope, magnetometer and barometer.

The main features of the Pixhawk Mini are:

- Processor: main STM32F427 Rev 3 based on 32 bit ARM Cortex® M4

Figure 6.1: Pixhawk Mini FCU (from [3]).

core with 180 MHz CPU and an IO processor STM32F103 based on the Cortex® M3 core with 72 MHz CPU;

- Interfaces: UART serial port for GPS, spektrum DSM/DSM2/DSM-X® satellite compatible RC input, Futaba S BUS® compatible RC input, PPM sum signal RC input, I2C for digital sensors, CAN for digital motor control with compatible controllers, ADC for analog sensors and micro USB port;
- Weight: 15.8 g;
- Dimensions: 38 x 43 x 12 mm.

The MAVLInk [22] protocol is used for serial communication between FCU and companion, PX4 firmware [23] is the one supported by Pixhawk Mini and QGroundcontrol [24] is the software used for Pixhawk Mini configuration and real time information.

6.1.2 Companion computer

The companion computer (Figure 6.2) is the part of the drone system used to interface and communicate with PX4 on the FCU (Pixhawk Mini) using the MAVLink protocol. It enables a variety of functionalities, such as the possibility to execute processes that require heavy CPU load. During the flight test in the closed arena, the companion computer is used to receive:

- the position from the Ground Station, connected to the Mo-Cap system;
- the commands coming from the Ground Station;

these informations are sent to the FCU through the serial communication. On both drones the NanoPi NEO Air [4] in Figure 6.2, that has the characteristics reported in Table 6.1, is used.

From a software point of view the Robot Operating System (ROS) [25] and MAVproxy are installed into the companion:

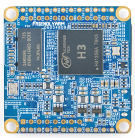


Figure 6.2: NanoPi NEO Air (from [4]).

Name CPU	Quad-core Cortex-A7 1.2 GHz
RAM	512 MB
Wireless	2.4 GHz 802.11 $b/g/n$
Dimensions	40 x 40 mm
Weight	7.9 g
Power	5 V - 2 A

Table 6.1: NanoPi NEO Air features.

- ROS: it is used to communicate with Ground Control Station through ROS messages. In particular Mavros [26], that is a ROS package, provides communication driver for Pixhawk Mini autopilot with MAVLink communication protocol. Additionally it provides UDP MAVLink bridge for Ground Control Station (*e.g.*, QGroundControl);

- MAVproxy: package intended for a minimalist portable and extendable Ground Control Station (GCS) for any UAV supporting the MAVLink protocol. Thanks to it, it is possible to control the Pixhawk Mini from GCS by means of the MAVLink protocol.

6.1.3 Drones

Follower drone

The racer UAV, codename ANT-R, represented in Figure 6.3 is used as follower drone. It is a quadcopter with 4 motors, Electronic Speed Controls (ESCs) and propellers, 1 Lithium-Ion Polymer battery (LiPo), the Pixhawk Mini FCU unit and the NanoPi NEO Air as companion.
Racer drone characteristics are reported in Table 6.2.

Target drone

The octocopter, codename CARRIER-1, represented in Figure 6.4 is used as target drone. It has 8 motors, Electronic Speed Controls (ESCs) and propellers, 2

Feature	ANT-R	CARRIER-1
Weight	0.73 kg	2.9 kg
Dimensions	19 x 17 x 8.5 cm	39 x 37 x 14 cm
Battery	LiPo 2650 mAh 4S	LiPo 8000 mAh 6S + 2200 mAh 3S
Propellers	3 blades 5045	2 blades 6535
Motors	Emax RS2205-2300KV	T-motor F40 PRO II 1600KV

Table 6.2: Drones characteristics.

Figure 6.3: Follower drone ANT-R.

Figure 6.4: Target drone CARRIER-1.

Lithium-Ion Polymer batteries (the bigger one for motors and the smaller one for electronics), the Pixhawk Mini FCU unit and the NanoPi NEO Air as companion. Target drone characteristics are reported in Table 6.2.

6.1.4 Motion capture system (Mo-Cap)

The Motion Capture system is composed by 12 Infra-Red (IR) sensitive Optitrack [27] cameras (Figure 6.5a) which incorporate IR flood lights. The cameras, mounted on the FlyART, are fixed at calibrated positions and orientation so that the measurement subject is into the field of view of multiple cameras. Through markers sensitive to infrared light (Figure 6.5b) mounted on top of the drones, it is possible to track and define their positions into space. Each UAV mounts a different marker layout to be uniquely identified when the two drones fly at the same time.
To control the motion capture system, the Motive software is installed on the ground station, it is designed to control motion capture systems for various tracking application. It not only allows the user to calibrate the system, but it also provides interfaces for capturing and processing 3D data, that can be recorded or live-streamed. The accuracy of the position estimated by the UAV depends on the frequency with which the position informations are sent to it (cameras rate), which can be selected in the range 30 - 240 Hz.

(a) Infrared camera (from [28]). (b) Infrared markers.

Figure 6.5: Motion Capture System.

6.1.5 Ground Control Station

The Ground Control Station architecture is divided into two different OS's: Windows 10 [29], in which Motive is installed, and Linux OS (more precisely Ubuntu 16.04 [30]) that is used to execute ROS and MATLAB.
The GCS has two main functionalities: to provide the attitude and the position measured by the Mo-Cap system (at a frequency of 100 Hz) and to send the position and heading trajectories to the UAVs using dedicated MATLAB functionalities. In particular, the guidance law which controls the position of both UAVs and the non-linear time optimal landing algorithm are implemented in MATLAB code running on the GCS as a centralized control architecture with integration and set-point rates of 50 Hz.

6.2 Study on degradation of the frequency

At the beginning of experimental work, a series of tests, varying the frequency of algorithm integration rate and the frequency at which the set-point is sent, have been conducted. This has been done to investigate the impact of a degradation in the frequency on the landing procedure and to obtain a set of requirements to drive the design of the relative navigation system, in particular considering that in a real outdoor scenario the position measurements are collected at lower frequency, *e.g.*, with GPS, than in case of indoor Motion Capture system.
To conduct experiments in safe conditions, the tests consisted in landing the follower on the target still on the ground. The in-plane synchronization has been obtained using the position controller described in Section 4.2.1 with the addition of an integral term on the position error to achieve zero steady-state positioning

error during landing as shown in the following equation:

$$\begin{aligned} N_f^o(k) &= N_t(k) + k_i N_i(k) \\ E_f^o(k) &= E_t(k) + k_i E_i(k), \end{aligned} \tag{6.1}$$

where k is a generic time instant, a value of 0.1 has been chosen for the integral gain k_i and $N_i(k)$ and $E_i(k)$ are the state of the integrator defined as follows:

$$\begin{aligned} N_i(k) &= N_i(k-1) + N_r(k) T_{int} \\ N_i(0) &= 0, \end{aligned} \tag{6.2}$$

$$\begin{aligned} E_i(k) &= E_i(k-1) + E_r(k) T_{int} \\ E_i(0) &= 0, \end{aligned} \tag{6.3}$$

The algorithm used to land is the three-states bang-zero-bang algorithm described in Section 4.3 with the simplification given by the fact that the target is on ground, *i.e.*, $\dot{D}_t(k) = 0 \ \ \forall k$.
In Table 6.3 all frequencies at which the tests have been conducted are reported. In all cases a set-point rate that is half of the integration rate is considered.

Integration rate $[Hz]$	Set-point rate $[Hz]$
100	50
80	40
75	37.5
60	30
50	25
40	20
25	12.5
10	5
5	2.5
2	1

Table 6.3: Tested frequencies.

The most important quantities on which the attention can be focused so as to carry out a comparison are the position error, the landing (*i.e.*, touch down) velocity and the landing time.
The absolute value of the in-plane position error can be computed at each time instant of the descent through equation (4.1) and equation (4.2). In Figure 6.6 the error of each test is plotted with the safety cone.

Ground contact detection is a key element in an autonomous landing procedure. To identify the touch down instant the acceleration measurements are used. In fact, when there is the touch down, clear peaks in each acceleration component

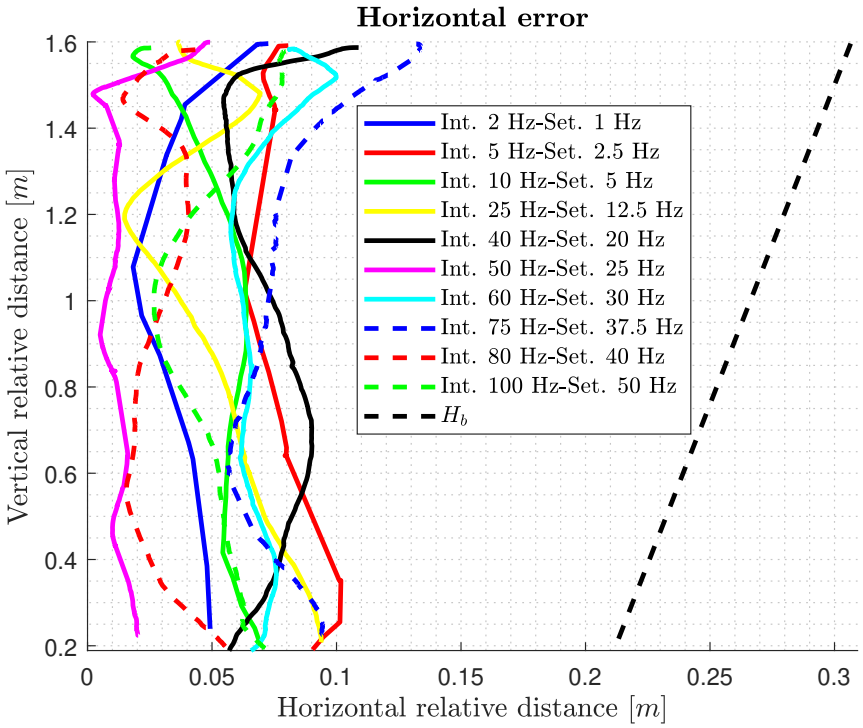


Figure 6.6: Horizontal error at different integration and set-point frequencies.

can be seen. At this point it seems reasonable that acceleration signals can be used in order to estimate the contact instant.
Two different methods are implemented: the first one identifies the touch down instant as the one at which the a_z acceleration experiences a variation over a certain threshold (chosen empirically based on the a_z time history), the second one defines a particular quantity, namely the Mean Acceleration Variance, and uses a procedure from [31]. The Mean Acceleration Variance is defined in equation (6.4), where a_x, a_y and a_z are the acceleration components along body-axes and N is the length of the time window in which the computation is performed ($N = 50$ during tests).

$$MAV(t) = \sum_{k=t-N}^{t} \frac{1}{N} \left\{ [a_x(k) - a_x(k-1)]^2 + [a_y(k) - a_y(k-1)]^2 + [a_z(k) - a_z(k-1)]^2 \right\} \tag{6.4}$$

The touch down instant can be identified as the one at which the MAV is above a threshold chosen empirically to be 750 m^2/s^4.
In Figure 6.7 an example of touch down instant identification is shown for the case of 75 Hz of integration rate. Details about ground contact detection using mean acceleration variance can be found in [31].

Once identified the touch down instant, the landing velocity and the landing time

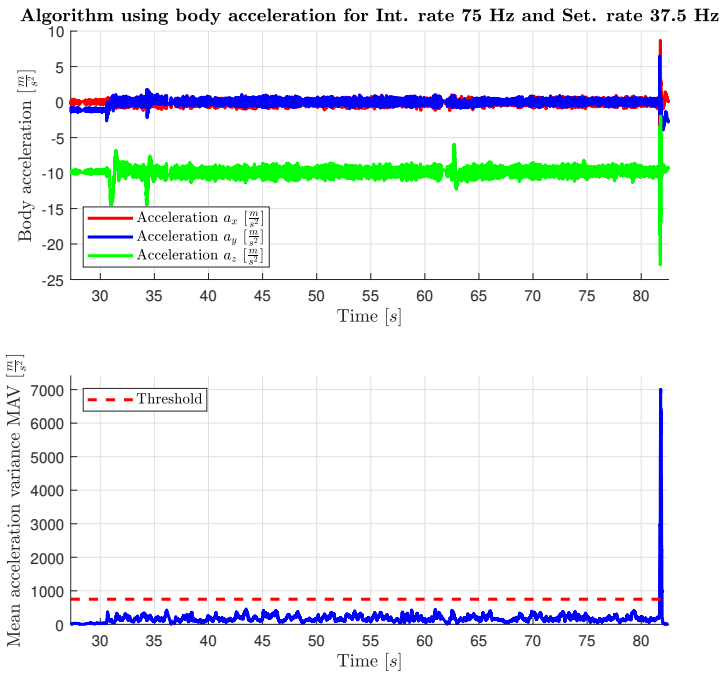


Figure 6.7: Example of ground contact identification with $1/T_{int} = 75\ Hz$.

can be obtained from sensors measurements (the velocity time history is estimated onboard by an Extended Kalman Filter). In Table 6.4 the landing velocity obtained with both methods used to identify the touch down are reported, while the final horizontal error and the time to land calculated from the start of the landing algorithm to the touch down are shown in Table 6.5. A comparison of the results can be done between the two touch down identification methods, and it can be seen that except for a few cases the results are quite similar to consider the two procedures equivalents.
In Figure 6.8 the landing velocity variation with frequency evaluated with both methods are shown, plotting also three samples before and three samples after the touch down to see the variance.

In Figure 6.9 the landing time variation with frequency can be seen, while in Figure 6.10 the horizontal distance of the racer from the centre of the carrier at touch downt.

As can be seen from the figures a clear trend of landing time, velocity end error variation with frequency cannot be identified. For this reason the only conclusion that can be reached is that the in-plane position error and the landing velocity are limited, under 12 cm and 0.25 m/s respectively, for frequency range from 2 to 100 Hz. A possible interpretation of this behaviour is that at low frequency, as

Integration rate $[Hz]$	Set-point rate $[Hz]$	Landing velocity $[\frac{m}{s}]$	
		a_z method	MAV method
100	50	0.1198	0.1198
80	40	0.1449	0.1359
75	37.5	0.1739	0.1739
60	30	0.1230	0.0873
50	25	0.1312	0.1076
40	20	0.1393	0.1393
25	12.5	0.1752	0.1752
10	5	0.1468	0.1468
5	2.5	0.1828	0.1828
2	1	0.1407	0.1407

Table 6.4: Landing velocity and horizontal error at touch down identified with MAV with a threshold of 750 $[m^2/s^4]$ compared with the first method.

Integration rate $[Hz]$	Set-point rate $[Hz]$	Landing time $[s]$	Horizontal error $[m]$
100	50	7.065	0.0717
80	40	7.185	0.0567
75	37.5	7.095	0.0936
60	30	7.145	0.0620
50	25	7.035	0.0203
40	20	7.265	0.0555
25	12.5	6.925	0.0937
10	5	7.015	0.0696
5	2.5	6.704	0.0904
2	1	6.755	0.0492

Table 6.5: Landing time and horizontal error with touch down identified with first method.

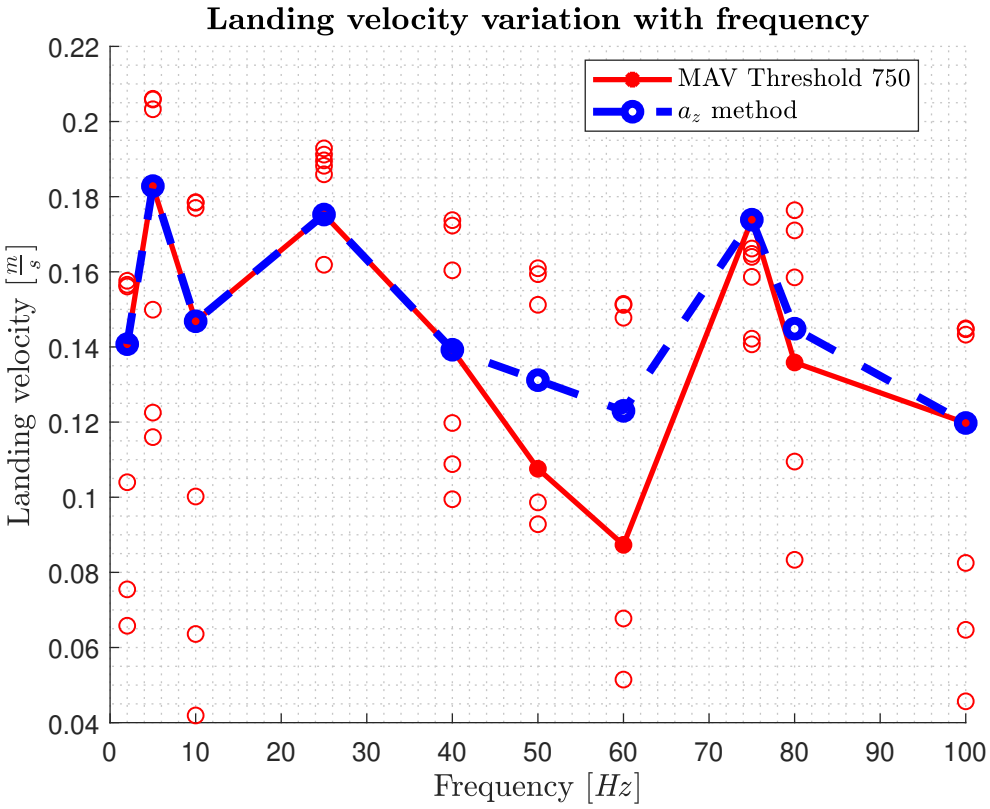


Figure 6.8: Landing velocity variation with frequency.

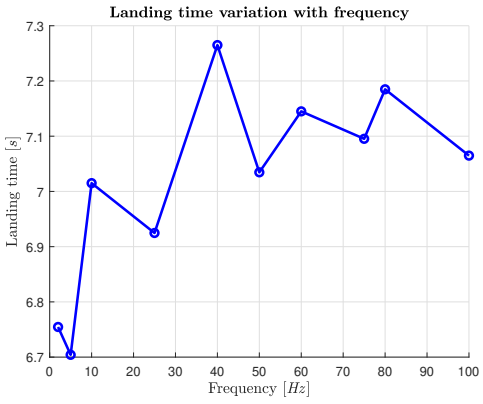


Figure 6.9: Landing time.

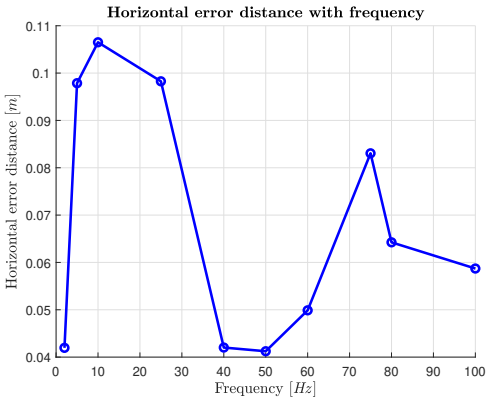


Figure 6.10: Horizontal error.

expected, the algorithm works in a worse way because the set-point are received too slowly while over 50 Hz frequency worsens because the Ground Computer is not able to sent the set-point at the desired rate.

6.3 Landing in presence of noise

After the tests about landing in different conditions of integration and set-point frequency, the landing with different signal-to-noise ratio of position measurement has been studied. In particular, the idea has been to test if it is possible to land in a "simulated outdoor environment": zero mean Gaussian discrete white noise is summed to the Mo-Cap measured position which is then converted to GPS coordinates, the standard deviation (STD) can be selected by the user, an independent realization of noise is obtained for each of xyz coordinates; finally the signal is subsampled at 5 Hz (GPS frequency).
Because of the presence of noise, a simple filter is implemented for position measurement: a moving average (see equation (6.5) as an example on the Down component) of a length of 50 samples is used. In equation (6.5) N is the number of samples used for the mean and the symbol $\tilde{(\cdot)}$ identifies the filtered measurement. After the initial transient phase, *i.e.*, collecting N samples, the filter equations are:

$$\tilde{D}_f(k) = \frac{1}{N} \sum_{i=k-N}^{k} D_f(i), \tag{6.5}$$

$$\tilde{D}_t(k) = \frac{1}{N} \sum_{i=k-N}^{k} D_t(i). \tag{6.6}$$

In Table 6.6 test conditions are reported. For safety reasons the tests have been performed with the target on ground, because at this time the only objective is to see how the repositioning and landing algorithm works varying the noise on the follower UAV position measurement.

Test	Noise STD	Moving average	Time window length of moving average
A	5 cm	No	1 samples
B	0 cm	Yes	50 samples
C	5 cm	Yes	50 samples
D	10cm	Yes	50 samples

Table 6.6: Noise tests.

The following quantities for Test D (Table 6.6), with 10 cm of noise standard deviation, are presented as examples:

- position and velocity of the follower used in the algorithm (5 Hz) in Figure 6.11,
- the acceleration, velocity and position trajectory computed by the bang-zero-bang algorithm, in particular the position is the one used as set-point for the follower, in Figure 6.12,
- the horizontal relative distance plotted against the time in Figure 6.13,
- the horizontal distance, *e.g.*, in-plane position error, variation with vertical distance; in Figure 6.14 we can see if the follower goes out of the safety area.

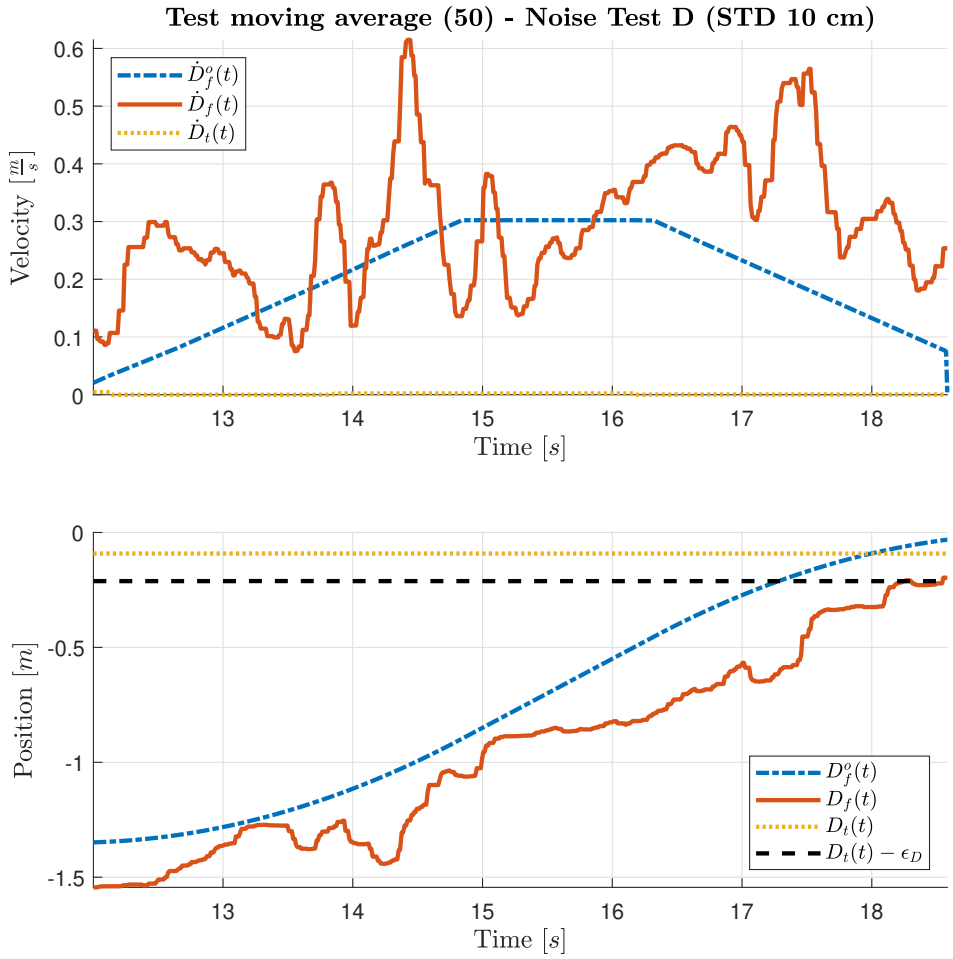


Figure 6.11: Follower position and velocity during Test D (STD 10 cm) landing.

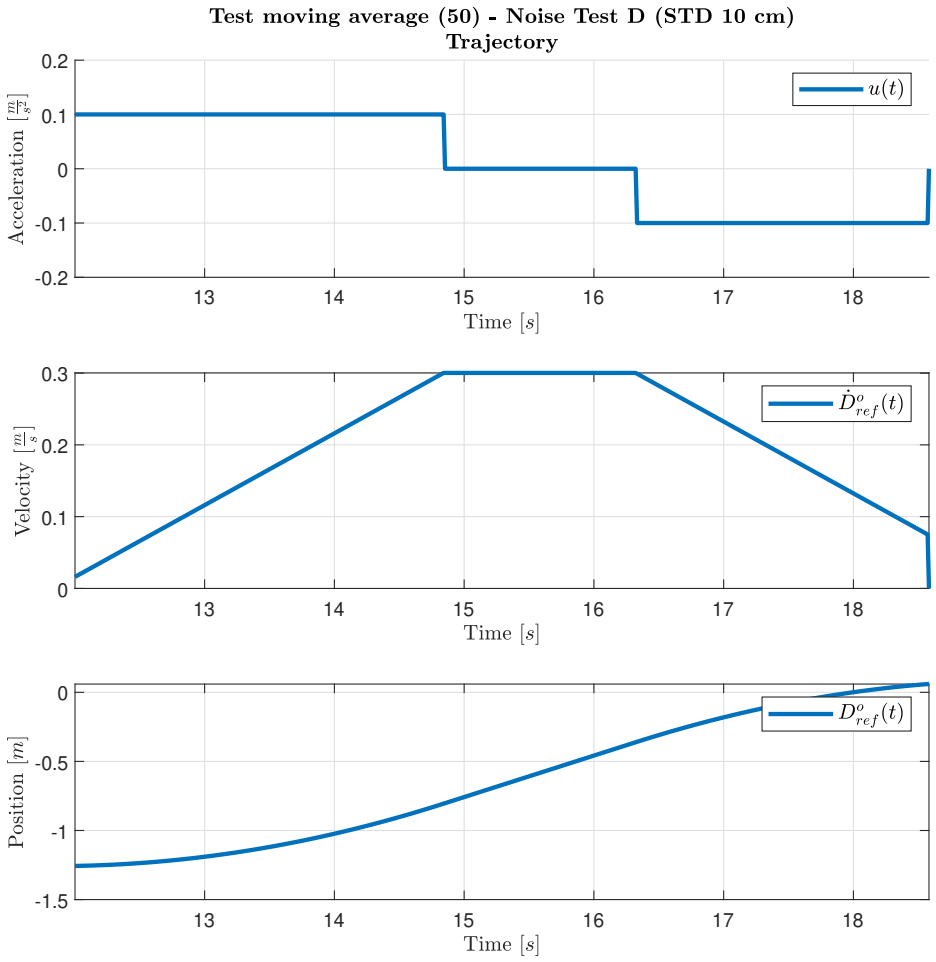


Figure 6.12: Landing trajectory computed for Test D (STD 10 *cm*).

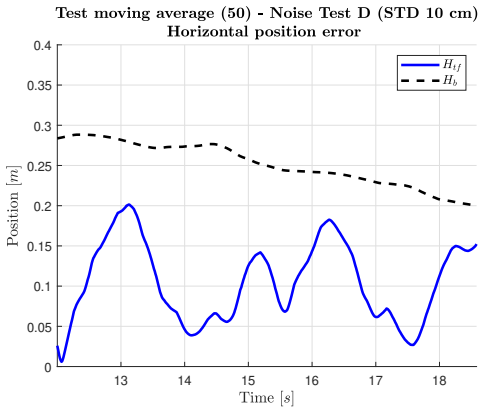


Figure 6.13: Horizontal error during Test D (STD 10 *cm*).

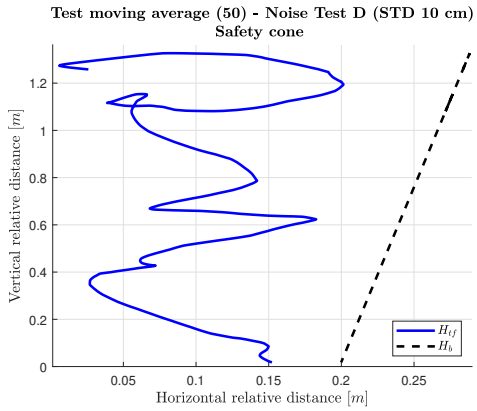


Figure 6.14: Safety cone during Test D (STD 10 *cm*).

In Figure 6.15 the in-plane error is plotted against vertical distance for tests with different noise levels, while Figure 6.16 shows the different values of the root mean square of the in-plane position error for the tests.

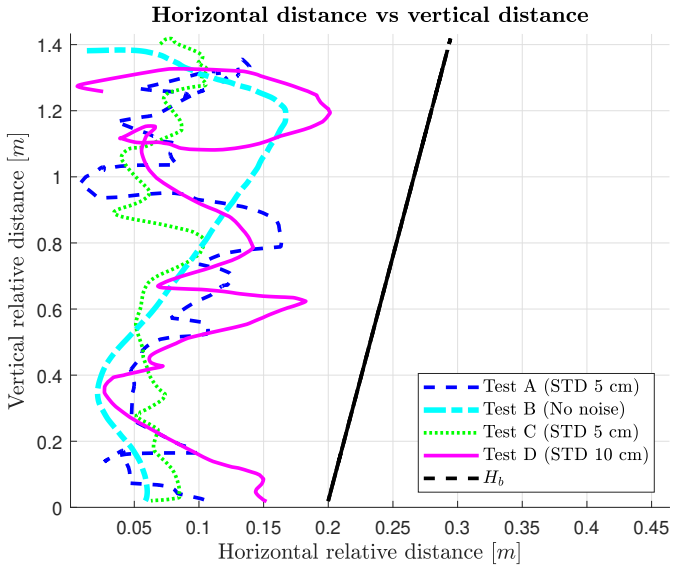


Figure 6.15: In-plane position error for tests with different noise levels.

Finally, in Figure 6.17 the North and East relative position of the follower with respect to the target for each instant of the landing procedure with different noise levels are shown. The beginning of the landing is identified by the asterisk marker while the final position by the circle. It must be noted that the case with noise of 10 cm is the one with greater values of relative position (error) as expected.

6.4 In-plane landing

The landing with target moving in the plane has been performed with the parameters reported in Table 6.7, this corresponds to a platform moving along a circular trajectory at 15 cm/s. In particular the simple position controller and two sets of gains for the in-plane acceleration augmented position controller of the follower are tested.

In Section 6.4.1 the results of the simple controller are shown, in Section 6.4.2 the performance of two sets of gains used for the augmented position controller are evaluated. Finally a comparison between the simple and augmented controllers under the same experiment conditions is done.

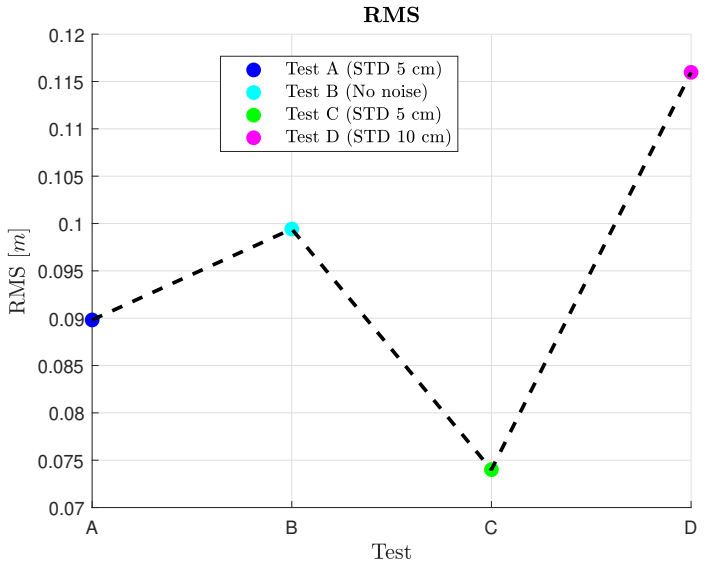


Figure 6.16: Root mean square of the in-plane position error for the tests with different noise levels.

Parameter	Measurement unit	Value
Integration time step T_{int}	$[s]$	0.02
a_{max}	$[m/s^2]$	1
v_{max}	$[m/s]$	2
$v_{D_{max}}$	$[m/s]$	0.3
$a_{D_{min}}$	$[m/s^2]$	-0.1
$a_{D_{max}}$	$[m/s^2]$	0.1
ϵ_D	$[m]$	0.02
Target initial position	$[m]$	$[1.5\ -2\ -1.3]^T$
Follower initial position	$[m]$	$[0\ -3.5\ -2.8]^T$
Centre of the circular trajectory	$[m]$	$[0\ -2\ -1.3]^T$
Radius of the circular trajectory R	$[m]$	1.5
Pulsation for the circular trajectory ω_t	$[rad/s]$	0.1

Table 6.7: Parameters for in-plane landing experiments.

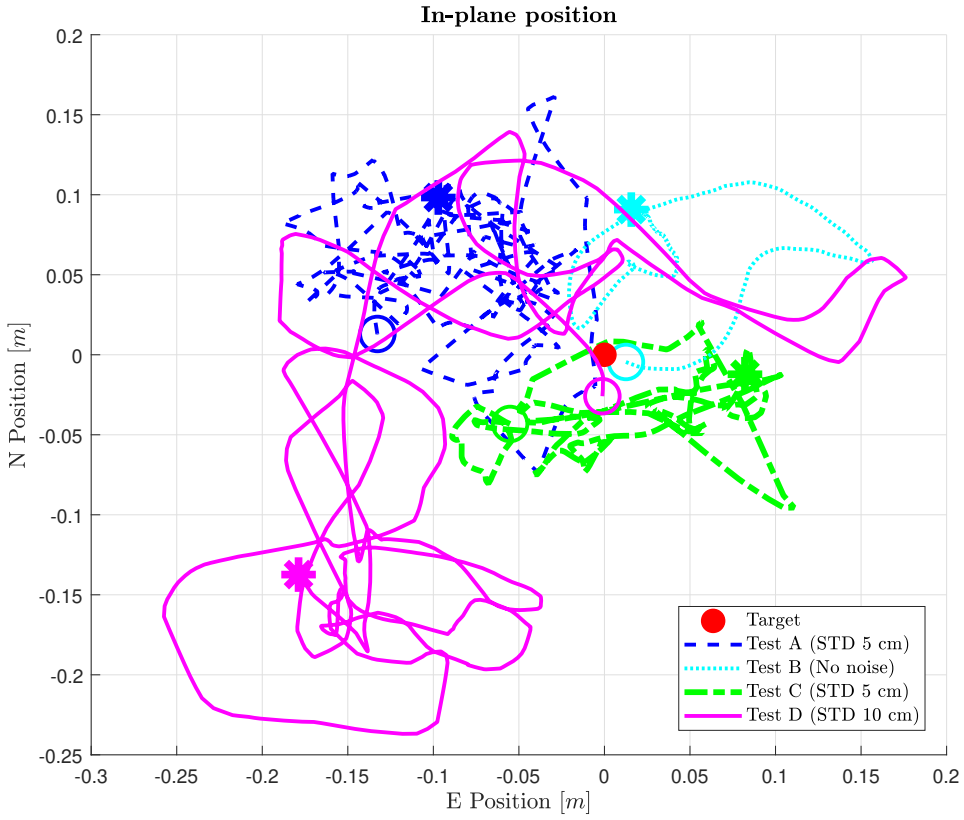


Figure 6.17: In-plane relative position of the follower with respect to target during landing with different noise levels.

6.4.1 Collaborative case: simple position controller

In Figure 6.18 and Figure 6.19 the in-plane positions and velocities of the target and of the follower are shown as example of synchronization using the simple position controller; in fact, in these two figures it can be noted that the follower position set-point almost coincides with the target position.
Figure 6.20 shows the Down position and velocity of the landing procedure, where the descent trajectory is computed with the bang-zero-bang algorithm (Figure 6.21).

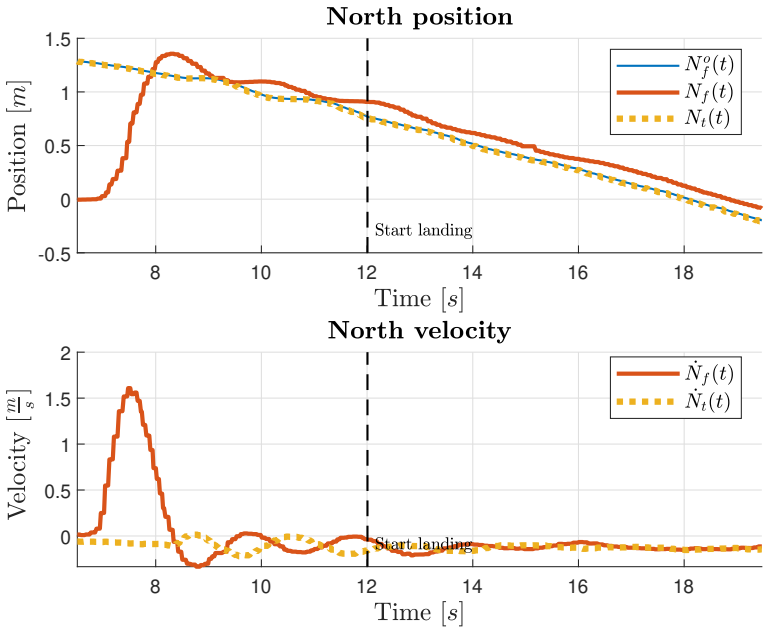


Figure 6.18: North position and velocity from the beginning of the synchronization with simple position controller.

6.4.2 Non-collaborative case: augmented position controller

The first two sets of gains in Table 4.2 are the ones used for the augmented position controller.
In Figure 6.22 and Figure 6.23 the in-plane positions and velocities of the target and of the follower are shown as an example.
In Figure 6.24 are shown the Down position and velocity of the landing procedure, with the descent trajectory computed with the bang-zero-bang algorithm (Figure 6.25). It can be noted that around 13 s the algorithm is paused because the synchronization constraint is not satisfied.
The same gain values are also successfully tested with target moving on the circular trajectory with constant velocity of 30 cm/s ($\omega_t = 0.2\ rad/s$) but with larger time

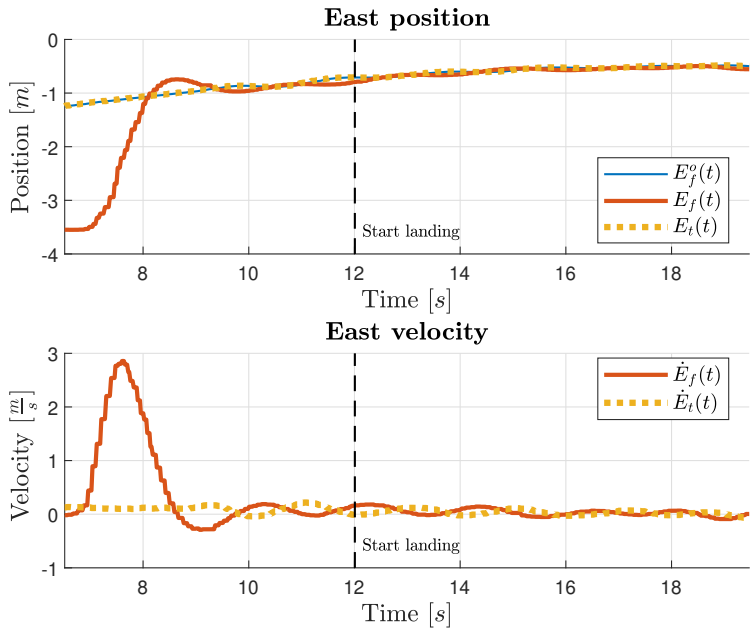


Figure 6.19: East position and velocity from the beginning of the synchronization with simple position controller.

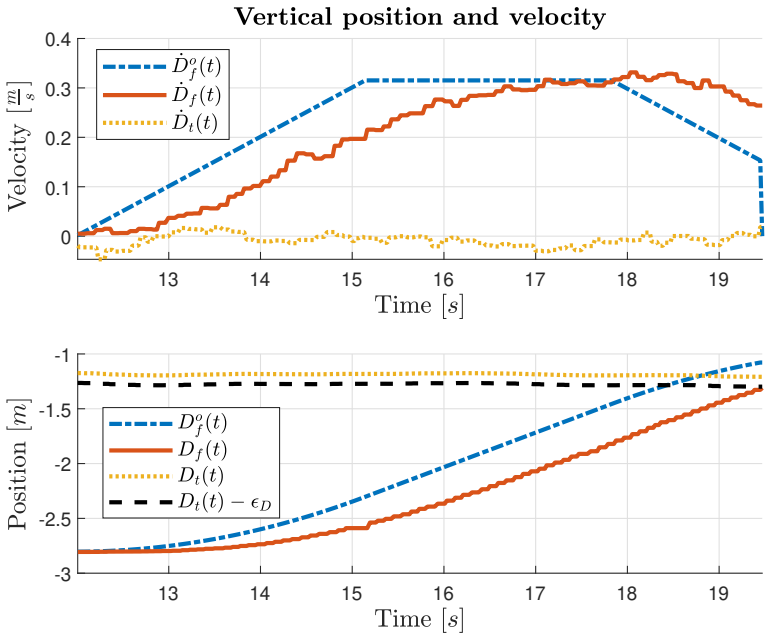


Figure 6.20: Down position and velocity during landing with simple position controller.

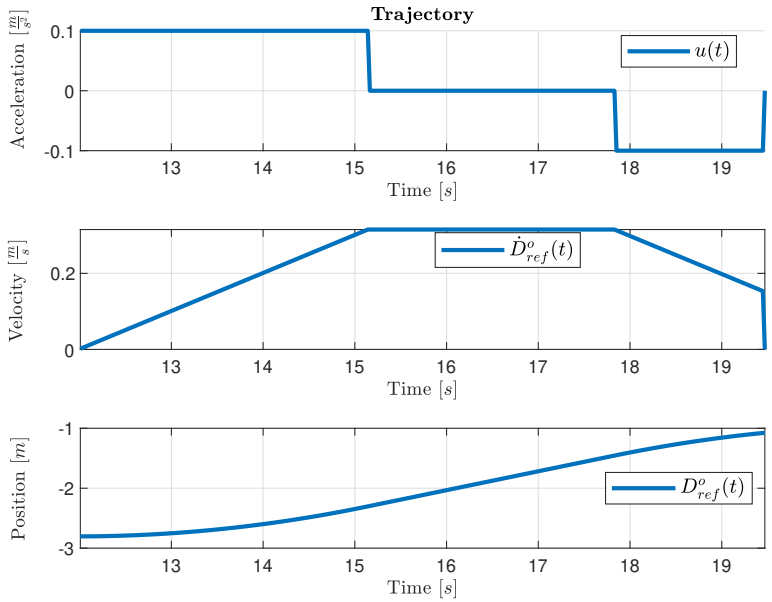


Figure 6.21: Landing trajectory in the test with simple position controller.

to land and larger horizontal final error values (in one case the follower landed on the safety cone limit). Results are summarized in Table 6.8.

6.4.3 Comparison of the position controllers

A comparison between the simple and augmented in-plane position controllers can be performed in terms of horizontal error from the beginning of the synchronization to the touch down (Figure 6.26): the simple position controller is the fastest one but has an error at steady state greater than the augmented position controller, the augmented position controller with the lowest value of the proportional gain (Table 4.2 Gain-set A) is the slowest one but it is also the most accurate in the low pulsation case. It must be noted that the position controller with the largest value of the target trajectory frequency ($\omega_t = 0.2\ rad/s$) has bad performance; in this case it would be better to use different values of the gain parameters. In Table 6.8 final horizontal distance, relative velocity at touch down and time to land for each test are reported.

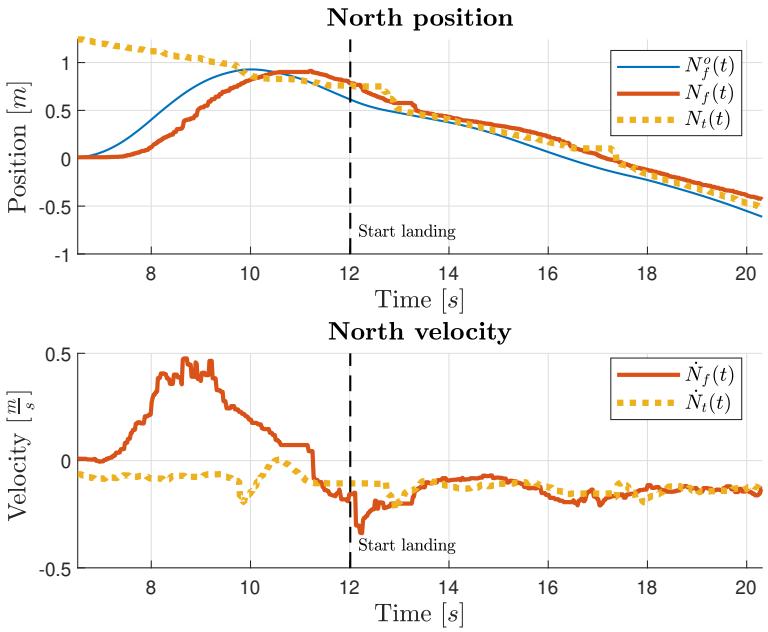


Figure 6.22: North position and velocity from the beginning of the synchronization with the augmented position controller gain-set A (Table 4.2).

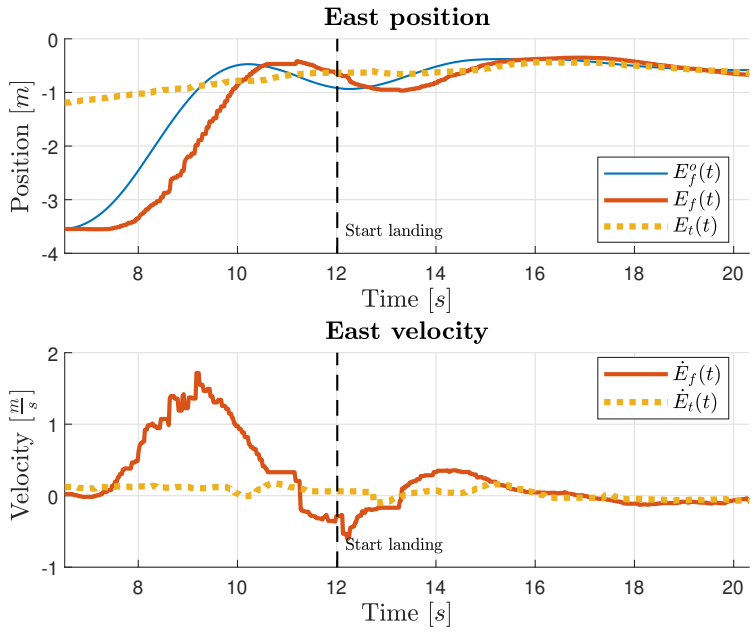


Figure 6.23: East position and velocity from the beginning of the synchronization with the augmented position controller gain-set A (Table 4.2).

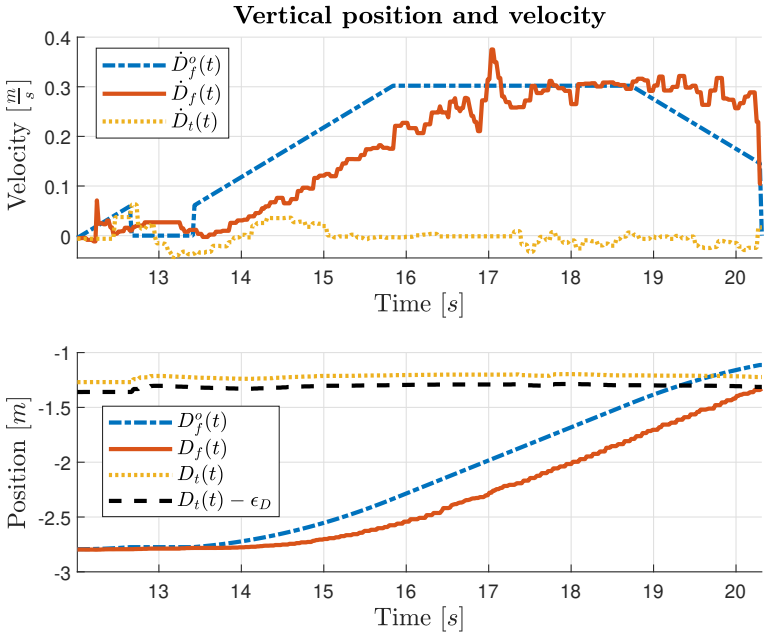


Figure 6.24: Down position and velocity during landing with the augmented position controller gain-set A (Table 4.2).

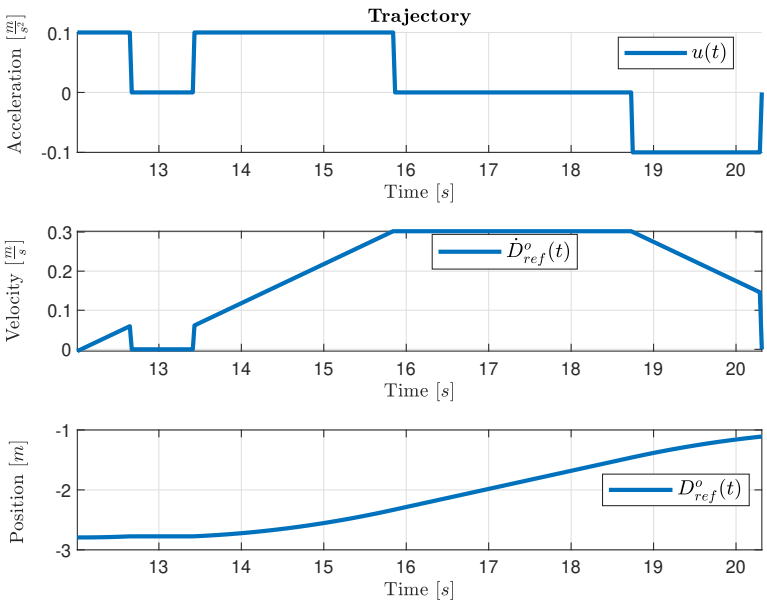


Figure 6.25: Landing trajectory in the test with the augmented position controller gain-set A (Table 4.2).

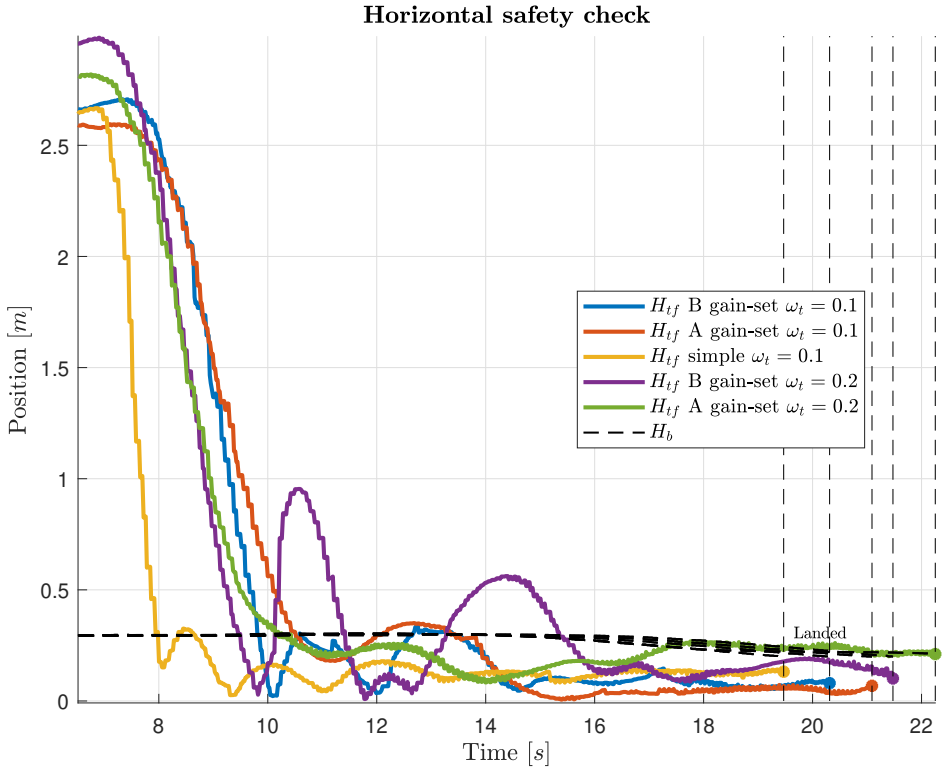


Figure 6.26: Horizontal error comparison for the two controllers in different conditions (Table 4.2).

Controller and gain-set	Final error $[m]$	$\dot{D}_r$ touch down $[m/s]$	Time to land $[s]$
Simple $\omega_t = 0.1$	0.1317	-0.2456	7.46
Augmented $\omega_t = 0.1$, A	0.0682	-0.2335	9.08
Augmented $\omega_t = 0.1$, B	0.0871	-0.2536	8.27
Augmented $\omega_t = 0.2$, A	0.1952	-0.2719	11.16
Augmented $\omega_t = 0.2$, B	0.1014	-0.3160	9.46

Table 6.8: Landing performance comparison with gain-set of Table 4.2.

6.4.4 Comparison between experiments and simulations results

After the real flight tests, a verification of the model used for the in-plane synchronization has been performed. In particular relative positions and velocities of the experiment, carried out in the same conditions of the simulation presented in the previous Section 4.4 with acceleration controller parameters A of Table 4.2, are used to compute the in-plane trajectory for synchronization given as input to the dynamic model of the follower. Figure 6.27 shows the good behaviour of the follower trying to follow North and East position and velocity of the target that is moving along a circular trajectory with radius 1.5 m, centred in $[N, E]^T = [0, 1.5]^T$.

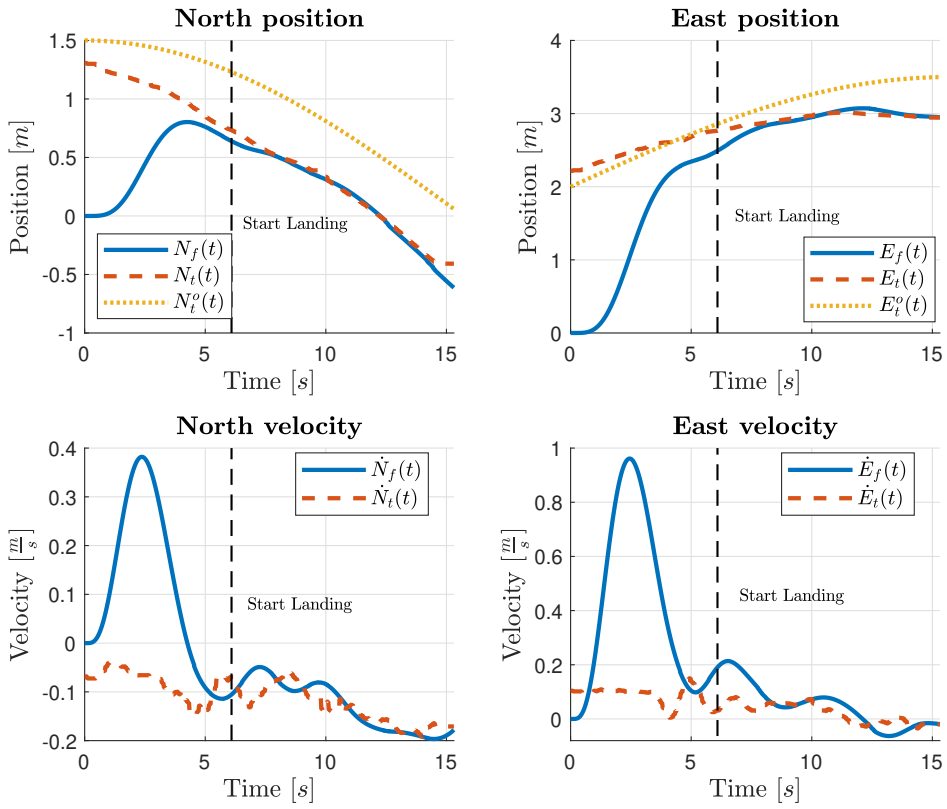


Figure 6.27: North and East positions and velocities simulated starting from real experiment data of target motion with A parameters of Table 4.2.

In Figure 6.28 positions simulated from the model and positions measured during the real experiment present similar trend. This means that the model used for the simulation is reliable. Also in terms of position error in Figure 6.29 can be seen that the safety check behaviour is similar.

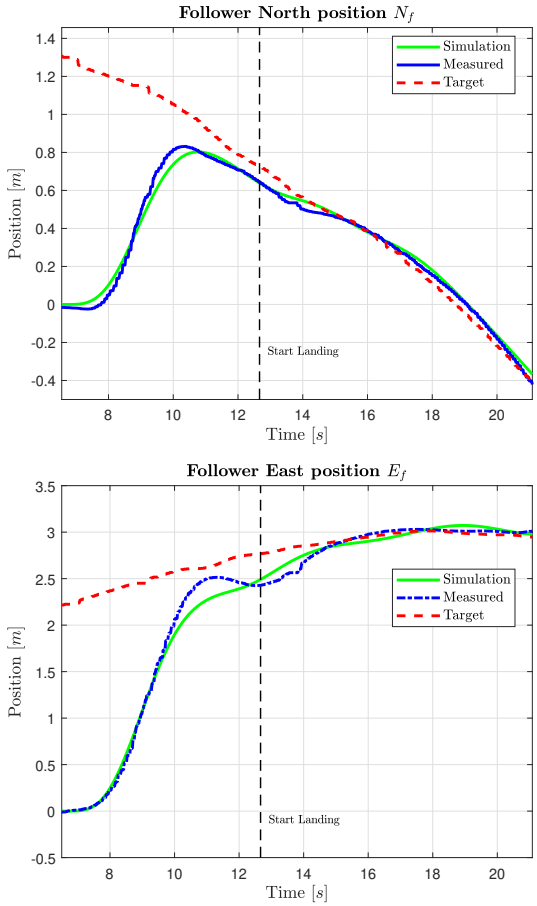


Figure 6.28: Comparison between position of the follower simulated and position of the follower during in-plane landing real experiment with A parameters of Table 4.2.

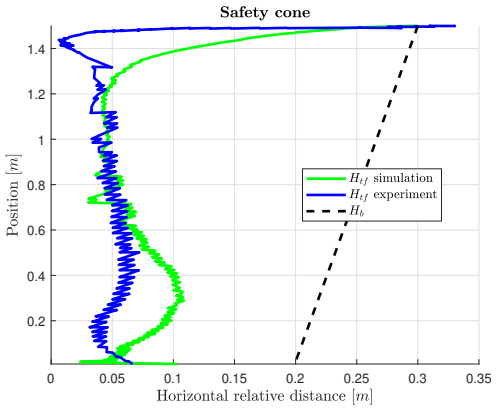


Figure 6.29: Comparison between simulated safety cone and real experiment data during in-plane landing.

Test	$\hat{g}(0)$	$\hat{h}(0)$	$\hat{z_0}(0)$
A	0	0.1	-1.3
B	0	0	-1.3

Table 6.9: Initial guess $\hat{\theta}(0)$ for the estimation.

6.5 Landing on oscillating target

In this section results of the landing on oscillating target experiments are shown. In the first part the experiments conducted to test the estimation module are presented, then the results for landing on oscillating target are shown. Finally a comparison in terms of τ between experiments and simulations is made.

6.5.1 Test for estimation module

First of all, the estimation module has been tested in two ways: in the first case post-processing flight data and in the second estimating in real time.
During both experiments the target oscillates with frequency $\omega = 0.5\ rad/s$ and amplitude of oscillation $A = 0.1\ m$ for 60 s in undisturbed conditions, and then for the same period in disturbed conditions, *i.e.*, with the follower hovering over the target.
Different values of the forgetting factor have been tested during the post-processing estimation, then the value of $\lambda_0 = 0.98$ has been selected and used in the real-time case because it gives the best results.
The estimation tests have been performed with different values of the initial guess, because the more the guess is far from the real value, the more is the time needed to RLS to converge. In Table 6.9 two initial guesses are reported, the first consists in the exact values used to generate the set-point for target motion, while the second consists in wrong values (except for the height that is supposed to be known).

In Figure 6.30 the position and velocity estimated by RLS with exact initial guess are compared with the measured data; it must be noted that actually they correspond to the parameter values used to generate the set-point target motion, but, when the follower is above the target, the motion of the latter is quite different from the set-point because of aerodynamics disturbance, so also this initial guess is indeed "wrong" but it constitutes a good initial guess for the estimation.

In Figure 6.31 the results for wrong initial guess are shown; in this case the estimate needs more time to converge.
Figure 6.32 and Figure 6.33 show real and estimated target position and velocity respectively; in the moment in which the follower arrives above the target, approximately at 70 s, the estimation gets worse but after a transient time it converges

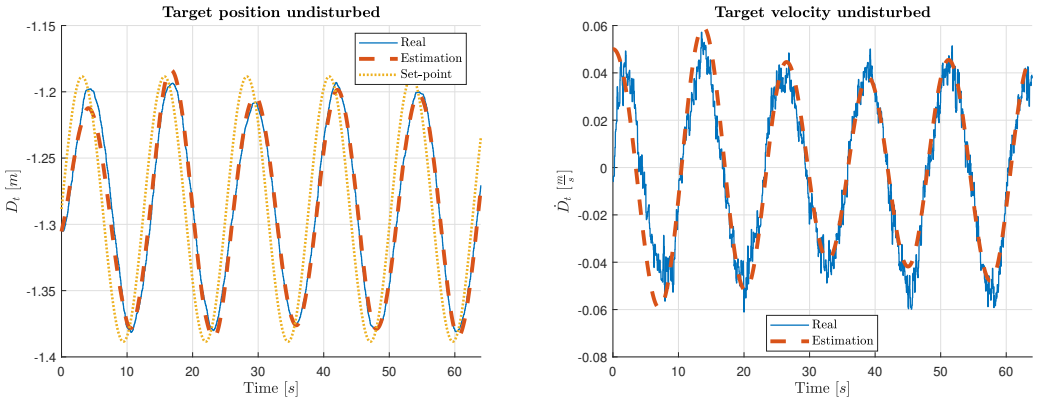


Figure 6.30: Target position and velocity estimation with initial guess A ("exact") (Table 6.9).

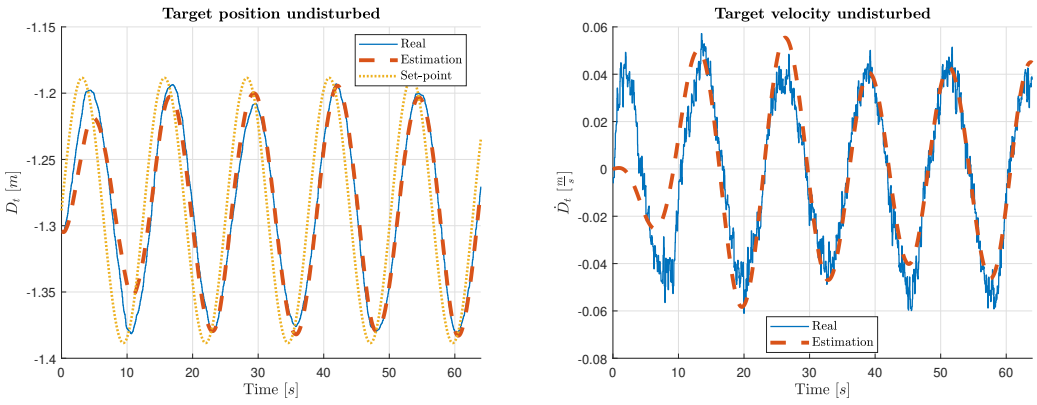


Figure 6.31: Target position and velocity estimation with initial guess B ("wrong") (Table 6.9).

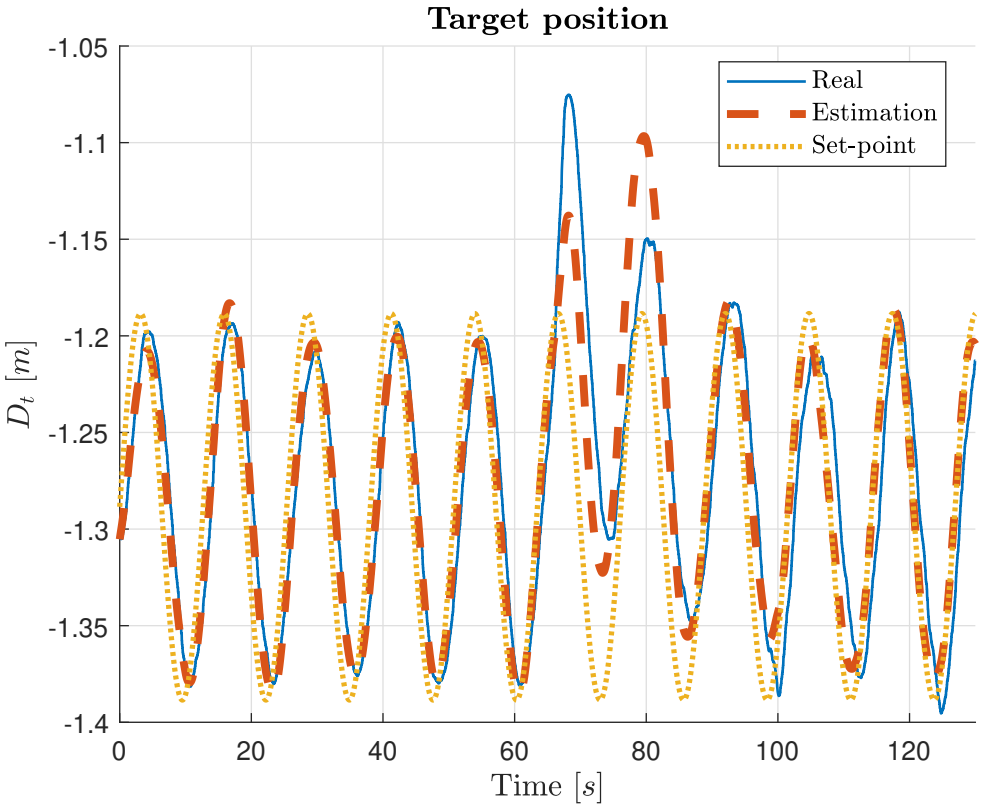


Figure 6.32: Target position estimation in undisturbed and disturbed conditions with initial guess A (Table 6.9).

again.
The error between estimated and real target position and velocity is shown in Figure 6.34. An increment is clearly visible in the disturbed case.

In Figure 6.35 the estimated parameters of the Fourier decomposition of the target motion are compared with the values that they would have in order to make equation (5.3) equal to equation (5.1) with set-point data of Table 5.1.

From the obtained results it follows that, for safety reasons, another check based on the estimation must be introduced in the landing algorithm: if the error ε_k, defined as the difference between measured and estimated velocity at the current instant, is greater than a chosen threshold, the landing procedure is paused and the follower maintains its position. In this way also in the case in which the follower is still, because of the poor quality of the estimate, at low relative distance from the target and the latter is moving upward, before it can touch the follower, the check on the vertical distance makes the follower disarmed and land without any problem.

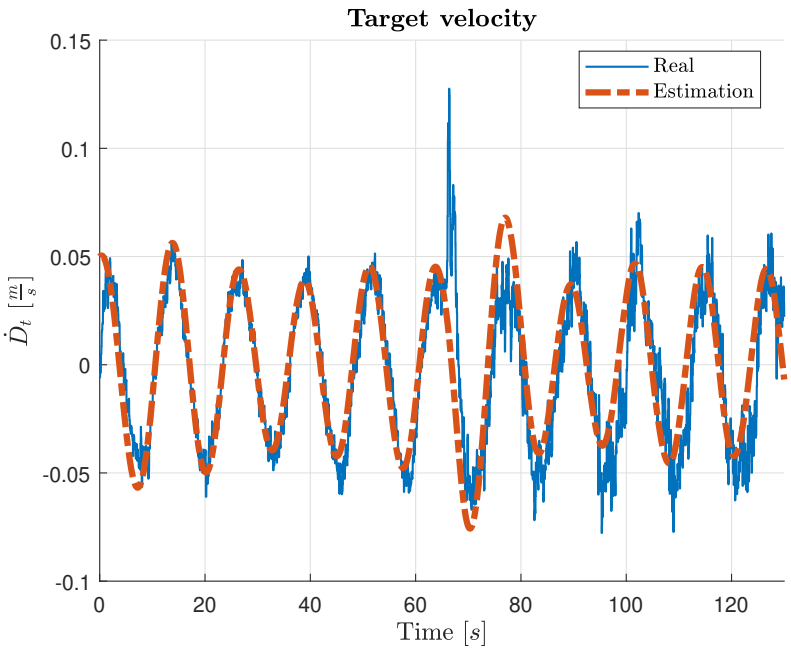


Figure 6.33: Target velocity estimation in undisturbed and disturbed conditions with initial guess A (Table 6.9).

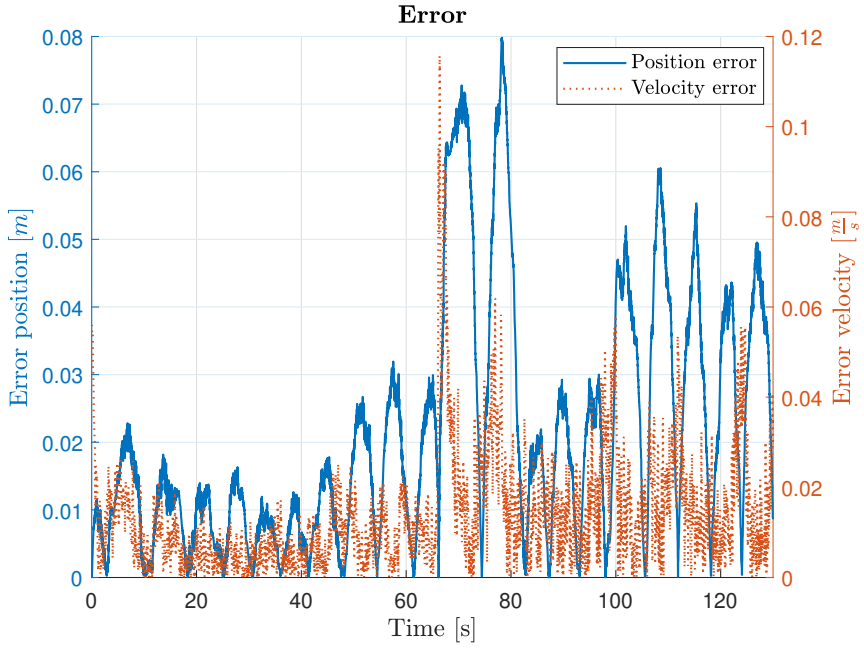


Figure 6.34: Target position and velocity error for the estimate with initial guess A (Table 6.9).

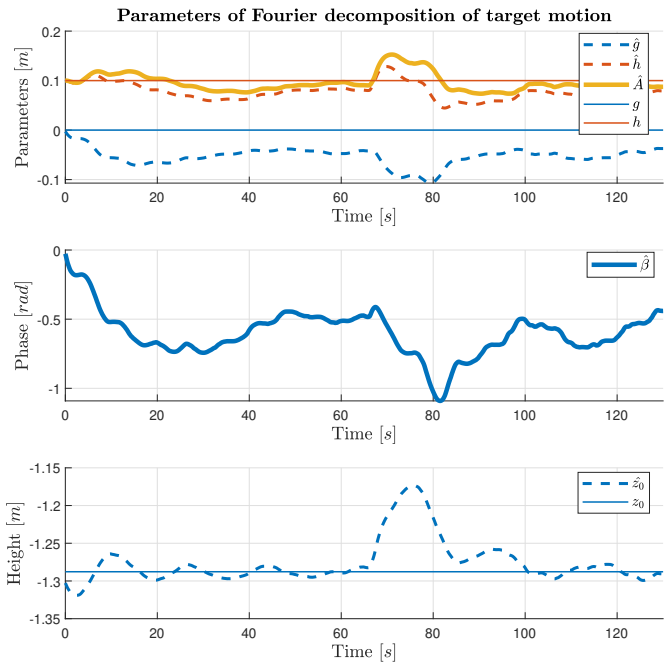


Figure 6.35: Estimated parameters with initial guess A (Table 6.9).

Finally, in Figure 6.36 simulation and real experiment results conducted with the parameters of Table 5.1 are compared: τ is delayed with respect to the simulation case because of follower disturbance.

6.5.2 Landing on oscillating target test

The landing algorithm with the oscillating target has been tested in the three cases reported in Table 6.10.

Test	Amplitude A $[m]$	Phase β $[deg]$	ω $[rad/s]$
LND A	0.1	0	0.5
LND B	0.2	0	0.5
LND C	0.1	0	0.7

Table 6.10: Data for the sinusoidal target motion for each test.

The parameters used are reported in Table 6.11. It must be noted that the maximum descent velocity has been reduced with respect to the in-plane experiments; this has been done so as to reduce the relative landing velocity (touch down) that might be higher in this case because the target is moving vertically.

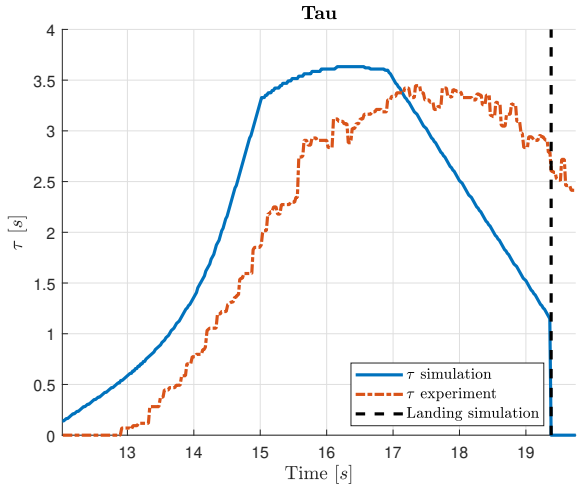


Figure 6.36: τ comparison between real experiment and simulation.

Parameter	Measurement unit	Value
Integration time step T_{int}	$[s]$	0.02
a_{max}	$[m/s^2]$	1
v_{max}	$[m/s]$	2
$v_{D_{max}}$	$[m/s]$	0.2
$a_{D_{min}}$	$[m/s^2]$	-0.1
$a_{D_{max}}$	$[m/s^2]$	0.1
ϵ_D	$[m]$	0.02
Target initial position	$[m]$	$[0\ -2\ -1.3]^T$
Follower initial position	$[m]$	$[0\ -3.5\ -2.8]^T$
Threshold on velocity estimation error	$[m/s]$	0.06

Table 6.11: Parameters for sinusoidal landing experiments.

In the LND A and LND B cases (Table 6.10) the landing is performed with success, the estimation of position and velocity are good and shown for the LND A case in Figure 6.37. The LND C case, at higher oscillating frequency, shows a

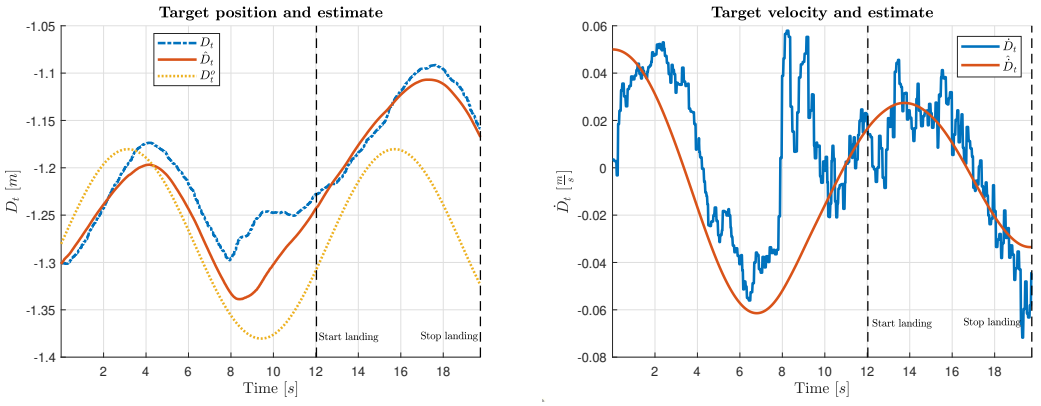


Figure 6.37: Position and velocity estimation of the target motion in LND A case (Table 6.10).

particular behaviour. This because the safety check on the quality of the velocity estimate is activated; in fact when the error on the velocity estimate exceeds the threshold (chosen empirically looking at simulation, Section 6.5.1) the landing is paused. This can be seen in Figure 6.38 where the landing trajectory computed by the bang-zero-bang algorithm presents an interval about 18 s in which the position set-point is constant.
The trajectory results in the time histories in Figure 6.39 and Figure 6.40. The position and velocity estimates are shown in Figure 6.41 and it can be seen that the quality of the estimate is not satisfactory.

A comparison between the three cases is shown in Table 6.12. In terms of horizontal position error all cases presents similar values and similar behaviour (Figure 6.42); the great differences are in the touch down velocity and in the time to land. The velocity has a strong dependency on the time at which the touch down occurs because it depends on the target motion. The longest time to land of the last case is due to the low quality estimate: in fact in Figure 6.43 it can be seen that case LND C is the one that exceeds the threshold also after the transient phase in which the target loses height because of the presence of the follower on it.

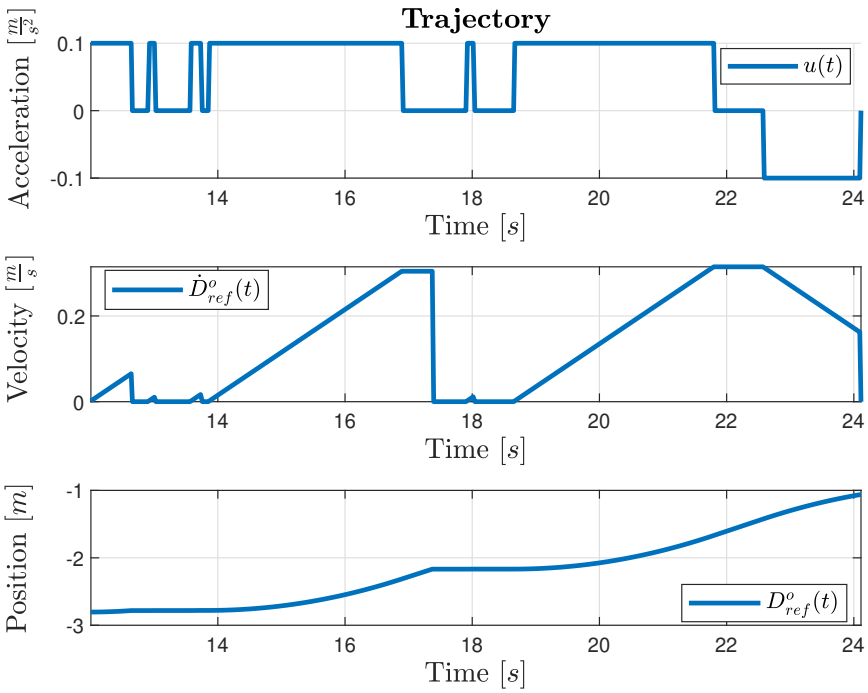


Figure 6.38: Landing trajectory computed by the algorithm in LND C case (Table 6.10).

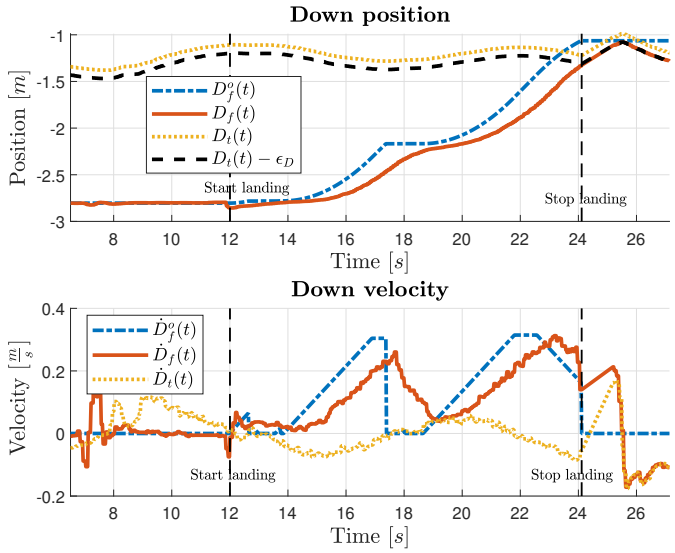


Figure 6.39: Down position and velocity for the entire duration of the experiment LND C (Table 6.10).

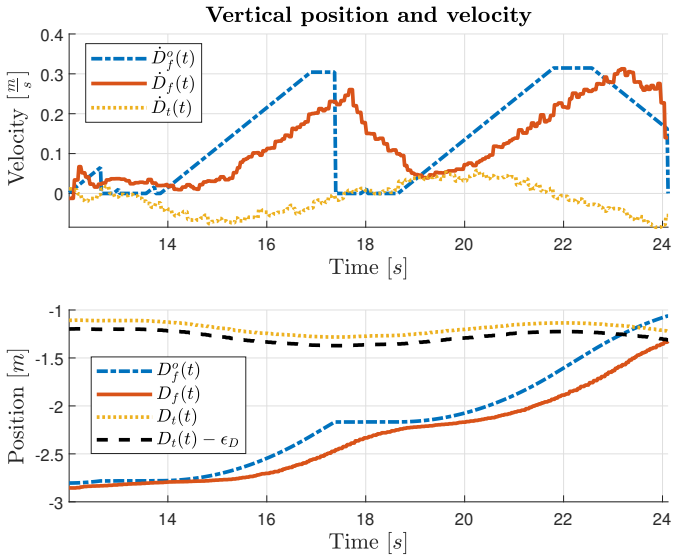


Figure 6.40: Down position and velocity during landing phase of experiment LND C (Table 6.10).

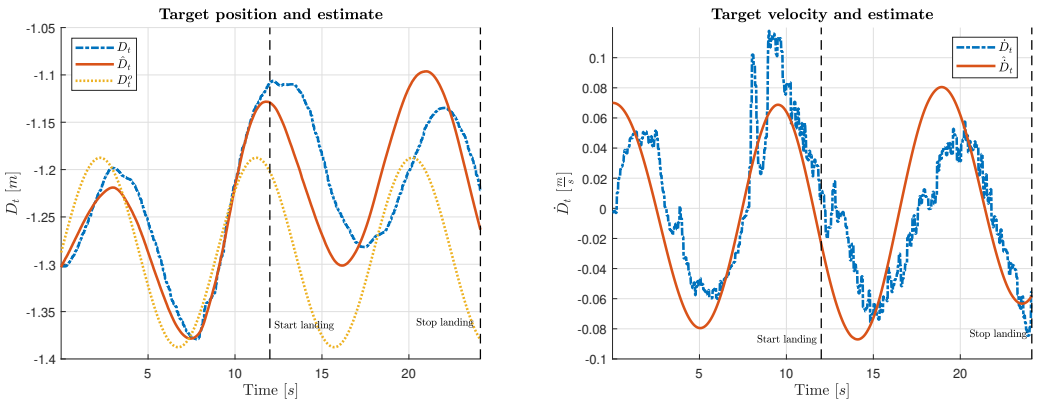


Figure 6.41: Position and velocity estimation of the target motion in LND C case (Table 6.10).

Test	Final error position $[m]$	$\dot{D}_r$ touch down $[m/s]$	Time to land $[s]$
LND A	0.0480	-0.2733	7.7230
LND B	0.0276	-0.2424	8.0956
LND C	0.0488	-0.1953	12.1084

Table 6.12: Results for the landing with target sinusoidal motion for each test.

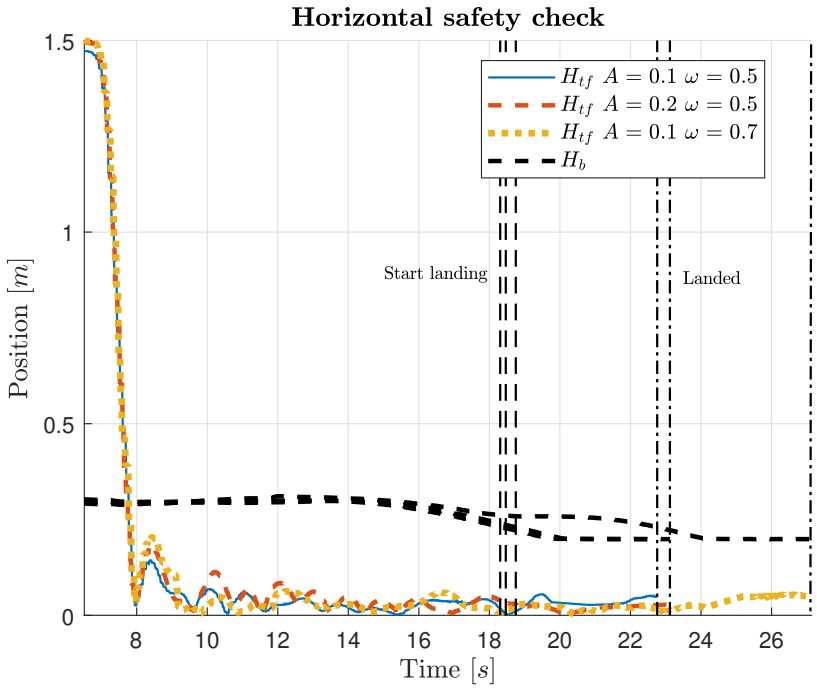


Figure 6.42: Time history of the horizontal position error during all the sinusoidal landings.

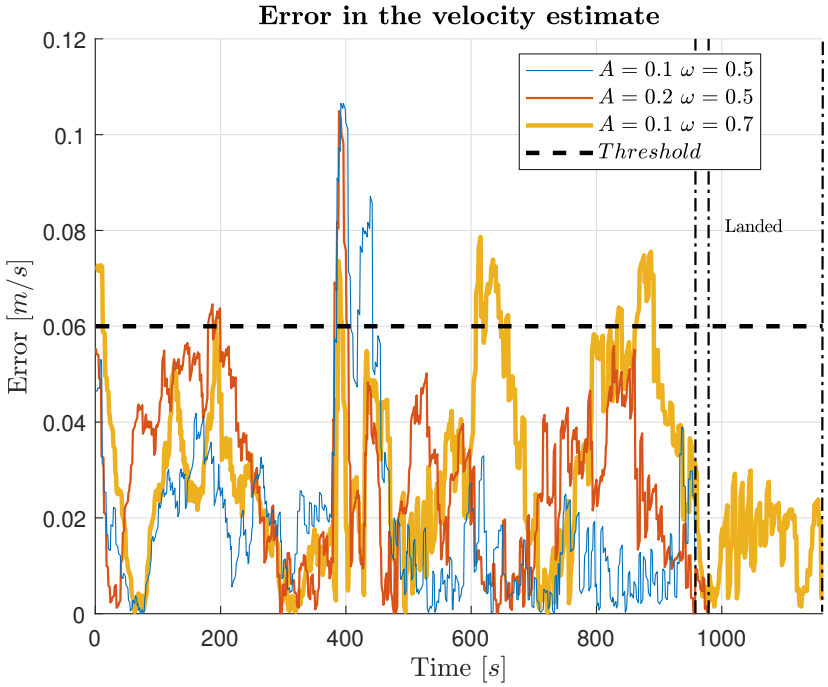


Figure 6.43: Error in the velocity estimates in the three sinusoidal landing conditions analysed.

Chapter 7

Conclusions

In this thesis, the air-to-air autonomous landing manoeuvre for multirotor UAVs has been studied. The problem of the interaction between more aircraft during flight is of great interest in UAV operations, such as search and rescue, surveillance and air-to-air automatic refuelling (AAAR).
The conducted activity starts from the previous work [13] that deals with autonomous landing of the follower on the target while the latter is hovering.
The purpose of this work has been to simulate, implement and experimentally validate tracking and landing guidance laws so as to perform an autonomous landing of the follower on the target while the latter is moving in the plane or oscillating vertically.
In the case of target moving in-plane, two different conditions have been investigated: the first one in a collaborative case in which the exact target state is known by the follower, the second one in a non-collaborative case, in which the follower has to track the target based only on relative position and velocity. The problem of landing a multirotor UAV over another one involves a number of non-trivial problems to be solved: because of aerodynamic interactions between the two drones, *e.g.*, the target is disturbed by the wake of the follower that flies above it, thus the control law gains of the two drones have been retuned in order to improve disturbance rejection performance; in sinusoidal landing an estimation of the target oscillatory motion in terms of position and velocity has been required.
Finally experimental activity has been conducted to validate the proposed algorithm: at the beginning, tests with the target on ground have been used to design requirements for the relative navigation system studying the impact of a degradation in frequency and signal-to-noise ratio of the position measurements, then the landing algorithms with target moving in-plane and oscillating vertically were successfully tested.

The following results are obtained:

- studying the effect of degradation in the rate at which the algorithm works and at which the set-points are sent to the drones, no clear tendency in

touch down velocity, final error position and time to land variation with frequency is identified;

- landing in presence of Gaussian noise on the position measurements sampled at the same rate as GPS measurements, *i.e.*, simulating the outdoor operating conditions, has been successfully conducted up to a standard deviation of 10 cm with grounded target;
- it has been possible to land with target moving in the plane with both the simple and the augmented position controllers: in particular with the simple position controller good behaviour in terms of landing velocity and time to land is shown but with final in-plane position error higher then the one obtained with augmented position controller; the latter is tested with two gain-sets with good performance for in-plane target velocity of 15 cm/s while with degraded performance at target velocity of 30 cm/s;
- Landing with vertically oscillating target has been successfully performed despite the inaccurate velocity estimates obtained in some conditions.

Starting from results obtained recommendations and possible future development are:

- study and test the landing in presence of Gaussian noise with hovering and moving target,
- improve the in-plane synchronization control law for the simple position controller, in particular trying to reduce the final in-plane position error for example with an integral term on the position error in the follower set-point,
- improve the in-plane synchronization control law for the augmented position controller at high velocity for example implementing Adaptive Robust Control (ARC) or Model Predictive Control (MPC),
- try alternative motion estimation methods so as to obtain a better estimate of the target motion for the landing with oscillating target,
- study the relative navigation problem by means of different technologies, *e.g.*, assisted and unassisted vision, ultrasound, radio,
- try to develop and validate a new set of guidance, navigation and control laws enabling air-to-air UAV landing in a realistic scenario such as arising, *e.g.*, in the framework of a search and rescue mission, so as to perform an autonomous landing during outdoor flight test.

Bibliography

[1] Chrobotics.
http://www.chrobotics.com/library/understanding-euler-angles.

[2] Px4 documentation.
https://docs.px4.io/en/airframes/airframe_reference.html.

[3] L. Meier. Pixhawk Mini (Discontinued) parameter list documentation.
https://docs.px4.io/en/flight_controller/pixhawk_mini.html.

[4] Friendly ARM. NanoPi NEO Air.
http://wiki.friendlyarm.com/wiki/index.php/NanoPi_NEO_Air.

[5] M. Hassanalian and A. Abdelkefi. Classifications, applications, and design challenges of drones: a review. *Progress in Aerospace Sciences*, 91:99–131, 2017.

[6] P. Giuri, A. Marini Cossetti, M. Giurato, D. Invernizzi, and M. Lovera. Air-to-air automatic landing for multirotor uavs. In *5th CEAS Conference on guidance, navigation and control (EuroGNC)*, Milan, Italy, April 2019.

[7] B. Hu, L. Lu, and S. Mishra. A control architecture for time-optimal landing of a quadrotor onto a moving platform. *Asian Journal of Control*, 20(5):1701–1712, 2018.

[8] B. Hu, L. Lu, and S. Mishra. A control architecture for fast and precise autonomous landing of a vtol uav onto an oscillating platform. In *American Helicopter Society 71st Annual Forum*, pages 4–7, 2015.

[9] B. Hu, L. Lu, and S. Mishra. Fast, safe and precise landing of a quadrotor on an oscillating platform. *2015 American Control Conference (ACC)*, pages 3836–3841, 2015.

[10] A. Borowczyk, D.-T. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny. Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle. *IFAC-PapersOnLine*, 50(1):10488–10494, 2017.

[11] F. Alarcón, M. García, I. Maza, A. Viguria, and A. Ollero. A precise and gnss-free landing system on moving platforms for rotary-wing uavs. *Sensors*, 19(4):886, 2019.

[12] J. Kim, Y. Jung, D. Lee, and D. H. Shim. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252. IEEE, 2014.

[13] A. Marini Cossetti and P. Giuri. Air-to-air automatic landing for multirotor UAVs, 2018. Master's thesis.

[14] B.L. Stevens, F.L. Lewis, and E.N. Johnson. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. Wiley, third edition, 2015.

[15] R.W. Prouty. *Helicopter performance, stability, and control*. PWS Engineering, 1986.

[16] L. Lu and B. Yao. A performance oriented multi-loop constrained adaptive robust tracking control of one-degree-of-freedom mechanical systems: Theory and experiments. *Automatica*, 50(4):1143–1150, 2014.

[17] G. van der Veen, J.-W. van Wingerden, M. Bergamasco, M. Lovera, and M. Verhaegen. Closed-loop subspace identification methods: an overview. *IET Control Theory and Applications*, 7(10):1339–1358, 2013.

[18] Px4 parameter list documentation. `https://dev.px4.io/en/advanced/parameter_reference.html`.

[19] D. Lee, T. Ryan, and H. J. Kim. Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. In *IEEE International Conference on Robotics and Automation, Saint Paul, USA*, 2012.

[20] A. Monneau, N. M'Siridi, S. Mavromatis, G. Varra, M. Salesse, and J. Sequeira. Adaptive Prediction for Ship Motion in Rotorcraft Maritime Operations. In *5th CEAS Specialist Conference on Guidance, Navigation and Control (EuroGNC)*, MILAN, Italy, April 2019.

[21] A. Quarteroni and F. Saleri. *Calcolo scientifico: Esercizi e problemi risolti con MATLAB e Octave*. Springer Milan, 2008.

[22] L. Meier. MAVLink. `https://mavlink.io/en/`.

[23] L. Meier. PX4 firmware. `https://github.com/PX4/Firmware`.

[24] Drone Control. Qgroundcontrol. `http://qgroundcontrol.com/`.

[25] Open Source Robotic Foundation. Ros. http://wiki.ros.org/Documentation.

[26] L. Meier. Mavros. https://github.com/mavlink/mavros.

[27] Motive. Optitrack. https://optitrack.com/products/motive/.

[28] Optitrack camera. https://tracklab.com.au/products/hardware/optitrack-prime-41/#prettyPhoto.

[29] Microsoft. Windows 10 pro. https://www.microsoft.com.

[30] Canonical. Ubuntu 16.04 lts. https://ubuntu.com.

[31] D. Del Cont-Bernard. Ground effect analysis for a quadrotor platform, 2016. Master's thesis.