Politecnico di Milano
School of Industrial and Information Engineering
Master of Science in Aeronautical Engineering

Autonomous landing of an UAV on a moving ground vehicle

Advisor: Prof. Marco LOVERA
Co-Advisor: Prof. Jérôme LE NY

Thesis by:
CINELLI Tommaso Matr. 918809

Academic Year 2019–2020

Ai miei genitori e ai miei fratelli.

Acknowledgments

Nulla v'è infatti che allieti tanto
l'animo quanto gli esempi delle
virtù che brillano nel carattere
delle persone con cui viviamo.

Marco Aurelio

Firstly, I would like to express my thanks to Prof. Jérôme Le Ny for his help and guidance during all the work and Prof. Marco Lovera for the support.

I would like to thank all the people that have been part of this last incredible year in Montreal. To all the "ZUM family", Bea, Carla, Eloy, Giuseppe, Jeremy, Silvia, Miriam, Ilaria, Marta, Sara, Marina, Paricia and Julia, with whom I spent one of the most amazing years of my life and that made the cold canadian winter seem to be warmer. To the Polytechnique friend and project mate Marc, that I would like to thank for introducing me to the topic of this thesis.

Uno dei più grandi ringraziamenti ai miei amici nonché coinquilini Carlo e Carlo (a voi scegliere l'ordine), con la quale ho condiviso ogni giorno e notte, studio e vacanza, lavoro e svago in questi anni universitari, e che sono stati fondamentali nella mia crescita non solo accademica, ma soprattutto come persona.

All'incredibile team Move-ez, con la quale abbiamo passato gioie, soddisfazioni, ansie e traguardi in una delle esperienze che rimarrà sempre con noi. A Daniela, instancabile ottimista e lavoratrice, sempre pronta a tirare su il morale del gruppo, Vittorio, con la testa sempre da qualche parte in giro per il mondo e col pensiero costantemente a qualche idea nuova, e Mario, compagno simbiotico di università, esempio di come con creatività e tenacità si possa arrivare ovunque.

Allo zio Rocco, ovvero lo zio ingegnere, che ha sempre creduto e supportato questa carriera nel mestiere che spero essere all'altezza di ereditare e continuare, seguendo le orme del patriarca.

Ai miei fratelli, che, anche se in diverse parti del mondo, mi sono sempre stati vicini e che rappresentano da sempre un modello. A Eleonora, costante e tenace, sempre alla ricerca della conoscenza profonda delle cose, a Lorenzo, inarrestabile e dinamico, sempre capace di mettersi in gioco con nuove sfide, e a Francesco, alla quale passo il testimone universitario, coraggioso e determinato nelle cose che gli piacciono.

Alla fine perché più importante, il ringraziamento più sentito va ai miei genitori che hanno permesso tutto questo. A mia madre, esempio di come perseveranza, passione e dedizione debbano essere le doti che ci guidano costantemente nella nostra vita, sia nel lavoro che nella famiglia. A mio padre, che mi ha trasmesso l'interesse per le gioie della vita e l'importanza di espandere sempre gli orizzonti della cultura.

Abstract

The recent progress in drone technology has fostered interest towards Unmanned Aerial Vehicles (UAVs). However, the relatively short battery life limits their adoption for some activities. Hence, deploying and recovering UAVs from ground vehicles (GVs) could extend vehicle's authonomy and open up more efficient and innovative applications. Drones could be carried by trucks, take-off when close to the location of reconnaissance or delivery, and autonomously land on the carrier, with the possibility of being recharged between operations sites. Landing on a small available area represents the most critical phase, thus requiring a precise and reliably system for the localization of the landing pad. Current standard outdoor UAV's navigation systems, based on Inertial Measurement Unit (IMU) and Ground Navigation Satellite Systems (GNSS), may not result to be as accurate as desired, thus opening the possibility of a vision-based system for locating the landing site.

The purpose of this thesis is to investigate the problem of the autonomous landing of an UAV on top of a moving ground vehicle, using commercially available and relatively low-cost sensors. The study is focused on providing a guidance law to perform the mission, as well as a navigation system which uses visual information to support the estimation of position and velocity. Finally, the system is tested and validated through computer simulation using MATLAB and Simulink.

Sommario

Recenti sviluppi tecnologici e una lunga lista di potenziali applicazioni, che includono la consegna tramite droni, sorveglianza, ricerca e soccorso, hanno portato ad un crescente interesse per gli aeromobili a pilotaggio remoto (UAV). Tuttavia, la limitata autonomia della batteria rappresenta una grande sfida per la possibilità di operazioni di lunghe distanze. Per questo motivo, il rilascio e recupero da parte di veicoli di terra potrebbe permettere l'estensione della durata della missione degli UAV e un loro impiego più efficiente. Il drone potrebbe essere trasportato da un furgone, decollare quando vicino alla posizione di ricognizione o consegna, dopodiché atterrare autonomamente sul veicolo che li ha portati e possibilmente essere ricaricato durante il trasporto tra le varie zone di operazione. La fase più critica della missione è rappresentata dall'atterraggio sulla limitata area del furgone, necessitando un sistema sufficientemente preciso e affidabile per localizzare la zona di atterraggio. Attuali sistemi di navigazione, basati su piattaforme inerziali (IMU) o sistemi di navigazione satellitare (GNSS), potrebbero non risultare abbastanza accurati questo compito, aprendo perciò la possibilità dell'utilizzo di un sistema di visione per localizzare il sito di atterraggio.

Lo scopo di questa tesi è investigare il problema dell'atterraggio autonomo di un UAV sopra di un veicolo di terra in movimento, utilizzando sensori già disponibili nel mercato e di basso costo. Lo studio è focalizzato nel presentare un sistema di guida che possa svolgere la missione e un sistema di navigazione che utilizza informazioni visive per stimare posizione e velocità. Infine, il sistema è testato e validato attraverso una simulazione al computer, utilizzando MATLAB e Simulink.

Contents

List of Figures

List of Tables

Introduction

The interest for Unmanned Aerial Vehicles (UAVs) has increasingly grown in recent years due to their variety of applications, e.g., site surveillance, aerial monitoring or search-and-rescue operations. In particular, one of the promising perspectives is their use related to parcel delivery for commercial purposes: Amazon, the world's largest online retailer, has been working on the *Prime Air* project since 2016 [3], aiming to develop an innovative drone delivery service with operations starting in the next few years. However, the UAVs' short battery autonomy restricts the service only to areas relatively close to dedicated Prime Air warehouses. This opens the possibility for new hybrid solutions, such as the combination of UAVs and mobile ground vehicles (GVs). Most notably, UPS, the American multinational package delivery company, tested in 2017 a shipment using a drone deployed and recovered from the roof of a delivery truck [1] and illustrated at Figure 1. It was shown how the versatility of UAVs can be combined with the longer range autonomy of a truck. Nevertheless, the recovery could have been made only with the vehicle stopped at the side of the road and waiting for the drone.

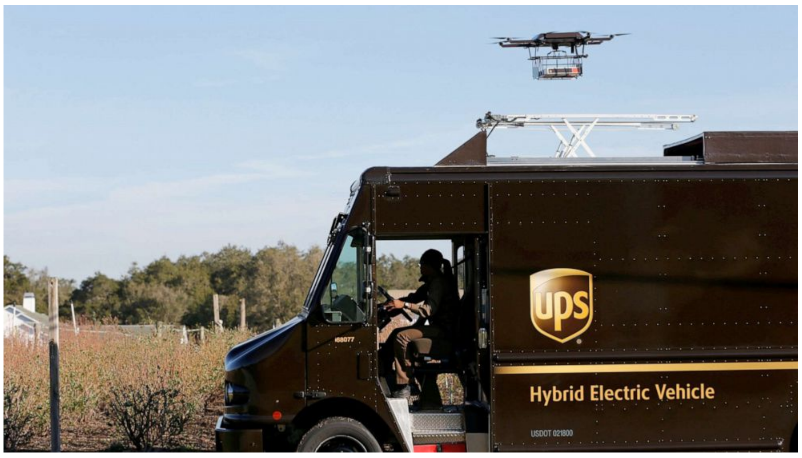

Figure 1: UPS drone delivery service (from [1])

A huge potential of time optimization could be made if the UAV could return to the GV still moving along the street and going to the next delivery place. This is the starting point of this work, where the strategy of a system that can be used for an application of this type is going to be studied.

In this thesis, the objective is to develop a set of guidance, navigation and control laws enabling an autonomous landing of an UAV on top of a moving ground vehicle. This is going to be achieved considering only commercially available and relatively low-cost sensors. In particular, a vision system using a camera installed on the UAV will provide position information used to locate the GV. Moreover, the system should be able to perform state estimation even in the case of temporarily loss of information from the camera, integrating other sensors through an Extended Kalman filter.

The system is finally tested by means of a computer simulation, that tries to simulate in the most precise way a possible real application of the system.

State of the art

In the literature there are many existing works concerning the autonomous landing of UAVs on marine, ground vehicle or air moving platforms. In this section, an overview of some recent studies is provided.

Borowczyk et al. [4] investigated the problem of autonomous landing of a quadcopter on a high speed moving ground vehicle, equipped with a landing pad, on which a visual fiducial AprilTag [5] was placed. The system architecture to achieve this goal consisted in:

- a Kalman filter for relative position and velocity estimation, using commercially available and relatively low-cost sensors. This allowed the integration of the six-degrees-of-freedom visual estimate given by the detection of the AprilTag from a gimballed camera, GPS data and IMU;
- a Proportional Navigation (PN) based guidance law, commonly used to guide missile trajectories, for the long range approach phase;
- a Proportional-Derivative (PD) controller for the terminal landing phase, using acceleration and attitude controls.

The system was experimentally tested for the landing of a micro aerial vehicle (MAV) on a moving car travelling at a speed of 50 km/h.

Gozzini [6] investigated the problem of automatic landing of a small UAV on a multi rotor carrier drone moving along a circular trajectory with constant speed. A trajectory generation module provided the reference set-points for the UAV to track, ensuring the horizontal alignment with the landing pad and a time-optimal vertical trajectory, through a bang-zero-bang algorithm. The autonomous landing was successfully validated through experimental activity in the Flying Arena for

Rotorcraft Technologies (FlyART) of the Aerospace Systems and Control Laboratory (ASCL) of Politecnico di Milano.

Gonçalves et al. [7] proposed an approach for an autonomous UAV landing using velocity vector field as closed-loop strategy to guide the drone towards the landing point. The method did not require global localization, but relative position estimation using two nested ArUco [8] markers, which were identified using a vehicle's on-board down-facing camera.

Kim, Jung et al. [9] proposed a vision-based landing system for a quadrotor on a moving platform. An Unscented Kalman filter estimated position and velocity of the target using simple color blob detection by a smartphone as viable on-board image acquisition and computation platform. The system was validated by outdoor flight test.

Thesis structure

The thesis is organized as follows:

- In Chapter 1, modeling and simulation of a multirotor UAV and a GV is presented; in particular reference frames, rotation formalism, flight dynamics of the UAV and kinematic model of the GV are described. Finally, the notation used for the rest of the work is presented.
- In Chapter 2, the camera model and the computer vision algorithm used to estimate relative position between UAV and GV is discussed. The mathematical formulation used to simulate and process the frames seen from the camera is described.
- In Chapter 3, the basic principles of the integrated navigation system are provided. Sensors' models, Kalman filter algorithm and navigation system set-up are presented.
- In Chapter 4, the control system architecture to perform the autonomous landing is described. The attention is focused on the trajectory generation module responsible for generating the set-points to be tracked by the UAV.
- In Chapter 5, a description of the simulation environment in MATLAB and Simulink is presented. Finally, the results of the simulation are reported and discussed.

Chapter 1

Problem statement

This first chapter introduces the adopted conventions and formalisms used in this work in order to avoid ambiguities. Although mathematical symbols, notations and definitions are more or less standardized in literature, many authors differ in terminologies: for this reason, frames of reference, coordinate systems and rotation formalisms are presented. Therefore, models for a multirotor Unmanned Aerial Vehicle (UAV) and a Ground Vehicle (GV) are described and finally architectures of the navigation and the control system used for the landing problem are shown.

1.1 Definitions of reference frames

In order to describe the dynamics of a rigid body, at least an inertial and a body frame are needed. In addition to these last, sensor's frame with respect to which instruments express their measurements should be presented as well.

A generic reference frame in $\mathbb{R}^3$, characterized by 3 orthonormal axis $(\hat{x}, \hat{y}, \hat{z})$ and origin O will be denoted by the notation in brackets $\{a\}$. Quantities measured with respect to that specific frame are indicated by a superscript $(.)^a$. Definitions of adopted reference frames are presented in this section.

1.1.1 ECI and ECEF frames

An inertial frame is a reference frame in which Newton's laws of motion apply. It is therefore not in acceleration, but may be in uniform linear motion. The Earth Centered Inertial (ECI) can be considered of this class. It has its origin in the center of the Earth, *x-axis* pointing toward the *vernal equinox* (the intersection between the equatorial plane and the orbit plane around the Sun), and *z-axis* along the Earth's spin axis. The *y-axis* completes the right-handed coordinate system.

The Earth Centered Earth Fixed (ECEF) is the frame defined up to a rotation of the ECI around the *z-axis* with the same Earth's rotation rate of

$$\omega_{ie} \approx \left(\frac{1+365.25 \text{ cycle}}{(365.25)(24)\text{hr}}\right)\left(\frac{2\pi\text{rad/cycle}}{3600\text{sec/hr}}\right) = 7.292115 \times 10^{-5}\frac{\text{rad}}{\text{sec}},$$

considering both the daily rotation around the Earth and the year revolution around the Sun. The ECEF frame results to be more practical to express a point lying on a fixed position of the Earth's surface.

The *x-axis* of the ECEF points the intersection between the Greenwich meridian and the equatorial plane. Coordinates with respect to ECEF frame are often expressed using geodetic coordinates of *longitude, latitude* and *altitude* (ϕ, λ, h). These last are ellipsoidal coordinates that use the approximation of the Earth as a *geoid* (i.e. the rotation of an ellipse around its minor axis). The geodetic surface of the Earth can be imagined as the mean shape that the Earth would take if the solid surface would be completely covered with sea water.

Although the real shape of the Earth is not a geoid, it represents a good approximation of its shape. One of the most used world geodetic model is the WGS84, which is also the model used for the Global Positioning System (GPS).

1.1.2 Local geodetic NED frame

The North-East-Down (NED) frame conventionally expresses the coordinates of points that remain in a delimited region on the surface of the Earth. It is determined by fitting a tangent plane to the geodetic reference ellipsoid at a point of interest (the origin). The *x-axis* points to true North, the *z-axis* points to the ground and perpendicular to the tangent plane, the *y-axis* is determined accordingly and points East. For applications that involve local navigation over a short time interval of seconds, like the problem under study, the assumption of flat, non-rotating Earth can be made. This helps to simplify equations because NED is considered as inertial reference frame. It will be denoted by $\{i\}$ and with origin O. Illustration of ECI, ECEF and NED frames are shown in Figure 1.1.

1.1.3 Body frame

It is a frame rigidly attached at a fixed point to the vehicle of interest. The *x-axis* points towards the forward direction of the vehicle, the *z-axis* points to the bottom of the vehicle making the xz plane coincident with the symmetry plane, the *y-axis* completes the right-handed orthogonal coordinate system.

Since the landing problem considers two moving vehicles, we define:

- $\{d\}$: the frame of the *drone* with origin at point D (the center of gravity of the UAV),
- $\{t\}$: the frame of the *target* ground vehicle with origin at point T (the center of gravity of the car).

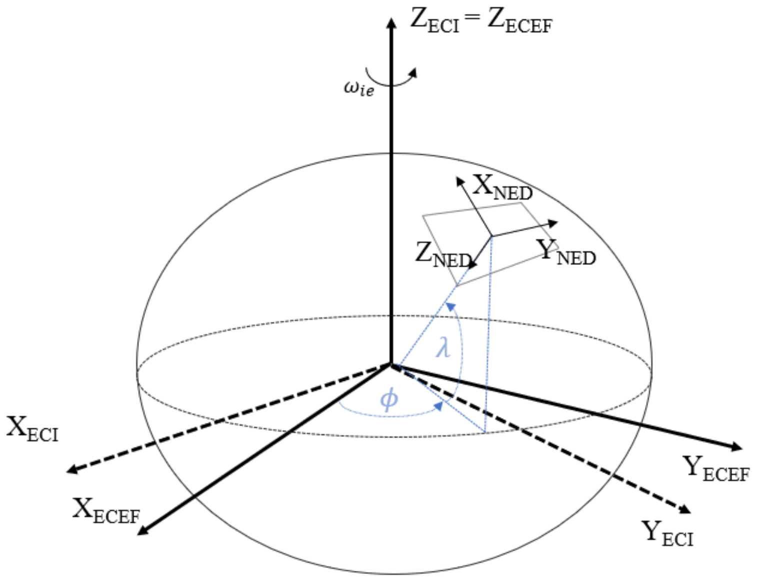

Figure 1.1: Reference frames ECI, ECEF and NED

1.1.4 Sensors and camera frames

Sensors of a moving vehicle can have their own reference frame that may differ in orientation and position from the body frame. In the case of *strap down* sensors, they are rigidly attached to the vehicle and it is sufficient to know position and rotation from body frame when installed. This will be the case of all the sensors that will be considered from now on, except for the camera imaging sensor.

The camera is installed on the UAV and is supposed to be gimballed, with three-axis rotation degree of freedom. The camera reference frame is denoted $\{c\}$, centered at origin C with *z-axis* pointing towards the direction of sight of the camera and *x-axis* always parallel to the North-East plane.

The illustration of body frames and camera frame can be seen in Figure 1.2.

1.2 Reference frame transformation

This section presents the methods to transform vectors in different coordinate systems.

1.2.1 Direction cosine matrix

Let $\mathbf{v}$ be a generic vector, $\{a\}$ a frame with origin O and axis $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ and $\{b\}$ a frame with origin O' and axis $(\mathbf{I}, \mathbf{J}, \mathbf{K})$. The vector $\mathbf{v}$ can be expressed as

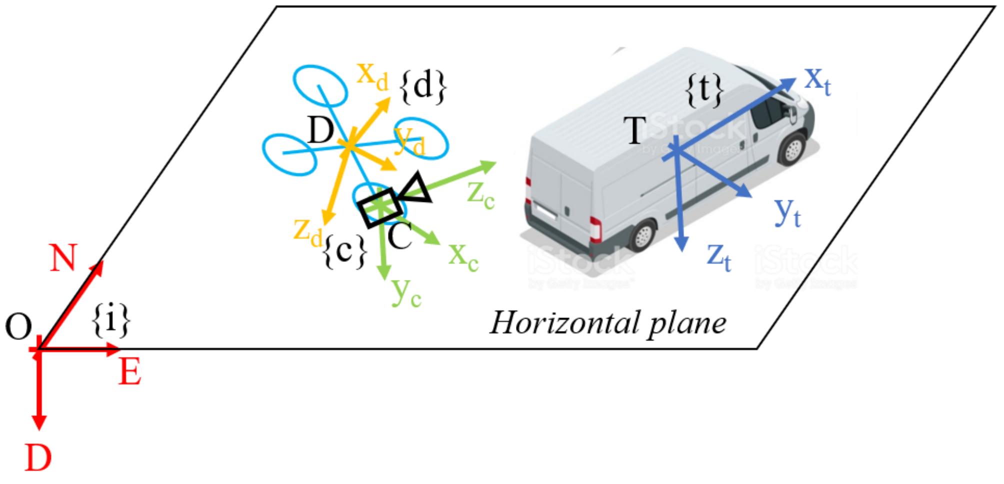

Figure 1.2: Body and camera reference frames

$\mathbf{v}^a = [x^a, y^a, z^a]^T$ in coordinates of $\{a\}$ or $\mathbf{v}^b = [x^b, y^b, z^b]^T$ in coordinates of $\{b\}$. Hence, vector $\mathbf{v}^b$ can be written as

$$\mathbf{v}^b = x^b\mathbf{I} + y^b\mathbf{J} + z^b\mathbf{K}.$$

To write $\mathbf{v}^b$ of last equation in coordinates of $\{a\}$, it is sufficient to express vectors $(\mathbf{I}, \mathbf{J}, \mathbf{K})$ with respect to $\{a\}$

$$\begin{aligned} \mathbf{v}^a &= x^b\mathbf{I}^a + y^b\mathbf{J}^a + z^b\mathbf{K}^a = [\mathbf{I}^a\,\mathbf{J}^a\,\mathbf{K}^a]\mathbf{v}^b \\ &= R^a_b\mathbf{v}^b. \end{aligned} \tag{1.1}$$

Matrix $R^a_b = [\mathbf{I}^a\,\mathbf{J}^a\,\mathbf{K}^a]$ is called *direction cosine matrix* (or rotation matrix) and it allows to pass from coordinate representation of a vector in frame $\{b\}$ to $\{a\}$. An interesting aspect of the notation of R^a_b is that it has a quite intuitive mnemonic aspect: symbols that appear both as subscript in matrix and superscript in adjacent vector *cancel* each other.

1.2.2 Euler angles

Euler angles constitute a practical parametrization of rotation matrices to pass from inertial $\{i\}$ to a generic body frame $\{b\}$. Rotation matrix R^i_b is obtained by the multiplication of three standard rotation matrices around (x, y, z) axis, given by

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & -\cos\phi \end{bmatrix}, R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, R_z = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Euler angles (ϕ, θ, ψ) are commonly known as roll, pitch and yaw, respectively. There are many possible choices on the order of rotation, for this work it is used the ZYX sequence given by $R_b^i = R_z R_y R_x$, so

$$R_b^i = \begin{bmatrix} c\psi c\theta & -c\theta s\psi & s\theta \\ c\phi s\psi + c\psi s\phi s\theta & c\phi c\psi - s\phi s\psi s\theta & -c\theta s\phi \\ s\phi s\psi - c\phi c\psi s\theta & c\psi s\phi + c\phi s\psi s\theta & c\phi c\theta \end{bmatrix}, \tag{1.2}$$

where, for compactness of notation $c\alpha = \cos\alpha$ and $s\alpha = \sin\alpha$.

The derivatives of the Euler angles are defined as Euler rates $(\dot{\phi}, \dot{\theta}, \dot{\psi})$. The relation between Euler rates and angular rates in body frame $\Omega^b = [p, q, r]^T$ is given by

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix},$$

where matrix on the right will be denoted W_n

$$W_n = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{1.3}$$

1.3 Alternative coordinate representations

Coordinates of points and vectors in $\mathbb{R}^3$ are commonly expressed by means of 3-dimensional column vectors. Two alternative representations using augmented and homogeneous coordinates are presented, as they will be particularly useful for the rest of the discussion of the thesis.

1.3.1 Augmented coordinates

The definition of a vector $\mathbf{v}$ in the Euclidean sense in physics (sometimes referred as *free vector*), is an entity endowed with a magnitude and a direction, but not located at a specific place. For example, velocity, forces and accelerations are described by vectors. Thus, if $\mathbf{v}^a$ and $\mathbf{v}^b$ represent the same vector expressed in two different reference frames related by pure translation, $\mathbf{v}^a = \mathbf{v}^b$ since the rotation matrix R_b^a is equal to the identity matrix.

A point in the Euclidean space (sometimes referred as *bounded vector*), is an entity defined by a magnitude, a direction and also an origin. Hence, if $\{a\}$ is a an orthonormal coordinate system with origin O and axis $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, the point P^a is the unique triplet (x_P, y_P, z_P) that allows to express point P as a linear combination of the basis vector of $\{a\}$

$$P^a = (P - O)^a = x_P \mathbf{i} + y_P \mathbf{j} + z_P \mathbf{k}.$$

Its definition is strictly related to the origin of the coordinate system with respect to which the point is described. If considering another coordinate system $\{b\}$ characterized by a translation of the origin to point O' and rotation of the axis by matrix R_a^b, point P is described by coordinates $P^b = [x'_P, y'_P, z'_P]^T$ related with P^a by the expression

$$\begin{aligned} P^b &= (P - O')^b = R_a^b (P - O')^a = R_a^b (P - O)^a + R_a^b (O - O')^a \\ &= R_a^b P^a + O^b. \end{aligned} \tag{1.4}$$

Unlike vectors, the geometric transformation between reference frames of points requires not only the application of a matrix multiplication, but also a vector addition $O^b = (O - O')^b$.

It would be convenient to handle points and vectors in the same way by using only one operator. This can be achieved by using a different set of coordinates, called the *augmented coordinates*. In robotics literature they are often called *homogeneous coordinates*, however it won't be used this notation as not to confuse with homogeneous coordinates described at Section 1.3.2 used for the projective model of the camera [10].

Taking a point $P = [x, y, z]^T$ and a vector $\mathbf{v} = [v_x, v_y, v_z]^T$ in cartesian coordinates, their expression in augmented coordinates is defined as

$$\bar{P} := [x, y, z, 1]^T, \tag{1.5}$$

$$\bar{\mathbf{v}} := [v_x, v_y, v_z, 0]^T, \tag{1.6}$$

obtained by adding a fourth dimension coordinate with value 1 if expressing a point or value 0 if a vector. From now on, elements in augmented coordinates are denoted by the bar symbol $\bar{(.)}$. Points and vectors expressed in this set of coordinates keep the following properties:

- Sums and differences of vectors are vectors
- The sum of a vector and a point is a point
- The difference between two points is a vector
- The sum of two points is meaningless

Considering the frame transformation from $\{a\}$ to $\{b\}$ discussed previously the *augmented representation* (4x4) matrix is defined as

$$T_a^b = \begin{bmatrix} R_a^b & O^b \\ 0_{1x3} & 1 \end{bmatrix}.$$

Hence, the transformation of a point like in equation (1.4) can be easily represented using augmented coordinates

$$\bar{P}^b = \begin{bmatrix} P^b \\ 1 \end{bmatrix} = \begin{bmatrix} R_a^b P^a + O^b \\ 1 \end{bmatrix} = \begin{bmatrix} R_a^b & O^b \\ 0_{1x3} & 1 \end{bmatrix} \begin{bmatrix} P^a \\ 1 \end{bmatrix} = T_a^b \bar{P}^a.$$

For a generic vector $\mathbf{v}$, the transformation will be

$$\bar{\mathbf{v}}^b = \begin{bmatrix} \mathbf{v}^b \\ 0 \end{bmatrix} = \begin{bmatrix} R_a^b & O^b \\ 0_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v}^a \\ 0 \end{bmatrix} = T_a^b \bar{\mathbf{v}}^a.$$

Augmented coordinates represent a unified tool to operate with both points and vectors using the same transformation (a matrix multiplication).

1.3.2 Homogeneous coordinates

Homogeneous coordinates (or projective coordinates) are a special set of coordinates used in projective geometry and with particular importance for computer vision. Projective geometry will be discussed in Section 2.2.1 to explain the image creation process of a standard camera and homogenous coordinates will constitute an useful representation to write projective transformations (non-linear in Euclidean space) by means of linear matrix multiplications.

A point $p = [x, y]^T$ in the Euclidean space $\mathbb{R}^2$ is expressed in projective space $\mathbb{P}^2$ as a new set of homogeneous coordinates defined as $\tilde{p} = [x, y, 1]^T$. The main property of homogeneous coordinates, denoted by the upper tilde symbol $(\tilde{.})$, is that two points related up to a scale factor represent the same point. Hence, it is said that point $\tilde{p}_1 = [x, y, 1]^T$ is *equivalent* to point $\tilde{p}_2 = [2x, 2y, 2]^T$, and more generally, for any scalar $k \neq 0$

$$\tilde{p}_1 = [x, y, 1]^T \sim [kx, ky, k]^T = \tilde{p}_2,$$

where symbol $(\sim)$ represents the *equivalent* relation.

To notice that to retrieve a point from the Projective space to the Euclidean space, it is only needed the division of the third component of the homogeneous vector. Another useful property of homogeneous coordinates is that if the third component is zero $[x, y, 0]^T$, dividing by the last coordinate the corresponding Euclidean point has infinite value. This shows how finite points in the Projective space can represent points in the Euclidean space *at infinity*.

1.4 Quadrotor and ground vehicle models

In this section the models for the dynamics of the two vehicles under study are presented.

1.4.1 Kinematic and dynamic model of a quadrotor

A quadrotor UAV can be represented as a rigid body with constant mass m and inertia matrix J (expressed in body frame) affected by gravitational and propulsive forces, moving in 3D space. Its movement is achieved by the combination of angular velocities ω_i of four propellers. This results in four different thrusts that produce a main force F applied in the centre of mass and angular moments along the three body axis $\boldsymbol{\tau} = [\tau_\phi, \tau_\theta, \tau_\psi]^T$. Position in inertial coordinates of the UAV is denoted $\boldsymbol{\xi} = [x, y, z]^T$, its Euler angles as $\boldsymbol{\eta} = [\phi, \theta, \psi]^T$, linear and angular velocities in body frame $\mathbf{V} = [u, v, w]^T$ and $\boldsymbol{\Omega} = [p, q, r]^T$ respectively.

As seen in Sections 1.2.1 and 1.2.2, linear and angular velocities are related by

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= R_b^i \mathbf{V}, \\ \dot{\boldsymbol{\eta}} &= W_n^{-1} \boldsymbol{\Omega}. \end{aligned}$$

Newton's second law for linear and angular motion in body frame states

$$m\ddot{\boldsymbol{\xi}} = R_b^i \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}, \tag{1.7}$$

$$J\dot{\boldsymbol{\Omega}} = -\boldsymbol{\Omega} \times J\boldsymbol{\Omega} + \boldsymbol{\tau}, \tag{1.8}$$

where symbol $(\times)$ is used for the cross product and g is the gravitational constant.

Every propeller located as in Figure 1.3 provides a thrust force f_i for $i = 1, ..., 4$ in body frame with direction along *z-axis* (note that to have a thrust pointing upwards, the values of f_i should be negative). Each of these forces depends on the angular velocity of the propellers ω_i by

$$f_i = c_T \omega_i^2,$$

where c_T is a constant lumped thrust parameter that takes into account the properties of the propeller (e.g. disk area, radius, thrust coefficient). A reaction torque along z due to propeller drag acting on the airframe may be modeled as

$$q_i = c_Q \omega_i^2,$$

where coefficient c_Q is a constant torque drag parameter which also depends on similar properties as c_T.

The contribution of each propeller to the main thrust is given by

$$F = f_1 + f_2 + f_3 + f_4.$$

The distance between rotor 1-2 and 3-4 along y and distance between rotor 2-3 and 1-4 along x, is equal to d. Therefore, the contribution on torques τ_ϕ, τ_θ and τ_ψ is

$$\begin{aligned}\tau_\phi &= \frac{d}{2}(f_1 - f_2 - f_3 + f_4),\\ \tau_\theta &= \frac{d}{2}(-f_1 - f_2 + f_3 + f_4),\\ \tau_\psi &= \frac{c_Q}{c_T}(f_1 - f_2 + f_3 - f_4).\end{aligned}$$

Summarizing, the relation of individual rotor angular speeds and the system inputs of (1.7)-(1.8) can be written in matrix form as

$$\begin{bmatrix} F \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ \frac{d}{2}c_T & -\frac{d}{2}c_T & -\frac{d}{2}c_T & \frac{d}{2}c_T \\ -\frac{d}{2}c_T & -\frac{d}{2}c_T & \frac{d}{2}c_T & \frac{d}{2}c_T \\ c_Q & -c_Q & c_Q & -c_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}. \tag{1.9}$$

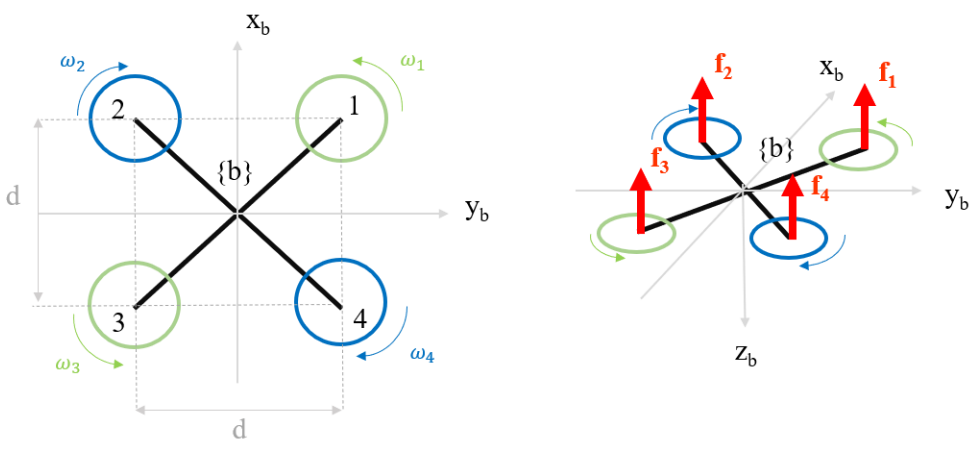

Figure 1.3: Drone's scheme, note that the values of thrust f_i are negative in the illustration

1.4.2 Ground vehicle model

An Ackermann steering vehicle model was chosen to simulate the kinematics of the ground vehicle on top of which the UAV should land. It consists on a rigid body moving on the horizontal North-East plane, composed by 4 wheels, two steerable with angle θ_{st} on front and two non-steerable on the rear. A scheme of the vehicle can be found at Figure 1.4. Denoting $\mathbf{v}_F$ and $\mathbf{v}_R$ the velocity of the point in the middle of the front and rear axle respectively, two non-holonomic constraint are imposed,

$$\mathbf{v}_R^b = \begin{bmatrix} u_R \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{v}_F^b = \begin{bmatrix} V_F \cos\theta_{st} \\ V_F \sin\theta_{st} \\ 0 \end{bmatrix}, \tag{1.10}$$

where velocities are expressed in body frame and V_F is the module of velocity of the front wheels. From the rigid body constraint, it follows that

$$\begin{aligned} u_R &= V_F \cos\theta_{st}, \\ \dot{\psi} &= \frac{V_F}{L}\sin\theta_{st}, \end{aligned} \tag{1.11}$$

where ψ is the yaw angle of the GV and L is the distance between front and rear wheels axles. The velocity in body frame of the center of gravity, located in the point on the middle between front and rear axles is

$$\mathbf{v}_{CG}^b = \begin{bmatrix} V_F \cos\theta_{st} \\ \frac{V_F}{2}\sin\theta_{st} \\ 0 \end{bmatrix}. \tag{1.12}$$

The kinematics equations for position $\xi_{CG} = [x_{CG}, y_{CG}, z_{CG}]^T$ of the center of gravity in inertial frame is

$$\begin{aligned} \dot{\xi}_{CG} &= R_b^i \mathbf{v}_{CG}^b, \\ \dot{\psi} &= \frac{V_F}{L}\sin\theta_{st}, \end{aligned} \tag{1.13}$$

where R_b^i is the rotation matrix

$$R_b^i = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

To notice that the inputs of the model are front wheel's speed V_F and steering angle θ_{st}.

1.5 Control system architecture

The objective of this thesis is to provide a simulation of the landing of an UAV on top of a ground vehicle, reproducing in a similar way the experiment of Borowczyk et al. [4]. In order to perform this, a control system architecture is proposed and it is composed of:

- a tracking control module,

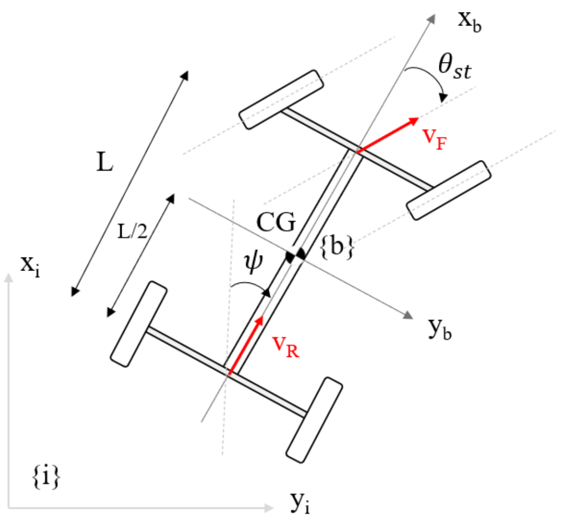

Figure 1.4: Ackermann vehicle scheme

- a trajectory generation module,
- a navigation system module,

each of these components is reproduced in Figure 1.5 and described in the this section.

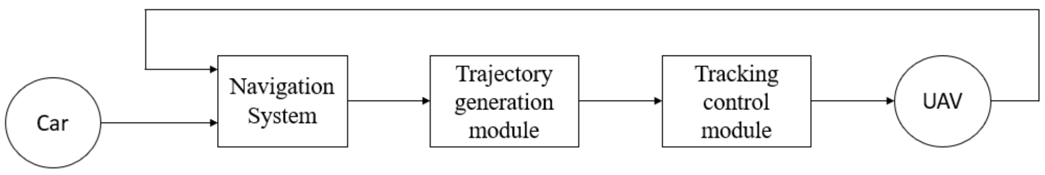

Figure 1.5: Control system architecture

1.5.1 Tracking control module

The tracking control module is a built-in controller running on the Flight Control Unit (FCU) of the drone performing the landing, which can be found schematized in Figure 1.6. Its purpose is to generate the values of desired angular speed of the four rotors (ω_i) from the input of setpoint position ($\boldsymbol{\xi}^o$) and yaw angle (ψ^0). This is achieved by a sequence of three blocks in order: a position controller, an attitude controller and a mixer.

The position controller is the outer loop with feedback in position ($\hat{\boldsymbol{\xi}}$) and linear velocity ($\hat{\mathbf{V}}$) that returns set-point attitude ($\boldsymbol{\eta}^0$) and thrust (F_c) from desired position and yaw input. Afterwards, the attitude controller, an inner loop on attitude ($\hat{\boldsymbol{\eta}}$) and angular velocity ($\hat{\boldsymbol{\Omega}}$), generates the required set-point moments ($\boldsymbol{\tau}_c$) to achieve the tracking. Finally, the mixer returns angular speed ω_i for the electric motors of the four rotors.

The design of a tracking control module is beyond the scope of this thesis, so the simulation uses a discrete time linear model of the dynamics of an UAV equipped with a tracking control module based on the thesis work of Gozzini G. [6]. It consists in a model identified by a black-box method based on a Predictor Based Subspace Identification PBSID algorithm using closed-loop experimental data (more details can be found in [6]). This model provides the decoupled dynamics on the North, East and Down directions.

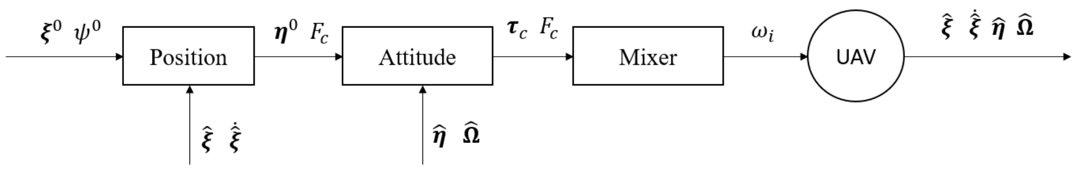
$\boldsymbol{\xi}^0$ ψ^0
Position
$\boldsymbol{\eta}^0$ F_c
Attitude
$\boldsymbol{\tau}_c$ F_c
Mixer
ω_i
UAV
$\hat{\boldsymbol{\xi}}$ $\hat{\dot{\boldsymbol{\xi}}}$ $\hat{\boldsymbol{\eta}}$ $\hat{\boldsymbol{\Omega}}$
$\hat{\boldsymbol{\xi}}$ $\hat{\dot{\boldsymbol{\xi}}}$
$\hat{\boldsymbol{\eta}}$ $\hat{\boldsymbol{\Omega}}$

Figure 1.6: Tracking control module scheme

1.5.2 Trajectory generation module

The trajectory generation module is a Proportional Derivative (PD) controller located before the tracking module. It generates set-point positions for the tracking module from relative position and velocity of UAV and GV estimated by the navigation system.

The trajectory is generated in both horizontal and vertical direction. On the horizontal plane, a North-East set-point acceleration is computed by means of strategy described in Section 4.1.1 and integrated twice to get the horizontal trajectory position. On vertical direction, a third order polynomial trajectory is imposed and vertical set-point acceleration is calculated accordingly as presented in Section 4.1.2. This last is integrated twice to obtain the desired vertical position as well.

Details on the trajectory generation module can all be found in Chapter 4.

1.5.3 Navigation system

The navigation system is responsible for the estimation of relative position and velocity between UAV and GV from measurements coming from different sensors and needed for the trajectory generation module. The sensors used are: UAV and GV accelerometers, camera imaging sensor mounted on the UAV, a radio-frequency range measurements module and a compass on the GV, all described in Section 3.2.

An Extended Kalman filter (EKF) algorithm was chosen as strategy for the *sensor fusion* of all the measurements coming at different frequencies. One of the advantages of the EKF choice is that it allows to estimate also the sensors' error states, resulting in a more accurate estimation of relative position and velocity.

All the details on the navigation system are found in Chapter 3.

1.6 Notation

It is going to be presented the notation that will be used from now until the rest of the work.

We define position, velocity and acceleration of the UAV with respect to the NED inertial frame

$$\mathbf{p}_d = \begin{bmatrix} N_d \\ E_d \\ D_d \end{bmatrix}, \qquad \mathbf{v}_d = \begin{bmatrix} \dot{N}_d \\ \dot{E}_d \\ \dot{D}_d \end{bmatrix}, \qquad \mathbf{a}_d = \begin{bmatrix} \ddot{N}_d \\ \ddot{E}_d \\ \ddot{D}_d \end{bmatrix}. \tag{1.14}$$

with subscript $(.)_d$ that refers to "drone", and position of the GV with respect to NED frame

$$\mathbf{p}_t = \begin{bmatrix} N_t \\ E_t \\ D_t \end{bmatrix}, \qquad \mathbf{v}_t = \begin{bmatrix} \dot{N}_t \\ \dot{E}_t \\ \dot{D}_t \end{bmatrix}, \qquad \mathbf{a}_t = \begin{bmatrix} \ddot{N}_t \\ \ddot{E}_t \\ \ddot{D}_t \end{bmatrix}, \tag{1.15}$$

with subscript $(.)_t$ that refers to "target". The relative position of GV with respect to the UAV is

$$\begin{aligned} \mathbf{p}_r &= \mathbf{p}_t - \mathbf{p}_d, \\ \mathbf{v}_r &= \mathbf{v}_t - \mathbf{v}_d, \\ \mathbf{a}_r &= \mathbf{a}_t - \mathbf{a}_d, \end{aligned} \tag{1.16}$$

with subscript $(.)_r$ that refers to "relative"

Chapter 2

Computer vision

In this chapter, a model used to simulate the frames of the camera mounted on the UAV is presented, as well as a computer vision method to reconstruct UAV's position and attitude from a tag detected in the image.

2.1 Overview

As already mentioned in Section 1.5.3, the integrated navigation system of the UAV combines information coming from multiple sensors, one of which is a camera installed on the UAV.

The purpose of the camera is to observe the landing scene and reproduce images that will be processed by a real-time computer vision algorithm, responsible to identify the landing point and derive information on the position of the UAV relative to the landing target, later used from the navigation system.

The overall strategy for the image simulation and position estimation can be summarized in the following points:

1. **Image acquisition**: the process of 2D projection of the observed 3D scene. It will be shown how frames are created under the assumption of a *pinhole* camera model.

2. **Projective transformation**: reconstruction of the transformation matrix that describes the projective process of the camera.

3. **Position estimation**: the projective transformation matrix is used to obtain information on the position of the camera.

Each point is schematized at Figure 2.1 and will be discussed in detail in the following sections.

During the work, a series of assumptions are made. First, the camera is assumed to be gimballed, which means it has three-axis rotational degree of freedom in order to be able to point any direction of space, the strategy for the control of the camera angles will be discussed in Section 4.3. The image is assumed to be not distorted by lens effects and last, the GV is assumed to be equipped on top by an image of a tag described in the next section. The theory presented in this chapter follows the Hartley and Zisserman *Multiple View Geometry in computer vision* [2] textbook.

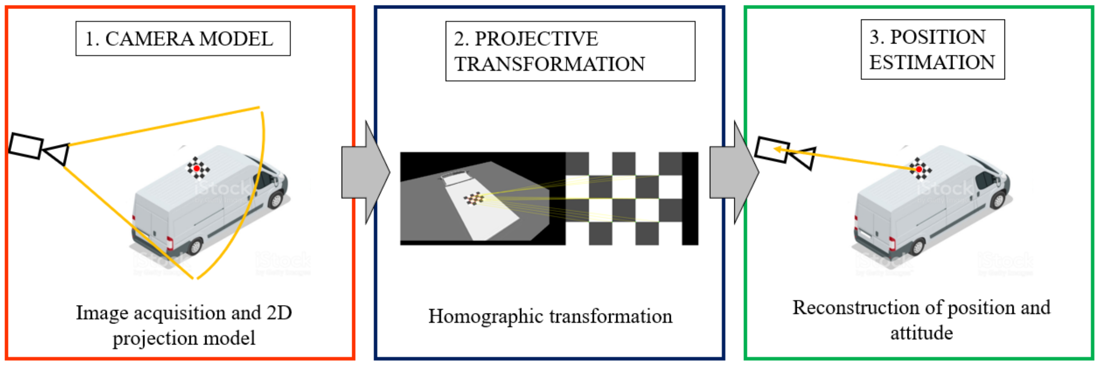
1. CAMERA MODEL
Image acquisition and 2D projection model
2. PROJECTIVE TRANSFORMATION
Homographic transformation
3. POSITION ESTIMATION
Reconstruction of position and attitude

Figure 2.1: Computer vision strategy

2.1.1 Fiducial tag

Fiducial tags are artificial landmarks designed to be easily recognised in an image from a computer vision algorithm. They are specifically created to be automatically detected and localized even in low resolution or cluttered images and sometimes in case of partial object obstruction of the tag. An important aspect of fiducial tags is that they can provide relative position and orientation of camera with respect to the tag. They were first developed for applications for augmented reality, where several popular systems include ARToolkit [11] and ARTag [12]. Later they have been widely adopted also by the robotics community where a notable system is AprilTag [5], developed by the University of Michigan.

For the purpose of this thesis a basic fiducial tag constituted by a flat (4x5) black-and-white checkerboard was chosen. The reason for this choice is that an already implemented library for the detection of a checkerboard can be found in the software used for the simulation (MATLAB), whereas the examples cited above are written in C++.

It is then supposed that a checkerboard of dimensions $L_x = 50\,cm$ wide and $L_y = 40\,cm$ high (depicted at Figure 2.2), is printed on top of the GV and its center is located in the desired landing point.

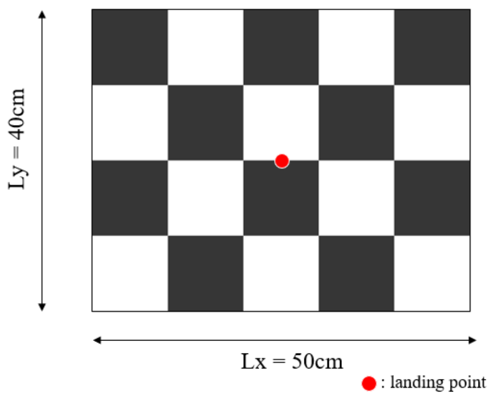

Figure 2.2: Checkerboard used for landing

2.2 Image acquisition

A camera sensor is a device capable to transform a 3D scene to a 2D projection by capturing the field of light emanating from the scene itself. In this section the geometric principles describing the process of creation of an image and used to simulate the camera frames are explained.

2.2.1 Camera central projection model

One of the most common camera models used in computer vision is the *central projection* model. Recalling the camera reference frame described in Section 1.1.4, the *z-axis* represents the direction of sight of the camera and is usually called *optical axis*. At focal distance $z = f$ is located the so called *image plane* and origin C is the *camera center*. The process of creation of an image can be described as the intersection between the image plane and the lines connecting points on the scene to the camera center. For this reason, this model is also often called *pinhole model*. An illustration is shown at Figure 2.3.

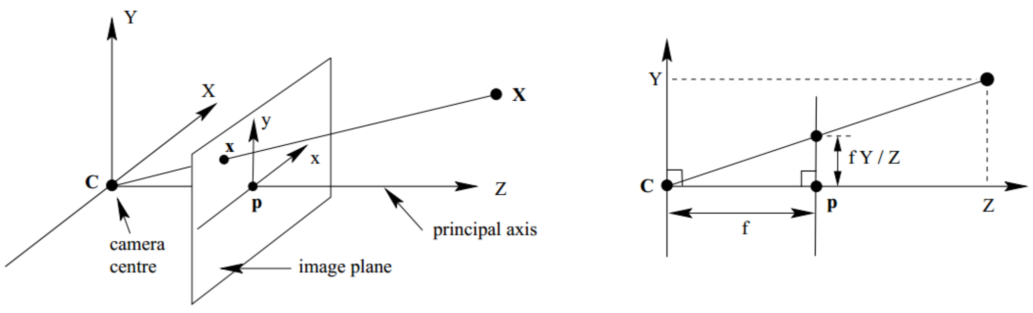

Figure 2.3: Central Projection model (from [2])

The relation between 3D-scene world coordinates (X, Y, Z) in the camera reference frame and 2D-image plane coordinates (x, y) can be expressed by considering similar triangles as seen from Figure 2.3, thus

$$\begin{aligned} x &= f\frac{X}{Z}, \\ y &= f\frac{Y}{Z}. \end{aligned} \tag{2.1}$$

Formula (2.1) describes the central projection model equations. Writing image coordinates $\tilde{p} = [x, y, 1]^T$ and world coordinates $\bar{P}^c = [X, Y, Z, 1]^T$ by means of homogeneous and augmented vectors respectively, the central projection is simply a linear expression of the type,

$$\tilde{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \\ 1 \end{bmatrix} \sim \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\tilde{\Pi}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \tilde{\Pi}\bar{P}^c, \tag{2.2}$$

recalling that homogeneous vectors are equivalent (sign $\sim$ as explained in Section 1.3.2) up to a scale factor (in this case Z). What is convenient and useful about this representation is that the central projection equations from a non-linear expression of equation (2.1) are now expressed by a linear matrix multiplication. There is no explicit division by Z because it is implicit in writing the equations in homogeneous coordinates.

Matrix $\tilde{\Pi}$ can be further factored into

$$\tilde{\Pi} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.3}$$

So far, the image plane was assumed to be continuous, however in reality the image plane is quantized. It consists of arrays and columns of light sensing elements which correspond to the pixels in the output image. The dimension of each pixel in this grid is denoted by ρ_u along x and ρ_v along y. Vector $\tilde{p} = [x, y, 1]^T$ in units of meters computed previously needs to be converted into $\bar{p} = [u, v, 1]^T$ in units of pixels. Pixel coordinates have a different origin and are measured from the corner of the image, hence a scaling and a shifting is done by the operation

$$u = \frac{x}{\rho_u} + u_0, \qquad v = \frac{y}{\rho_v} + v_0.$$

Which can be expressed in matrix form and homogeneous coordinates as

$$\bar{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\rho_u} & 0 & u_0 \\ 0 & \frac{1}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.4}$$

The elements of the matrix in (2.4) are the dimensions of the pixel (ρ_u, ρ_v) and the coordinates of the so called *principal point* (u_0, v_0), defined as the pixel coordinate of the intersection of the principal axis and image plane. It is worth noting that coordinates (u, v) are defined from zero to the pixel resolution value along (x, y).

From equations (2.2), (2.4) and using the factorization in (2.3) we get the expression of pixel coordinates as a function of 3D-world points,

$$\bar{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \underbrace{\begin{bmatrix} \frac{1}{\rho_u} & 0 & u_0 \\ 0 & \frac{1}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\Pi_0} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K\Pi_0\bar{P}^c. \tag{2.5}$$

As a result, the complete camera model is described by three matrices. The product of the first two matrices is typically denoted as the *camera calibration matrix* K and is a function of the intrinsic parameters, the quantities strictly related to the sensor's parameters like focal length and pixel dimensions. The (3x4) matrix Π_0 on the right is responsible for the dimensional reduction of a point from 3D to 2D, for this reason is often called the *standard projection matrix*.

2.2.2 Camera rotation and translation

It would be preferable to express coordinates of point $\bar{P}^c$ not in terms of the camera reference frame $\{c\}$, but in GV's coordinate frame $\{t\}$. The two frames are related by a rotation R_t^c and a translation $(T - C)$ as shown in Figure 2.4.

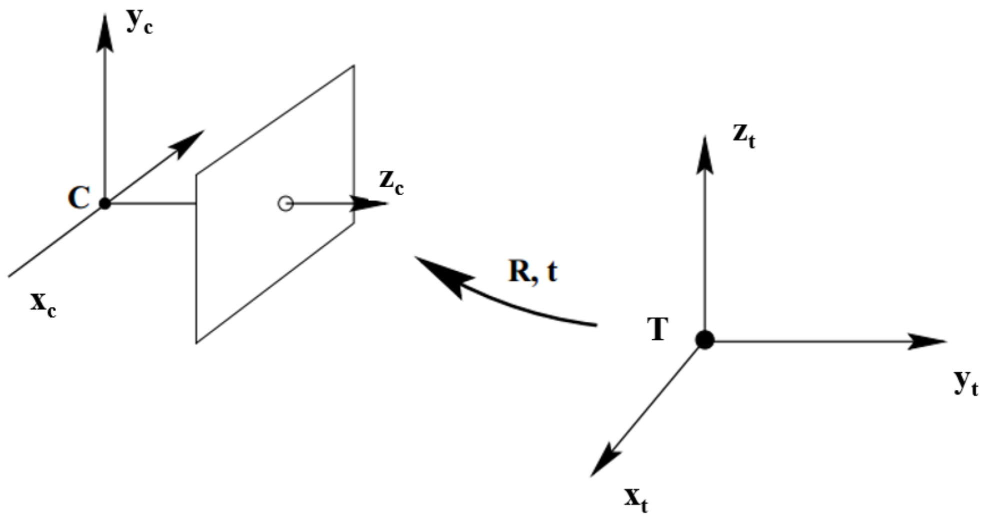

Figure 2.4: Reference frame transformation

The change of coordinates is performed by the augmented representation matrix T_t^c,

$$\bar{P}^c = \begin{bmatrix} R_t^c & T^c \\ 0_{1\text{x}3} & 1 \end{bmatrix} \bar{P}^t = T_t^c \bar{P}^t, \tag{2.6}$$

where $T^c = (T - C)^c$ represents the coordinate of GV's origin in the camera reference frame. Substituting equation (2.6) in (2.5), we get

$$\bar{p} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim K\Pi_0 T_t^c \bar{P}^t = \Pi_T \bar{P}^t. \tag{2.7}$$

The parameters of T_t^c depend on the camera orientation and position with respect to the GV, for this reason they are called *extrinsic parameters.* Matrix $\Pi_T = K\Pi_0 T_t^c$ is the *projection matrix* that includes both intrinsic and extrinsic parameters. Writing the three rows of Π_T as π_1^T, π_2^T, π_3^T, the expression in pixels of a generic point in GV's coordinates is

$$\begin{aligned} u &= \frac{\pi_1^T \bar{P}^t}{\pi_3^T \bar{P}^t}, \\ v &= \frac{\pi_2^T \bar{P}^t}{\pi_3^T \bar{P}^t}. \end{aligned} \tag{2.8}$$

2.3 Projective transformation

One of the effects of the image projection process described above, is that a regular and flat element can be seen from the camera as distorted and scaled, depending on the orientation of the camera with respect to this object. Figure 2.5 on the left shows a simulated frame during a generic moment of the landing approach. It can be noticed that in this image, the shape of the checkerboard on top of the vehicle is not anymore a perfect rectangle like its orthogonal representation at Figure 2.5 on the right. However, it will be demonstrated how the distortion that occurred between those two images will be the source of information for the estimation of position and attitude of the camera. In this section, a method to reconstruct the transformation of the tag that occurred between the two generic images at Figure 2.5 is presented.

2.3.1 Checkerboard detection

First, an algorithm to detect the checkerboard in the image should be addressed. The MATLAB function `detectCheckerboardPoints` in the Computer Vision Toolbox [13] was used for this purpose: the function takes as input a truecolor or grayscale image and returns the coordinates of the intersection points between columns and rows of a black and white checkerboard in the scene. These points

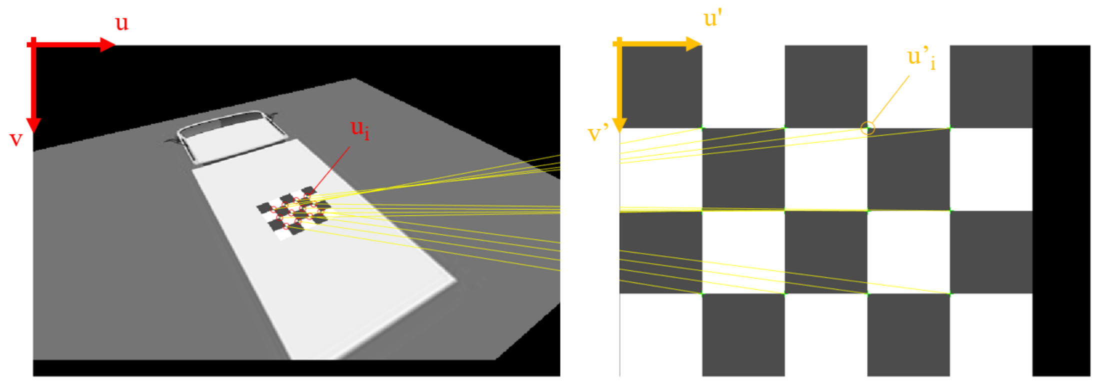

Figure 2.5: Checkerboard on a generic landing scene (on the left) and checkerboard tag image (on the right)

are returned in the ordered pattern shown in Figure 2.6. A test for the correct detection of the tag was set by checking that the number of output points was precisely $N = 12$, otherwise the frame was not considered.

This algorithm is based on the work of A. Geiger [14], developed originally with the aim of providing a way to use a checkerboard to calibrate a camera (i.e., finding its intrinsic parameters) with a single shot frame. It is a robust system, however it may have slower processing times (around 100 ms for a 640x480 image) with respect to other systems optimized for real time applications, like AprilTags (mean processing time of 22 ms for an image with the same resolution) and thus may result of better performance for an experimental application.

However, the checkerboard detection time was considered sufficient and adopted as fiducial tag. Future developments could be the implementation of alternative techniques like Apriltags, ARTags or ARToolkit fiducials.

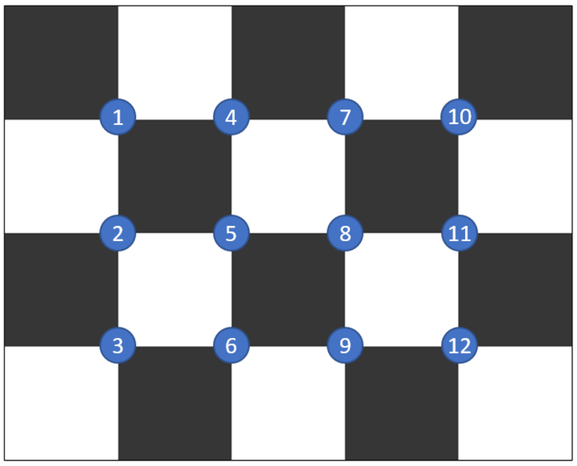

Figure 2.6: Order of detected checkerboard tag points

2.3.2 Homography transformation

The `detectCheckerboardPoints` function applied to an orthogonal reference image like in Figure 2.6 returns 12 *tag points* in homogeneous image coordinates $\mathbf{u}'_i = [u'_i, v'_i, 1]^T$, whereas applied to the landing scene obseved by the camera returns points of different coordinates $\mathbf{u}_i = [u_i, v_i, 1]^T$, where $i = 1, ...N$. The same convention of image coordinates as in MATLAB is used, with origin on top-left corner, *x-axis* pointing right, *y-axis* pointing down.

An *homography matrix* H, describing the transformation that occured between $\mathbf{u}_i$ and $\mathbf{u}'_i$ can be written as

$$H\mathbf{u}_i = \mathbf{u'}_i,$$

which sets the univocal transformation that occured in the checkerboard image between two frames like in Figure 2.5. The objective then, is to find an expression of H from the correspondece of points $\mathbf{u}_i$ and $\mathbf{u}'_i$.

A first consideration should be on the minimum number of points needed to solve the problem: matrix H is a (3x3) composed by 9 elements, but homogeneous coordinates are defined up to scale, so there are 8 unknowns. Each point $\mathbf{u}_i$ is defined by two coordinates (u_i, v_i), so the minimum number of points needed is 4.

2.3.3 Direct linear transformation

A simple linear algorithm that allows to find matrix H is the Direct Linear Transformation (DLT) algorithm. The transformation $H\mathbf{u}_i = \mathbf{u'}_i$ can be rewritten as a cross product $\mathbf{u'}_i \times H\mathbf{u}_i = 0$. If $\mathbf{h}_j^T$ is the j-th row of matrix H, we get

$$H\mathbf{u}_i = \begin{bmatrix} \mathbf{h}_1^T \\ \mathbf{h}_2^T \\ \mathbf{h}_3^T \end{bmatrix} \mathbf{u}_i = \begin{bmatrix} \mathbf{h}_1^T\mathbf{u}_i \\ \mathbf{h}_2^T\mathbf{u}_i \\ \mathbf{h}_3^T\mathbf{u}_i \end{bmatrix}.$$

Hence, the cross product becomes,

$$\mathbf{u'}_i \times H\mathbf{u}_i = \begin{bmatrix} v'_i\mathbf{h}_3^T\mathbf{u}_i - \mathbf{h}_2^T\mathbf{u}_i \\ \mathbf{h}_1^T\mathbf{u}_i - u_i\mathbf{h}_3^T\mathbf{u}_i \\ u'_i\mathbf{h}_2^T\mathbf{u}_i - v_i\mathbf{h}_1^T\mathbf{u}_i \end{bmatrix} = 0. \tag{2.9}$$

Collecting the elements of the unknown rows $\mathbf{h}_j^T$ into a column vector of dimension (9x1),

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix}.$$

Writing equation (2.9) as a linear expression with respect to $\mathbf{h}$,

$$\begin{bmatrix} \mathbf{0}^\top & -\mathbf{u}_i^\top & v'_i\mathbf{u}_i^\top \\ \mathbf{u}_i^\top & \mathbf{0}^\top & -u'_i\mathbf{u}_i^\top \\ -v'_i\mathbf{u}_i^\top & u'_i\mathbf{u}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = \mathbf{0}.$$

The last row of the matrix on the left is a linear combination of the upper two, so it is discarded. Thus, the set of equation becomes

$$\begin{bmatrix} \mathbf{0}^\top & -\mathbf{u}_i^\top & v_i'\mathbf{u}_i^\top \\ \mathbf{u}_i^\top & \mathbf{0}^\top & -u_i'\mathbf{u}_i^\top \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = A_i\mathbf{h} = \mathbf{0}. \tag{2.10}$$

Matrix A_i is function of the points $\mathbf{u'}_i$ and $\mathbf{u}_i$ and results to be a (2x9) matrix.

In the hypothetical case of having 4 different points' correspondences, we can build 4 independent A_i matrices, obtain the matrix A made by stacking the rows of each A_i and get the problem in the compact form of finding the non-trivial solution $\mathbf{h}$ of $A\mathbf{h} = 0$. Matrix A results to be a 8x9 matrix, so exists a null-space solution of $\mathbf{h}$ defined up to a non-zero scale factor. As discussed before, matrix H is defined up to scale, so it can be arbitrarily added a constraint on $\mathbf{h}$ such as $||\mathbf{h}|| = 1$.

However, since 12 points' correspondences were found, the equation $A\mathbf{h} = 0$ turns out to be overdetermined. In the ideal case where the correspondences are exact, the rank of matrix A should be still 8. However, since the correspondences are always disturbed by some noise and errors, the problem does not have an exact solution. A least-squares solution is then addressed by minimizing the norm $||A\mathbf{h}||$ and adding again the condition of $||\mathbf{h}|| = 1$. The solution is equal to the eigenvector of A^TA with least eigenvalue. In practice, a singular value decomposition (SVD) of the type $A = UDV^T$ is performed, where D is a diagonal matrix with non-negative elements, U and V are orthonormal matrices. The diagonal elements of D are sorted in descending order down the diagonal and $\mathbf{h}$ is the last column of V.

2.4 Extrinsic position estimation

In this last section is shown how from homography matrix H, position and attitude of the camera with respect to the GV can be reconstructed.

2.4.1 Relation between tag image and real checkerboard

An auxiliary *checkerboard frame* denoted $\{a\}$ with origin located at the center of the tag on top of the GV, xy plane on the tag plane and *z-axis* pointing upwards is introduced to facilitate the discussion. An illustration is found at Figure 2.7 on the right. Each tag point i lying on the xy plane of $\{a\}$ is identified by augmented coordinates $\bar{P}_i^a = [X_i^a, Y_i^a, 0, 1]^T$.

Points $\mathbf{u}_i'$ can be expressed from another image coordinate system centered at the middle of the tag image (u_0', v_0'), by coordinates $\mathbf{u}_i'' = [u_i'', v_i'', 1]^T$ described as

$$u_i'' = u_i' - u_0', \qquad v_i'' = -(v_i' - v_0'), \tag{2.11}$$

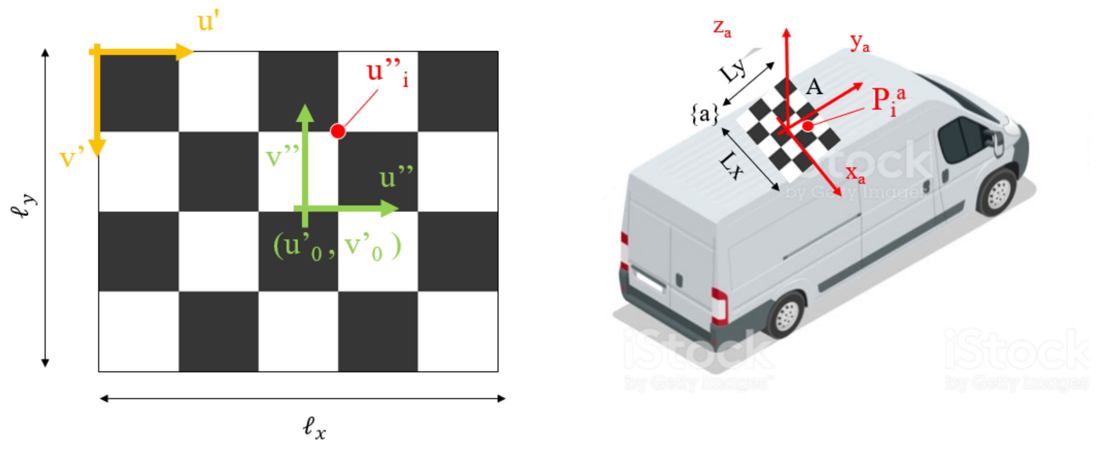

Figure 2.7: Aiding reference frame for image (on the left) and checkerboard reference frame (on the right)

and illustrated at Figure 2.7 on the left.

Comparing points $\mathbf{u}_i''$ on the image and $\bar{P}_i^a$, it can be noticed that they are related by a simple scaling along x and y direction by a factor $\lambda_x = \frac{L_x}{\ell_x}$ and $\lambda_y = \frac{L_y}{\ell_y}$, where $(L_x,\ L_y)$ are the real width and height in meters of the checkerboard and $(\ell_x,\ \ell_y)$ are the width and height in pixels of the checkerboard in the tag image. Hence, using relation (2.11) we get,

$$\begin{aligned} X_i^a &= \lambda_x u_i'' = \lambda_x(u_i' - u_0'), \\ Y_i^a &= \lambda_y v_i'' = -\lambda_y(v_i' - v_0'). \end{aligned}$$

The relation between real and image checkerboard points can also be written in matrix form,

$$\bar{P}_i^a = \begin{bmatrix} X_i^a \\ Y_i^a \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 & -\lambda_x u_0' \\ 0 & -\lambda_y & \lambda_y v_0' \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix}, \tag{2.12}$$

and by factorization of the matrix on the right

$$\bar{P}_i^a = \begin{bmatrix} X_i^a \\ Y_i^a \\ 0 \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \lambda_x & 0 & -\lambda_x u_0' \\ 0 & -\lambda_y & \lambda_y v_0' \\ 0 & 0 & 1 \end{bmatrix}}_{L} \begin{bmatrix} u_i' \\ v_i' \\ 1 \end{bmatrix} = QL\ \mathbf{u}_i'. \tag{2.13}$$

2.4.2 Position and attitude estimation

The checkerboard frame is rigidly attached to the GV, so the transformation between $\{t\}$ and $\{a\}$ is set by a constant augmented representation matrix T_a^t,

$$\bar{P}_i^t = T_a^t \bar{P}_i^a. \tag{2.14}$$

Introducing equation (2.13) in (2.7) and using (2.14), the correspondence between the detected points $\mathbf{u}_i$ on the landing scene image and the tag points on the orthogonal reference image $\mathbf{u}'_i$ is,

$$\mathbf{u}_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim K\Pi_0 T_t^c T_a^t Q L \mathbf{u}'_i. \tag{2.15}$$

Moreover, the homography matrix H obtained from the DLT algorithm is,

$$\mathbf{u}_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} u'_i \\ v'_i \\ 1 \end{bmatrix} = H\mathbf{u}'_i, \tag{2.16}$$

where each element of H is explicitly denoted as h_{ij}.

As (2.15) is an equivalent relation, it is defined up to a scalar factor equal to s, comparing (2.15) with (2.16) we get,

$$s(K\Pi_0 T_a^c Q L) = H.$$

The aim is to recover matrix T_a^c, hence,

$$s(\Pi_0 T_a^c Q) = K^{-1} H L^{-1}. \tag{2.17}$$

The term on the right $\tilde{H} = K^{-1}HL^{-1}$ is composed by the camera parameters matrix K, the homographic matrix H and matrix L which is function of the parameters of the real tag and the image tag.

The term on the left of (2.17) is,

$$\Pi_0 T_a^c Q = \begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & A_x^c \\ R_{21} & R_{22} & R_{23} & A_y^c \\ R_{31} & R_{32} & R_{33} & A_z^c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & A_x^c \\ R_{21} & R_{22} & A_y^c \\ R_{31} & R_{32} & A_z^c \end{bmatrix},$$

where R_{ij} are the elements of the rotation matrix R_a^c and $A^c = (A - C)^c$ is the center point of the checkerboard in the camera's reference frame.

Expressing $\tilde{h}_{ij}$ the elements of matrix $\tilde{H}$, we finally obtain,

$$\begin{bmatrix} R_{11} & R_{12} & A_x^c \\ R_{21} & R_{22} & A_y^c \\ R_{31} & R_{32} & A_z^c \end{bmatrix} = \frac{1}{s} \begin{bmatrix} \tilde{h}_{11} & \tilde{h}_{12} & \tilde{h}_{13} \\ \tilde{h}_{21} & \tilde{h}_{22} & \tilde{h}_{23} \\ \tilde{h}_{31} & \tilde{h}_{32} & \tilde{h}_{33} \end{bmatrix}. \tag{2.18}$$

Elements R_{ij} and A^c can be solved in the following way. Since the columns of a rotation matrix must all be unit magnitude, the value of s is found by computing the geometric average of the norm of the first two columns, explicitly

$$s = \pm\sqrt{(\tilde{h}_{11}^2 + \tilde{h}_{21}^2 + \tilde{h}_{31}^2)(\tilde{h}_{12}^2 + \tilde{h}_{22}^2 + \tilde{h}_{32}^2)}.$$

The sign of s is recovered by requiring that the tag appears in front of the camera, i.e., that $A_z^c > 0$. At this point, the first two columns of R_a^c can be calculated and the third column can be found by computing the cross product of the first two columns (as the rotation matrix should be orthonormal).

The position of the center of the tag is then $A^c = (\tilde{h}_{13}/s,\, \tilde{h}_{23}/s,\, \tilde{h}_{33}/s)$ in camera's coordinates, however the aim is expressing the position of the camera center, so

$$C^a = (C - A)^a = -(A - C)^a = -R_c^a\, A^c, \tag{2.19}$$

recalling that $\{a\}$ is rigidly attached to $\{t\}$,

$$\begin{bmatrix} C^t \\ 1 \end{bmatrix} = T_a^t \begin{bmatrix} C^a \\ 1. \end{bmatrix} \tag{2.20}$$

Thus, the output from the camera will be denoted by

$$\mathbf{p}_{cam} = C^t.$$

Finally the relative position of the camera with respect to GV in inertial coordinates is

$$(C - T)^i = R_t^i\, C^t = R_t^i\, \mathbf{p}_{cam}, \tag{2.21}$$

where R_t^i is calculated from the Euler Angles of the GV. To obtain the relative position $\mathbf{p}_r$ between UAV and GV defined in Section 1.6,

$$\begin{aligned} \mathbf{p}_r = \mathbf{p}_t - \mathbf{p}_d &= (T - D)^i = (T - C)^i + (C - D)^i \\ &= -R_t^i\, \mathbf{p}_{cam} + R_d^i\, (C - D)^d \end{aligned} \tag{2.22}$$

where R_d^i is the rotation matrix from inertial to UAV body frame and $(C - D)^d$ is a constant vector that describes the position of the camera rigidly attached to the UAV body frame.

Chapter 3

Integrated navigation system principles

This chapter describes the basic principles of the integrated navigation system responsible for the estimation of the relative position and velocity of the UAV with respect to the GV and required by the control system of the UAV for the landing guidance.

The main concepts behind integrated navigation, Kalman filter and Extended Kalman filter (EKF) are presented and their application for the landing problem is shown.

3.1 Types of navigation system

Navigation is the discipline that studies the determination of position and velocity of a moving body with respect to a known reference point [15]. Its applications are everywhere such as cars, airplanes, spacecrafts, ships and robotics. There are many navigation techniques, however two main cathegories can be identified. *Position fixing* uses identifiable external information to determine position directly. They can be man-made signals like Global Navigation Satellite Systems (GNSSs) or environmental features at known locations identified by a pilot or by a machine vision system. The main advantage is that the error in the determination of position does not depend on navigation time, but with the main drawback of depending on external factors and infrastructures.

The second system is *dead reckoning* that measures distance and direction traveled from an initial position by motion sensors like accelerometers. These sensors may be self-contained on-board the vehicle, requiring no external infrastructures and usually are a source of direct acceleration and velocity estimates at high update rates, information often necessary for the guidance and control modules. In contrast, acceleration and velocity obtained by differentiation of position fixing measures are less reliable because very sensitive to noise. The error on position estimated by dead reckoning, however, grows with time because the sequence of distance measurements errors accumulate.

An *integrated navigation system* determines position using more than one technology and can combine the advantages of both types of navigation methods. This is the approach followed in this work, where a dead reckoning system based on accelerometers mounted on the UAV and GV is integrated with position fixing measurements of a camera, a compass and a radio-frequency (RF) range system.

The sensor fusion is performed by an Extended Kalman filter described in Section 3.4. A series of considerations on sensors' models and errors is addressed in the next section.

3.2 Sensors

Sensors are devices of primary importance for navigation as they detect physical quantities and return variables on the state of motion or position of the vehicle. Ideal sensors would return the exact measure that we want to know, however in reality they are affected by a series of errors due to the technological realization of the instrument itself.

3.2.1 Sensor's errors

Navigation systems can reach high performances if sensors are modeled as precisely as possible, by estimating their deviation with respect to ideal sensors and improve the quality of the vehicle kinematic state estimate. Sensors' errors can be classified into two main categories. *Systematic errors* are deterministic and they are usually a bias or constant offset of the instrument that, for example, does not read zero when the quantity to be measured is zero, or a scale factor when the instrument constantly reads changes in quantity greater or lower than the actual changes. These errors are repeatable and can tried to be predicted; their representation is shown at Figure 3.1.

Random errors are caused by unknown and unpredictable changes on the instrument measure that may come, for example, from electronic noise in the circuit of the sensor. Due to their intrinsic stochastic nature, they cannot be predicted, however they may be described by means of their statistical properties.

A large class of random errors can be classified as white gaussian noise (WGN), defined as a process with constant value of power spectral density (PSD) in the frequency spectrum. In discrete time, a white noise sequence is a series of mutually uncorrelated random variables with zero-mean distribution. Taking samples w_i at time t_i, we have that

$$E(w_i w_j) = \begin{cases} \sigma_w^2 & \text{for } i = j, \\ 0 & \text{for } i \neq 0, \end{cases}$$

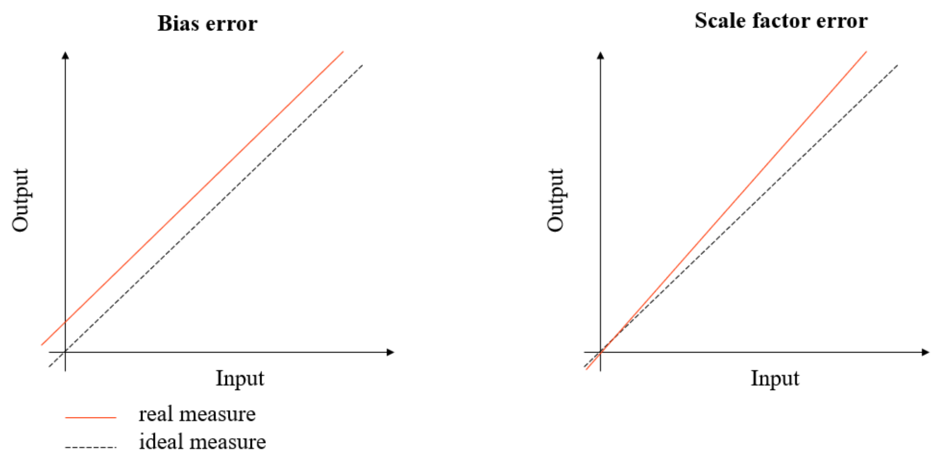

Figure 3.1: Systematic errors: bias and scale factor

where E is the expectation operator and σ_w^2 is the variance. All random errors assumed for the rest of the work are WGN: in general it is a strong assumption, since not all random errors fall necessarily into this category. However, this hypothesis is intrinsic in the Kalman filter formulation and it can be demonstrated that the filter keeps a good performance even if the system is affected by errors that are not exactly WGN, but similar.

3.2.2 Sensor models

The models for non-ideal sensors mounted on the GV and UAV are presented in this section.

Accelerometers and gyroscopes

Accelerometers and gyroscopes (or gyros) are electromechanical motion sensing devices used to measure linear acceleration forces and angular velocities respectively, used to sense movements of the vehicle on which they are mounted. Accelerations and angular rate measurements are provided in the three body axis of the vehicle in case of strap-down configurations. To retrieve these values in inertial frame, a change of coordinates should be performed, for example by means of Euler angles parameterization.

Many commercial UAVs include an inertial measurement unit (IMU) within the flight control unit which comprises already an assembly of gyroscopes and accelerometers. In this work, it is assumed that the measures of linear acceleration of the drone are already expressed in inertial coordinates and are gravity compensated. In fact, the FCU is able to transform the accelerations from body to inertial, using the estimation of roll, pitch and yaw angles obtained from gyroscopes and needed also by the control system. The model for the drone's linear acceleration measurement is

$$\tilde{\mathbf{a}}_d = (1 - s_d)\mathbf{a}_d + \mathbf{b}_d + \mathbf{w}_{a_d}, \tag{3.1}$$

where $\tilde{\mathbf{a}}_d$ is the measured acceleration in NED axis, s_d is a constant offset, $\mathbf{b}_d$ is a constant random bias of variance equal to $\sigma^2_{b_d}$ and $\mathbf{w}_{a_d}$ is a WGN of PSD equal to $\sigma^2_{a_d}$.

The movement of the GV is considered as two dimensional acting on the North-East plane. For this reason, a 2-sensitivity axis accelerometer is used for the determination of the accelerations $\boldsymbol{\mu}_t$ along x and y axis of GV's body frame. The measurement model is

$$\tilde{\boldsymbol{\mu}}_t = (1 - s_t)\boldsymbol{\mu}_t + \mathbf{b}_t + \mathbf{w}_{a_t}, \tag{3.2}$$

where $\tilde{\boldsymbol{\mu}}_t$ is the measured acceleration in GV's body axis, s_t is a constant offset, $\mathbf{b}_t$ is a constant bias of variance equal to $\sigma^2_{b_t}$ and $\mathbf{w}_{a_t}$ is a WGN of PSD equal to $\sigma^2_{a_t}$.

The 2D navigation assumption implies that the acceleration of gravity vector $\mathbf{g}$ does not affect GV's accelerometer's measures because always parallel to z. In practice, a small tilt of the GV in pitch and roll, introduces the projection of the gravity vector on the xy plane, which is measured by the accelerometers and interpreted as a fictitious movement, source of significant errors over time. For this reason, usuallly a 3-axis accelerometer is still used to capture this phenomenon, which for semplicity was assumed as not affecting the measures of the GV.

The GV is also equipped with a gyro with sensitive axis along z on GV's frame to measure angular velocity r_t and the sensor measurement is modeled as

$$\tilde{r}_t = (1 - s_r)r_t + b_r + w_r, \tag{3.3}$$

where $\tilde{r}_t$ is the measured yaw rate, s_r is a constant offset, b_r is a constant bias of variance equal to $\sigma^2_{b_r}$ and w_r a WGN with PSD equal to $\sigma^2_{r_t}$.

GV's compass

The estimation of the yaw angle ψ_t of the GV is made by integration of the gyro's yaw rate $\tilde{r}_t$ and an *attitude fixing* measurement coming from a compass installed on the GV and modeled as

$$\psi_{comp} = \psi_t + v_{comp}, \tag{3.4}$$

where ψ_{comp} is the yaw angle measured by the compass and v_{comp} is a WGN with PSD of σ^2_{comp}.

Camera sensor

The gimballed camera installed on the UAV, discussed in Chapter 2, is a position fixing aiding measurement which provides the measurement $\mathbf{p}_r$ of relative position of UAV and GV as shown in Section 2.4.2. The measure is affected by a series of errors in the computer vision algorithm and camera's parameters that are difficult to model, but that can be interpreted as WGN as well. From equation (2.22), the measurement of the camera is written as

$$\mathbf{p}_{cam} = -R_i^t(\psi_t)\,\mathbf{p}_r + \mathbf{v}_{cam}, \tag{3.5}$$

where, for sake of simplicity, points D and C are considered coincident. Rotation matrix R_i^t transforms from inertial to GV's reference frame, $\mathbf{p}_{cam}$ is the relative position computed by the computer vision algorithm and $\mathbf{v}_{cam}$ is a WGN of PSD σ^2_{cam}.

RF distance module

During the last moments of the landing approach, the detection of the checkerboard by the camera results to be compromised because the UAV flies too close to the tag, which goes outside the field of view of the camera. For this reason, an additional radio-frequency (RF) distance module is used to locate the UAV in the final steps of the landing, when camera information is not anymore available.

The UAV is equipped with a transceiver which emits RF signals that are received by other 4 transceivers located at the extremities of the tag (like in Figure 3.2) and that re-emit the same signal. By evaluating the time occurred between transmission and reception of the signal by the sensor on the UAV, and considering a constant speed of propagation of the RF wave, it is possible to reconstruct 4 range measures between the UAV and each of the 4 transceivers on the GV.

These 4 sensors are located at points $P^t_{RF_i} = (P_{RF_i} - T)^t$ for $i = 1, ..., 4$ with respect to GV's frame. The range between UAV and transceiver i is given by

$$\begin{aligned} \rho_{RF_i} &= ||(P_{RF_i} - D)|| = ||(P_{RF_i} - D)^i|| \\ &= ||(P_{RF_i} - T)^i + (T - D)^i|| = ||R^i_t P^t_{RF_i} + \mathbf{p}_r|| \end{aligned} \tag{3.6}$$

The actual measurements $\tilde{\rho}_{RF_i}$ from the RF distance module are affected by random errors v_{RF_i} described as WGN with PSD σ^2_{RF}, thus written as

$$\tilde{\rho}_{RF_i} = \rho_{RF_i} + v_{RF_i} \tag{3.7}$$

3.3 Discrete time Kalman filter

The Kalman filter is a sensor fusion algorithm that takes multiple measurements observed over time and produces estimates of unknown states of a system. It

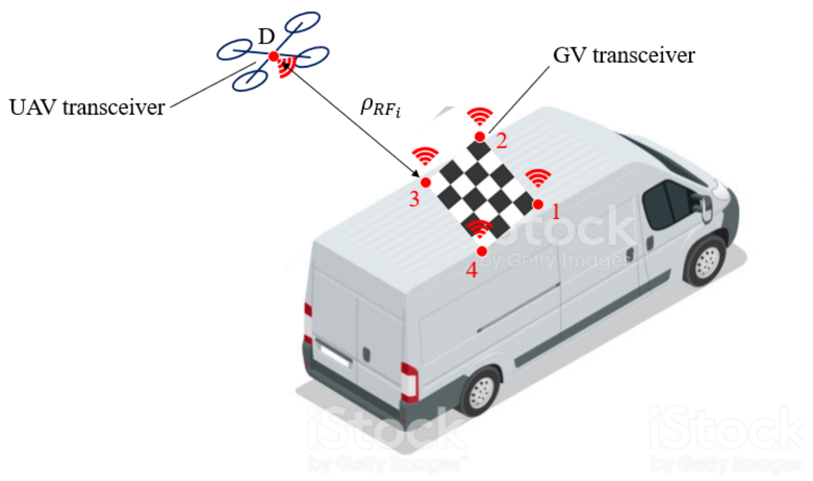

Figure 3.2: RF distance module architecture

uses the knowledge of deterministic and stochastic properties of the system and measurements in order to perform an optimal state estimation. The filter is named after Rudolf E. Kalman, which published this theory for the first time in 1960 [16]. Since then, the algorithm has seen its application in many fields, from the navigation computer for the Apollo program, to current smartphone technologies, but also in computer vision and finance.

In this section, the fundamental real-time and discrete form of the Kalman filter is explained.

3.3.1 Elements of the Kalman filter

The *state vector* $\mathbf{x}$ is a set of parameters describing the system (i.e., the *states*) which the Kalman filter aims to estimate. It can be constant or time varying and it usually includes position, velocity or attitude of the vehicle with respect to a particular frame of reference. It may also contain the states of the systematic errors introduced by sensors such as biases or scale factors. The Kalman filter *estimate* of the state vector is denoted $\hat{\mathbf{x}}$, whereas the *state vector residual* $\delta\mathbf{x} = \mathbf{x} - \hat{\mathbf{x}}$ is the difference between the state vector and its estimate.

The *error covariance matrix* Σ is the expectation of the square of the deviation of the state vector estimate from its true value, thus $\Sigma = E(\delta\mathbf{x}\delta\mathbf{x}^T)$. The elements on the diagonal Σ_{ii} are the variances of the corresponding states x_i, whereas the non-diagonal elements Σ_{ij} are the covariances between errors of x_i and x_j states.

The *system noise vector* $\mathbf{w}$ represents the uncertainties that may affect the system. It is defined by statistical properties of a zero-mean white gaussian noise, hence by a system noise covariance matrix $Q = E(\mathbf{w}\mathbf{w}^T)$. Matrix Q is diagonal if each noise element is uncorrelated from the others.

The *system model* describes how the Kalman filter states vary with time: it includes the kinematics of the vehicle under study, for example the velocity as derivative of position, or the dynamics of the sensors' errors.

The *measurement vector* $\mathbf{z}$ is a set of measures from position fixing aids (e.g., information coming from a camera, GPS or compass). The *measurement model* describes how $\mathbf{z}$ varies as a function of the true state $\mathbf{x}$ and affected by a measurement noise vector $\mathbf{v}$. The measurement noise covariance matrix R defines the expectation of the square of the measurement noise vector, hence $R = E(\mathbf{v}\mathbf{v}^T)$.

In the discrete time Kalman filter, only linear time invariant system and measurement models are considered, described by

$$\mathbf{x}_{k+1} = F_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k, \tag{3.8}$$

$$\mathbf{z}_k = H_k\mathbf{x}_k + \mathbf{v}_k, \tag{3.9}$$

where $\mathbf{u}$ is the input vector containing dead reckoning measurements and F_k, B_k, H_k are linear time invariant matrices. The subscript $(.)_k$ denotes each vector or matrix evaluated at time $t = t_k$.

3.3.2 Kalman filter algorithm

The discrete time Kalman filter algorithm is recursively performed by a navigation system at a rate which is usually equal to the frequency of the dead reckoning sensors. Each iteration is divided into two different phases:

1. **System propagation:** the prediction of the state vector and error covariance matrix from the last vector estimate and dead reckoning measurements. The state and covariance estimates performed during this stage are denoted with the superscript $(.)^-$.

2. **Measurement update:** the correction of the state estimate based on the position fixing measurements $\mathbf{z}_k$ if they are available at iteration time t_k. The state and covariance estimates in this phase are denoted with the superscript $(.)^+$.

Both phases are described by a series of equations in state space form that achieve an optimal estimation.

System propagation

The first step is the prediction of the state vector at time t_k based on previous estimates $\hat{\mathbf{x}}^+_{k-1}$, Σ^+_{k-1} and dead reckoning measurement $\mathbf{u}_{k-1}$. Using the discrete-time system model of (3.8) and considering a null noise $\mathbf{w}$, the state prediction is

$$\hat{\mathbf{x}}^-_k = F_{k-1}\hat{\mathbf{x}}^+_{k-1} + B_{k-1}\mathbf{u}_{k-1}. \tag{3.10}$$

It can be demonstrated that the error covariance matrix can be computed by

$$\Sigma_k^- = F_{k-1}\Sigma_{k-1}^+ F_{k-1}^T + Q_{k-1}, \tag{3.11}$$

where Q_{k-1} is the discrete-time system noise covariance matrix.

Measurement update

In general, the rate at which position fixing measures are available is lower than the filter's frequency. This leads to two different cases, if measure vector $\mathbf{z}_k$ is not available at instant t_k, the state estimate at iteration k is the one computed by the system propagation step, hence $\mathbf{x}_k^+ = \mathbf{x}_k^-$. In the case of $\mathbf{z}_k$ available, an update in $\mathbf{x}_k^-$ can be performed using the *measurement innovation* $\delta\mathbf{z}_k^- = \mathbf{z}_k - \hat{\mathbf{z}}_k^-$, where $\hat{\mathbf{z}}_k^- = H_k\hat{\mathbf{x}}_k^-$ is the predicted position fixing measurement using model (3.9). The new estimate is then

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\delta\mathbf{z}_k^-, \tag{3.12}$$

where K_k is the *Kalman gain* matrix, representing the weight of the measurement innovation in the state vector update. The criteria for computing K_k is by minimizing the error in the estimate $\hat{\mathbf{x}}_k^+$, resulting from the minimization of the trace of Σ_k^+. It can be demonstrated that this leads to the formula for the Kalman gain of the type

$$K_k = \Sigma_k^- H_k^T (H_k\Sigma_k^- H_k^T + R_k)^{-1}, \tag{3.13}$$

where R_k is the discrete-time measurement noise covariance matrix. The correction of the covariance matrix is

$$\Sigma_k^+ = (I - K_k H_k)\Sigma_k^-. \tag{3.14}$$

It can be noticed that in the expression of the Kalman gain of equation (3.13), there is an inverse dependency on the noise covariance matrix R_k, which is related to the uncertainty on the information of position fixing sensors at instant t_k. Considering R_k diagonal (each measure error is uncorrelated from the others), the less reliabile the position fixing measure z_i, the higher its variance (element $R_{k_{ii}}$ of the matrix), and the less its contribution on K_k. Hence, the weight of z_i on the state update $\hat{\mathbf{x}}_k^+$ results to be small due to the uncertainty of the the measure. The same reasoning can be inversely made for an high reliable position fixing measure.

3.4 Extended Kalman filter

The basic Kalman filter described in previous section has the limitation of being applied only to linear models. More complex systems, however, can present a nonlinearity associated either with the system model or with the measurement model or with both. The Extended Kalman Filter (EKF) is a version of the

Kalman filter which can be applied to continuous time nonlinear systems and measurement models of the type

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{w}(t)), \tag{3.15}$$
$$\mathbf{z}(t) = h(\mathbf{x}(t), \mathbf{v}(t)), \tag{3.16}$$

where f and g are two differentiable functions.

3.4.1 EKF algorithm

This section presents how a slight modification on the system propagation and measurement update steps of the Kalman filter leads to the algorithm of the EKF.

System propagation

Function f of (3.15) is used to compute the predicted state from previous estimate $\hat{\mathbf{x}}_{k-1}^{+}$ and dead reckoning measures $\mathbf{u}_{k-1}$ using a standard Euler method of the type

$$\hat{\mathbf{x}}_k^{-} = \hat{\mathbf{x}}_{k-1}^{+} + f(\hat{\mathbf{x}}_{k-1}^{+}, \mathbf{u}_{k-1}, 0)\Delta t, \tag{3.17}$$

where $\Delta t = t_k - t_{k-1}$ is the time interval of iteration k. Assuming that the error in the state vector estimate is small, a linearization can be applied to the state vector residual

$$\delta\dot{\mathbf{x}}(t) = A(t)\delta\mathbf{x}(t) + B(t)\mathbf{w}, \tag{3.18}$$

where

$$A(t) := \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{(\mathbf{x}(t),\mathbf{u}(t),0)}, \quad B(t) := \left.\frac{\partial f}{\partial \mathbf{w}}\right|_{(\mathbf{x}(t),\mathbf{u}(t),0)},$$

Calculating A_{k-1}, B_{k-1} and Q_{k-1} as $A(t)$, $B(t)$ and Q respectively, evaluated at $\hat{\mathbf{x}}_{k-1}^{+}$ and $\mathbf{u}_{k-1}$, we can write

$$\delta\mathbf{x}_k = \Phi_{k-1}\delta\mathbf{x}_{k-1} + \boldsymbol{\eta}_{k-1} \tag{3.19}$$

with $\Phi_{k-1} = e^{A_{k-1}\Delta t}$ and $\boldsymbol{\eta}_{k-1}$ is a zero-mean white gaussian noise of covariance

$$N_{k-1} = E(\boldsymbol{\eta}_{k-1}\boldsymbol{\eta}_{k-1}^T) = \int_0^{\Delta t} e^{A_{k-1}t}B_{k-1}Q_{k-1}B_{k-1}^T e^{A_{k-1}^T t}dt, \tag{3.20}$$

which can be approximated for small time steps as

$$N_{k-1} \approx \Delta t\, B_{k-1}Q_{k-1}B_{k-1}^T. \tag{3.21}$$

Hence, the predicted covariance matrix is found by

$$\Sigma_k^{-} = \Phi_{k-1}\Sigma_{k-1}^{+}\Phi_{k-1} + N_{k-1} \tag{3.22}$$

Measurement update

The measurement innovation $\delta\mathbf{z}_k^- = \mathbf{z}_k - \hat{\mathbf{z}}_k^-$, where $\hat{\mathbf{z}}_k^- = h(\hat{\mathbf{x}}_k^-, 0)$ is the predicted position fixing measurement, is assumed to be small. Thus, it can be approximated as a linear function of the state vector by

$$\delta\mathbf{z}_k^- \approx C_k\delta\mathbf{x}_k^- + D_k\mathbf{v}_k, \tag{3.23}$$

where C_k and D_k are obtained by linearization

$$C_k := \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{(\hat{\mathbf{x}}_k^-,0)}, \quad D_k := \left.\frac{\partial h}{\partial \mathbf{v}}\right|_{(\hat{\mathbf{x}}_k^-,0)},$$

The new Kalman filter gain for the EKF is

$$K_k = \Sigma_k^- C_k^T (C_k\Sigma_k^- C_k^T + D_k R_k D_k^T)^{-1},$$

and the correction on the state vector estimation results to be

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\delta\mathbf{z}_k^-, \tag{3.24}$$

whereas the covariance matrix update is

$$\Sigma_k^+ = (I - K_k C_k)\Sigma_k^-. \tag{3.25}$$

It was shown how, through linearization of functions f and g, the algorithm for the Kalman filter can be extended to nonlinear systems. A comparison between the two types of filters is shown at Figure 3.3.

3.5 Navigation system setup

The navigation system of the UAV uses an EKF algorithm for state estimation. Hence, two main elements that should be defined are the system model and the measurement model in the form of (3.15)-(3.16). Let's consider first what are the states that should be estimated. Since we are considering a landing problem, the interest is in the reconstruction of relative position and velocity of the UAV with respect to GV, rather than their absolute states. Moreover, having defined the models of the sensors in Section 3.2, it can be addressed also the estimation of the biases and scale factors errors.

In this section, the system and measurement models used by the EKF of the UAV is presented.

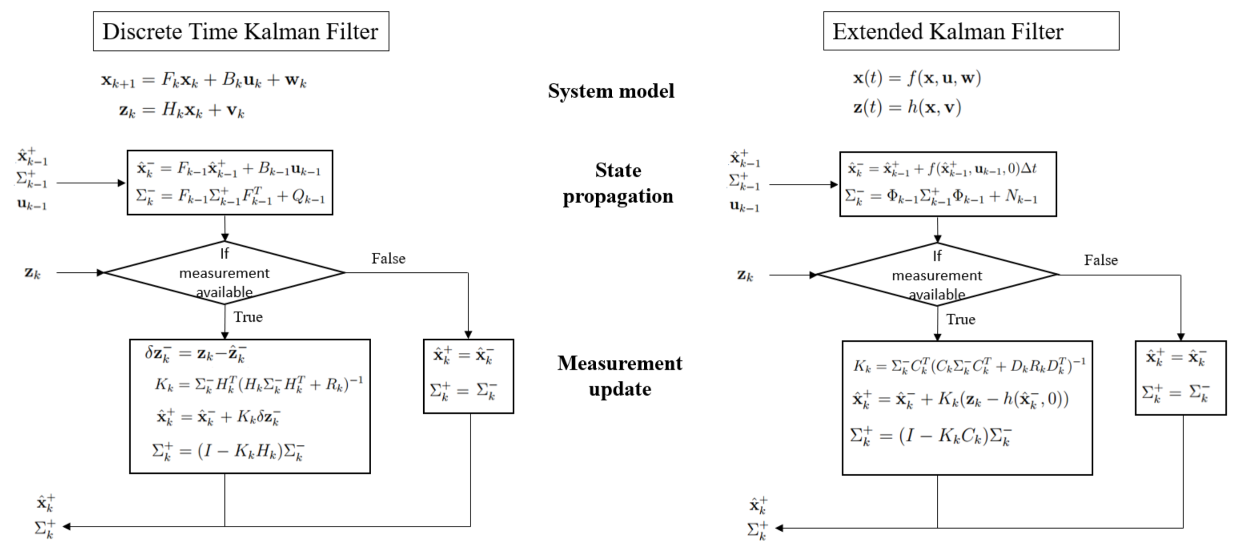
Discrete Time Kalman Filter
Extended Kalman Filter
System model
$\mathbf{x}_{k+1} = F_k\mathbf{x}_k + B_k\mathbf{u}_k + \mathbf{w}_k$
$\mathbf{z}_k = H_k\mathbf{x}_k + \mathbf{v}_k$
$\mathbf{x}(t) = f(\mathbf{x},\mathbf{u},\mathbf{w})$
$\mathbf{z}(t) = h(\mathbf{x},\mathbf{v})$
State propagation
$\hat{\mathbf{x}}_k^- = F_{k-1}\hat{\mathbf{x}}_{k-1}^+ + B_{k-1}\mathbf{u}_{k-1}$
$\Sigma_k^- = F_{k-1}\Sigma_{k-1}^+F_{k-1}^T + Q_{k-1}$
$\hat{\mathbf{x}}_k^- = \hat{\mathbf{x}}_{k-1}^+ + f(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}, 0)\Delta t$
$\Sigma_k^- = \Phi_{k-1}\Sigma_{k-1}^+\Phi_{k-1} + N_{k-1}$
If measurement available
True
False
Measurement update
$\delta\mathbf{z}_k^- = \mathbf{z}_k - \hat{\mathbf{z}}_k^-$
$K_k = \Sigma_k^- H_k^T(H_k\Sigma_k^- H_k^T + R_k)^{-1}$
$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k\delta\mathbf{z}_k^-$
$\Sigma_k^+ = (I - K_kH_k)\Sigma_k^-$
$K_k = \Sigma_k^- C_k^T(C_k\Sigma_k^- C_k^T + D_kR_kD_k^T)^{-1}$
$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-, 0))$
$\Sigma_k^+ = (I - K_kC_k)\Sigma_k^-$
$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^-$
$\Sigma_k^+ = \Sigma_k^-$
$\hat{\mathbf{x}}_{k-1}^+$, Σ_{k-1}^+, $\mathbf{u}_{k-1}$
$\mathbf{z}_k$
$\hat{\mathbf{x}}_k^+$, Σ_k^+

Figure 3.3: Comparison between discrete Kalman filter and Extended Kalman filter

3.5.1 System model

The kinematic equations for the relative position $\mathbf{p}_r$ and velocity $\mathbf{v}_r$ expressed in inertial coordinates are given by the differential equations

$$\begin{aligned} \dot{\mathbf{p}}_r &= \dot{\mathbf{p}}_t - \dot{\mathbf{p}}_d = \mathbf{v}_r, \\ \dot{\mathbf{v}}_r &= \dot{\mathbf{v}}_t - \dot{\mathbf{v}}_d = \mathbf{a}_t - \mathbf{a}_f. \end{aligned} \tag{3.26}$$

Acceleration of UAV and GV are obtained by the accelerometers described in Section 3.2. From UAV's accelerometer model of equation (3.1), the true acceleration expressed in inertial coordinates can be written as

$$\mathbf{a}_d = \frac{\tilde{\mathbf{a}}_d - \mathbf{b}_d - \mathbf{w}_{a_d}}{1 - s_d}.$$

The GV's accelerometer measures its values in body frame, thus a change of coordinates by means of rotation matrix R_t^i should be performed

$$\mathbf{a}_t = R_t^i(\psi_t)\boldsymbol{\mu}_t, \tag{3.27}$$

where rotation matrix is

$$R_t^i = \begin{bmatrix} \cos\psi_t & -\sin\psi_t & 0 \\ \sin\psi_t & \cos\psi_t & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Using (3.27) and sensor model of equation (3.2) , the acceleration of the GV in inertial frame is written as

$$\mathbf{a}_t = R_t^i(\psi_t)\frac{\tilde{\boldsymbol{\mu}}_t - \mathbf{b}_t - \mathbf{w}_{a_t}}{1 - s_t}$$

The yaw angle ψ_t of the GV should be estimated as well. Since roll and pitch are always null, its dynamics equation is

$$\dot{\psi}_t = r_t,$$

where information on the angular rate r_t is obtained from the GV's *z-axis* gyroscope, described by (3.3), hence

$$\dot{\psi}_t = \frac{\tilde{r}_t - b_r - w_r}{1 - s_r}.$$

The derivative of the biases and scale factor errors are constant for all the sensors.

Writing all the navigation equations together, the state model $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w})$ results to be

$$\begin{aligned}
\dot{\mathbf{p}}_r &= \mathbf{v}_r, \\
\dot{\mathbf{v}}_r &= R_t^i(\psi_t)\frac{\tilde{\boldsymbol{\mu}}_t - \mathbf{b}_t - \mathbf{w}_{a_t}}{1 - s_t} - \frac{\tilde{\mathbf{a}}_d - \mathbf{b}_d - \mathbf{w}_{a_d}}{1 - s_d}, \\
\dot{\psi}_t &= \frac{\tilde{r}_t - b_r - w_r}{1 - s_r}, \\
\dot{\mathbf{b}}_t &= 0, \\
\dot{\mathbf{b}}_d &= 0, \\
\dot{b}_r &= 0, \\
\dot{s}_t &= 0, \\
\dot{s}_d &= 0, \\
\dot{s}_r &= 0.
\end{aligned} \tag{3.28}$$

where the state vector is

$$\mathbf{x} = [\mathbf{p}_r \quad \mathbf{v}_r \quad \psi_t \quad \mathbf{b}_t \quad \mathbf{b}_d \quad b_r \quad s_t \quad s_d \quad s_r]^T,$$

the accelerometer measurement input is

$$\mathbf{u} = [\tilde{\mathbf{a}}_t \quad \tilde{\boldsymbol{\mu}}_d]^T,$$

and the noise vector,

$$\mathbf{w} = [\mathbf{w}_{a_t} \quad \mathbf{w}_{a_d} \quad w_r]^T$$

The state covariance matrix Q is the diagonal matrix

$$Q = \begin{bmatrix} \sigma_{a_t}^2 I_3 & & \\ & \sigma_{a_d}^2 I_3 & \\ & & \sigma_{b_r}^2 \end{bmatrix},$$

where σ_i^2 are the PSDs related to noise state i and I_3 is the identity matrix. The EKF algorithm is performed at the same frequency of the accelerometers and gyro which is set to $f_{EKF} = 100Hz$.

It can be noticed that the state model is nonlinear, aspect that justifies the choice of using an EKF and not the possibility for a standard Kalman filter.

3.5.2 Measurement model

The position fixing aiding measurement are given by the compass, the gimballed camera and the RF range system. Recalling their expression from Section 3.2, and writing them as function of the state vector $\mathbf{x}$, we can express the measurement model $\mathbf{z} = h(\mathbf{x}, \mathbf{v})$ as

$$\begin{aligned} \psi_{comp} &= \psi_t + v_{comp} \\ \mathbf{p}_{cam} &= -R_i^t(\psi_t)\,\mathbf{p}_r + \mathbf{v}_{cam}, \\ \tilde{\rho}_{RF,i} &= ||R_t^i(\psi_t)P_{RF_i}^t + \mathbf{p}_r|| + v_{RF} \quad \text{for } i = 1, ...4. \end{aligned} \tag{3.29}$$

The measurement vector is then,

$$\mathbf{z} = [\psi_{comp} \quad \mathbf{p}_{cam} \quad \tilde{\rho}_{RF,1} \quad \tilde{\rho}_{RF,2} \quad \tilde{\rho}_{RF,3} \quad \tilde{\rho}_{RF,4}],$$

the noise vecotr is

$$\mathbf{v} = [v_{comp} \quad \mathbf{v}_{cam} \quad v_{RF}]^T,$$

And the measurement noise covariance matrix is

$$R = \begin{bmatrix} \sigma_{comp}^2 & & \\ & \sigma_{cam}^2\, I_3 & \\ & & \sigma_{RF}^2\, I_4 \end{bmatrix}$$

Also the measurement model is nonlinear, so the EKF choice is again justified.

The scheme of the EKF can be found at Figure 3.4.

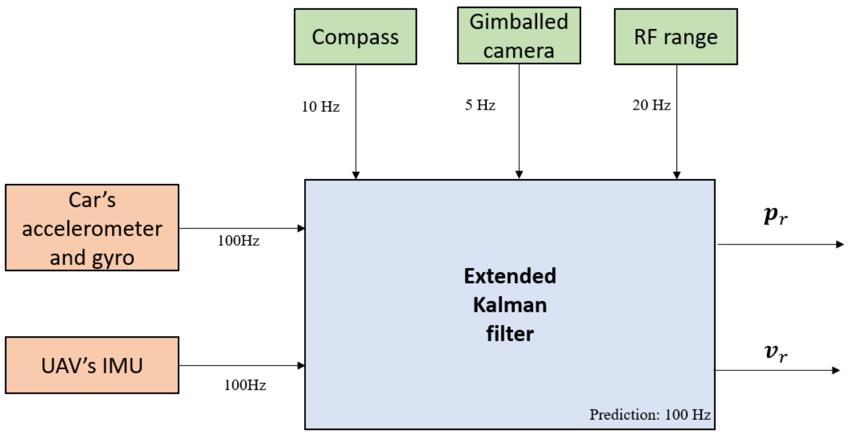

Figure 3.4: Scheme of the EKF

Chapter 4

Control system

The objective of this chapter is to present the control system architecture for an UAV capable to autonomously land on top of a moving GV. As already mentioned in Section 1.5, the control system consists in three main modules:

1. **Integrated navigation system**: widely discussed in Chapter 3, it is responsible to locate the UAV with respect to the GV's landing target and returns the states of relative position and velocity.

2. **Trajectory generation module**: from the state estimate of the navigation system, it generates set-point positions describing the nominal trajectory that the UAV should follow.

3. **Tracking control module**: a built-in controller running in the flight control unit that tracks the set-point position coming from the trajectory generation module. It consists in a Proportional Integral Derivative (PID) controller that uses estimations of position, velocity and attitude performed by an inner integrated navigation system.

The design of the tracking module is beyond the purpose of this thesis, hence a discrete-time linear system obtained by black box model identification of the UAV equipped with a tracking controller unit was used. This chapter is focused on the description of the trajectory generation module, the tuning of its parameters and finally a discussion on the control of the gimballed camera.

4.1 Trajectory generation module

The trajectory generation module provides the input for the tracking controller in terms of set-point position $\mathbf{p}_d^o = [N_d^o, E_d^o, D_d^o]^T$ of the UAV in NED coordinates.

The problem is three-dimensional, but, for sake of simplicity, it can be decoupled in: *horizontal control* for set-point generation in North-East coordinates and *vertical control* for trajectory along the Down axis.

4.1.1 Horizontal control

The strategy adopted for the trajectory generation in North-East coordinates follows the work of G. Gozzini [6] of the landing problem of a drone above another moving target drone. Set-point positions are obtained from the value of desired acceleration $\mathbf{a}^o_{NE}$ computed using the EKF estimates of North ($\hat{N}_r$, $\hat{\dot{N}}_r$) and East ($\hat{E}_r$, $\hat{\dot{E}}_r$) states, and it follows a Proportional-Derivative (PD) law of the type

$$\mathbf{a}^o_{NE}(t) = \begin{bmatrix} a^o_N(t) \\ a^o_E(t) \end{bmatrix} = K_{p,NE} \begin{bmatrix} \hat{N}_r(t) \\ \hat{E}_r(t) \end{bmatrix} + K_{d,NE} \begin{bmatrix} \hat{\dot{N}}_r(t) \\ \hat{\dot{E}}_r(t) \end{bmatrix},$$

where $K_{p,NE}$ and $K_{d,NE}$ are the proportional and derivative gains. A saturation on maximum horizontal acceleration module a_{max} can be imposed, as well as a deceleration if the UAV going faster than an horizontal speed limit v_{max}, resulting in a modified acceleration input $\mathbf{u}_{NE} = [u_N \ u_E]^T$. This last is integrated twice and position set-point for the tracking control module is obtained by

$$\begin{aligned}
\dot{N}^o_d(t) &= \dot{N}_d(t_0) + \int_{t_0}^{t} u_N(\tau)d\tau, \\
\dot{E}^o_d(t) &= \dot{E}^o_d(t_0) + \int_{t_0}^{t} u_E(\tau)d\tau, \\
N^o_d(t) &= N_d(t_0) + \int_{t_0}^{t} \dot{N}^o_d(\tau)d\tau, \\
E^o_d(t) &= E_d(t_0) + \int_{t_0}^{t} \dot{E}^o_d(\tau)d\tau.
\end{aligned} \tag{4.1}$$

In discrete time it is computed as

$$\begin{aligned}
\dot{N}^o_d(k) &= \dot{N}_d(k-1) + u_N(k)T_{int}, \\
\dot{E}^o_d(k) &= \dot{E}_d(k-1) + u_E(k)T_{int}, \\
N^o_d(k) &= N_d(k-1) + \dot{N}^o_d(k)T_{int}, \\
E^o_d(k) &= E_d(k-1) + \dot{E}^o_d(k)T_{int},
\end{aligned} \tag{4.2}$$

where, T_{int} is the integration time-step and the integration initial conditions are the initial position and velocity of the UAV at time $t_0 = 0\,s$.

The horizontal control scheme can be found at Figure 4.1.

4.1.2 Vertical control

The entire landing manoeuvre can be split into two phases. An *approach phase* is done keeping the drone at a constant altitude and terminates when the UAV has stabilized over the landing target; afterwards, a *descend phase* can start and

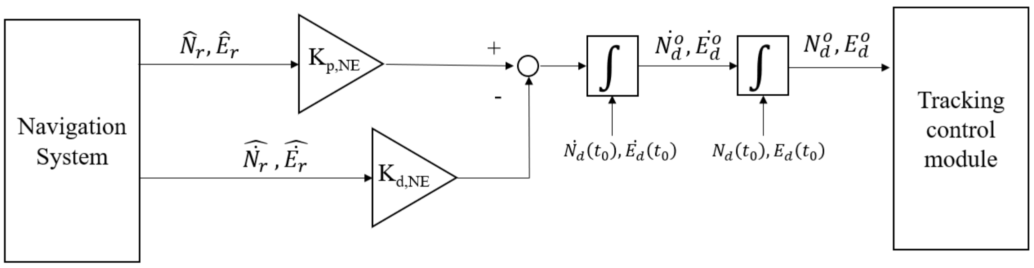

Figure 4.1: Trajectory module for horizontal control

terminates when the UAV touches the landing pad. The switch between the two phases is done by means of a safety check that guarantees the drone to be at a sufficiently close position with respect to the landing target in the horizontal direction.

Safety zone area

The descend phase can start only when the UAV satisfies the horizontal relative distance condition

$$S(t) = \sqrt{\hat{N}_r^2(t) + \hat{E}_r^2(t)} \le S^o(\hat{D}_r), \tag{4.3}$$

which uses a safety objective parameter S^o that depends on relative altitude $\hat{D}_r$ and defined as

$$S^o(\hat{D}_r) = m_s\,\hat{D}_r + q_s. \tag{4.4}$$

Coefficients m_s and q_s are computed by

$$m_s = \frac{d_2 - d_1}{2h_s},$$
$$q_s = \frac{d_1}{2},$$

where $d_1 = 0.5\,m$, $d_2 = 1\,m$ and $h_s = 2\,m$. This condition is interpreted by the fact that the descend starts only when the UAV is inside an area delimited by a truncated cone with lower base circle diameter d_1 at the landing target height and upper base circle diameter d_2 located at a distance h_s above. A representation can be found at Figure 4.2.

Descend algorithm

During the approach phase, a relative altitude of $h_s = 2\,m$ is kept constant. When the UAV satisfies the safety condition (4.3), a third order polynomial trajectory on relative Down direction $\tilde{D}_r$ is imposed and illustrated at Figure 4.3.

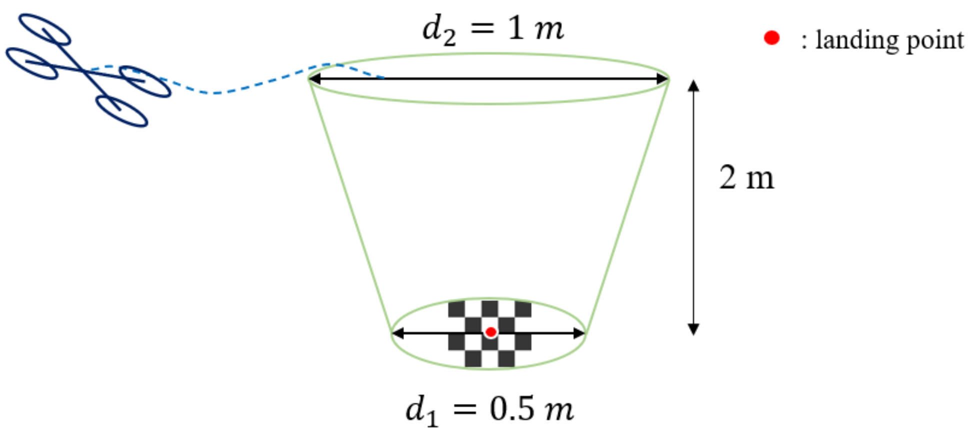

Figure 4.2: Safety cone

The trajectory has been chosen to perform a smooth and continuous descend, with final vertical speed equal to zero as to limit the landing impact. In practice, an engine cut-off is imposed when the UAV reaches a relative vertical distance of $5\,cm$.

Denoting t_{start} the time at which the UAV enters the safety area and t_{touch} the time at which the UAV should touch the landing pad, the desired time interval of the descend phase $\Delta t_{land} = t_{touch} - t_{start} = 10\,s$ is imposed. The expression of trajectory $\tilde{D}_r$ results to be

$$\tilde{D}_r(t) = a\Delta t^3 + b\Delta t^2 + c\Delta t + d$$

where $\Delta t = t - t_{start}$ and (a, b, c, d) are the coefficients of the third order polynomial. These coefficients are computed by imposing the following constraints

$$\begin{aligned}
\tilde{D}_r(t_{start}) &= h_s, \\
\tilde{D}_r(t_{land}) &= 0, \\
\dot{\tilde{D}}_r(t_{start}) &= 0, \\
\dot{\tilde{D}}_r(t_{land}) &= 0,
\end{aligned}$$

which represent, in order, the constraint of initial relative height when starting the descend, the trivial height at landing moment, initial and final speed of the

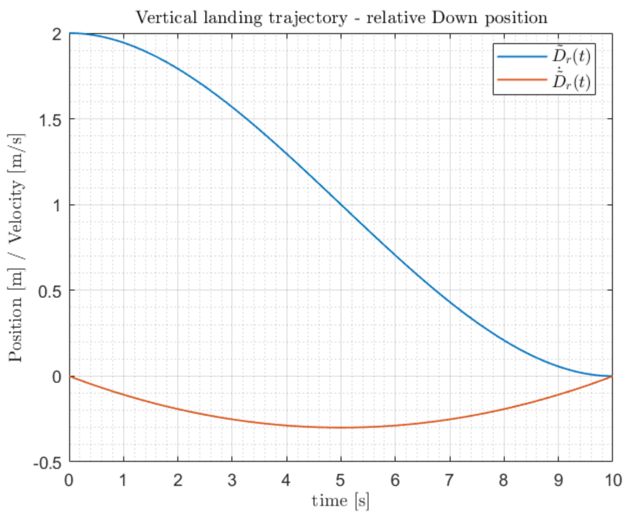

Figure 4.3: Imposed relative down trajectory

descend phase (both equal to zero). Thus, the values (a, b, c, d) result to be

$$\begin{aligned} a &= \frac{2h_s}{\Delta t_{land}^3}, \\ b &= -\frac{3h_s}{\Delta t_{land}^2}, \\ c &= 0, \\ d &= h_s. \end{aligned}$$

Following a similar strategy of the horizontal control, a vertical set-point acceleration a_D^o is computed from a PD law that uses EKF estimates and desired trajectory of the type

$$a_D^o(t) = K_{p,D}(\hat{D}_r(t) - \tilde{D}_r(t)) + K_{d,D}(\dot{\hat{D}}_r(t) - \dot{\tilde{D}}_r(t)), \tag{4.5}$$

where $K_{p,D}$ and $K_{d,D}$ are the proportional and derivative gains. Acceleration is then integrated twice to obtain vertical position set-point for the vertical tracking control module D_d^o by

$$\begin{aligned} \dot{D}_d^o(t) &= \dot{D}_d\,(t_0) + \int_{t_0}^{t} a_D^o(\tau) d\tau, \\ D_d^o(t) &= D_d\,(t_0) + \int_{t_0}^{t} \dot{D}_d^o(\tau) d\tau. \end{aligned} \tag{4.6}$$

In discrete time it is computed as

$$\begin{aligned} \dot{D}_d^o(k) &= \dot{D}_d(k-1) + a_D^o(k) T_{int}, \\ D_d^o(k) &= D_d(k-1) + \dot{D}_d^o(k) T_{int}, \end{aligned} \tag{4.7}$$

where integration initial condition is the initial vertical position of the UAV at time t_0.

Alternative techniques for the descend algorithm like bang-zero-bang [6], constant vertical speed [4] or an imposed vector field [7] can be thought as well.

4.2 Gains control tuning

In this section, a methodology to compute the values of the trajectory control module gains ($K_{p,NE}$, $K_{d,NE}$, $K_{p,D}$, $K_{d,D}$) based on a pole placement approach is proposed.

For sake of simplicity, and considering the dynamics on the three axis decoupled, the discussion will be made on the dynamics along the North axis. The same principles, however, are applied to East and Down direction. Let's consider a generic expression of the dynamics along North direction of the UAV equipped with the tracking control module, described by the differential equation

$$\begin{aligned} \dot{\mathbf{x}}_N &= f_N(\mathbf{x}_N, N_d^o), \\ \mathbf{y}_N &= \begin{bmatrix} N_d \\ \dot{N}_d \end{bmatrix} = h_N(\mathbf{x}_N, N_d^o), \end{aligned} \tag{4.8}$$

where f_N, g_N are generic functions, $\mathbf{x}_N$ is the dynamic state, N_d^o is the desired set-point position input and $\mathbf{y}_N$ is the output.

In this work, system (4.8) is described by a discrete-time linear system obtained by model identification. In general, f_N and g_N can also be unknown functions, but with the possibility to measure output responses $\mathbf{y}_N$ from imposed step input as we will see later.

In this section, the approximation of (4.8) by a linear second order system and the pole placement approach to compute the PD gains of the trajectory control module are presented.

4.2.1 Second order approximation

A generic asymptotically stable system of the second order is described by

$$\ddot{N}_d + 2\omega_N \xi_N \dot{N}_d + \omega_N^2 N_d = \omega_N^2 N_d^o \tag{4.9}$$

where ω_N and ξ_N are the natural frequency and the damping factor, N_d is the state and N_d^o is the input. The response in time domain of (4.9) to a step input of magnitude μ is

$$N_d(t) = \mu \left[1 - \frac{1}{\sqrt{1-\xi_N^2}} e^{-\xi_N \omega_N t} \sin\left(\omega_N \sqrt{1-\xi_N^2}\, t + \alpha \right) \right], \tag{4.10}$$

where $\xi = \cos(\alpha)$.

The percentage overshoot $PO_{\%}$ is defined as the percentage difference between the maximum value of $N_{d,max}$ and final value μ. It can be demonstrated that it depends only by the damping factor and its expression is

$$PO_{\%} = 100\frac{N_{d,max} - \mu}{\mu} = 100e^{-\frac{\xi_N \pi}{\sqrt{1-\xi_N^2}}}.$$

The settling time T_{a2} is defined as the time in which the value $N_d(t)$ reaches and stays at $\pm 2\%$ of its final value μ. From an approximation of formula (4.10) we get

$$N_d(t = T_{a2}) \approx \mu\left(1 - e^{-\xi_N \omega_N T_{a2}}\right) = 0.98\mu, \tag{4.11}$$

Even if system (4.8) is not of the second order, but is asymptotically stable and with a response similar to a second order one, we can compute ω_N and ξ_N from $PO_{\%}$ and T_{a2} of its step resonse by

$$\begin{aligned} \xi_N &= \frac{-\ln\left(\frac{PO_{\%}}{100}\right)}{\sqrt{\pi^2 + \ln^2\left(\frac{PO_{\%}}{100}\right)}}, \\ \omega_N &= \frac{-\ln(0.02\sqrt{1-\xi_N^2})}{\xi_N T_{a2}}, \end{aligned} \tag{4.12}$$

and thus obtain an approximation of (4.8) as a second order system described by (ω_N, ξ_N). In Laplace domain, the UAV dynamics can be characterized by a SISO transfer function of the type:

$$F(s) = \frac{N_d(s)}{N_d^o(s)} = \frac{\omega_N^2}{s^2 + 2\xi_N \omega_N s + \omega_N^2}, \tag{4.13}$$

where s is the complex variable.

4.2.2 Pole placement

Pole placement is a method employed in control system theory to place the closed-loop poles of a system in pre-determined conditions in the s-plane. This permits to locate directly the eigenvectors of the system, which control the characteristics of the response of the system.

It was chosen this method for tuning the PD gains of the trajectory generation module for the North direction, applied to the second order approximation of the UAV system of equation (4.13). It was assumed to have directly access to state $N_d(s)$, thus neglecting the navigation system and sensors dynamics. The feedback control scheme can be found at Figure 4.4.

The trajectory control module in Laplace domain is written as

$$PD(s) = \frac{K_{p,N} + K_{d,N}\, s}{s^2} \tag{4.14}$$

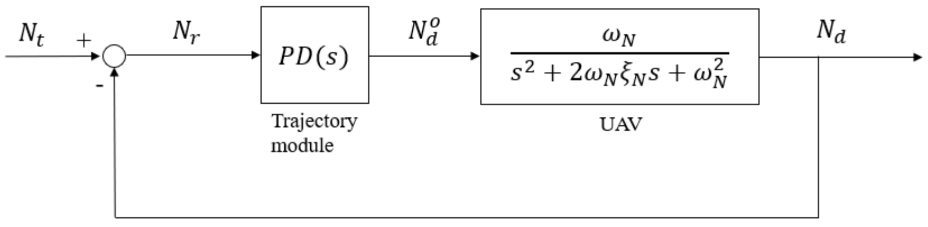

Figure 4.4: Feedback control scheme

where $K_{p,N}$ and $K_{d,N}$ are the proportional and derivative gains. The denominator of (4.14) represents the double integration that transforms desired acceleration a_N^o in set-point position output N_d^o of (4.1).

In practice, values of $K_{p,N}$ and $K_{d,N}$ were found by using the `rltool` root locus design GUI of the Control System Toolbox [17] in MATLAB, that allows to select the feedback gains of a closed loop plant by visualizing both the location of the poles and the step response. It was imposed the introduction of a zero at $z_0 = 0.4$, thus resulting in $K_{p,N} = 0.4\,K_{d,N}$. Then, the value of $K_{p,N}$ was chosen as the one giving the smallest settling time of the closed loop system to the step input, which resulted to be $T_{a2} = 7.4\,s$. The values of the gains found are

$$\begin{aligned} K_{p,N} &= 0.32, \\ K_{d,N} &= 0.8. \end{aligned} \tag{4.15}$$

Finally, these gains were tested on the trajectory control module applied on the complete system of the UAV for the North direction and the results are shown in Figure 4.5. It can be noticed that the second order approximation reproduces in a quite accurate way the complete system response. In Figure 4.6 is reported the response of the system to the ramp and it shows that the steady state error is equal to zero, which is essential for the UAV to track the GV.

The gains of (4.15) are also used for the trajectory control in East direction.

For the gains in Down direction, it was followed the same procedure as described in this section and the final results are

$$\begin{aligned} K_{p,D} &= 3, \\ K_{d,D} &= 2.2. \end{aligned} \tag{4.16}$$

4.3 Gimballed camera control

As mentioned in Section 1.1.4, the camera which is used for the detection of the checkerboard is gimballed, which means it has three-axis degree of freedom that

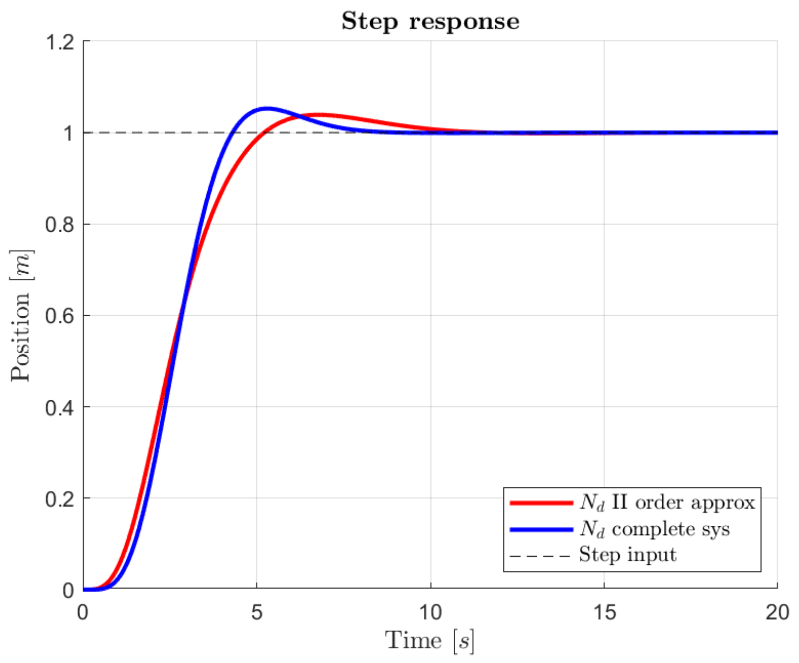

Figure 4.5: Unit step response of the approximated and complete system

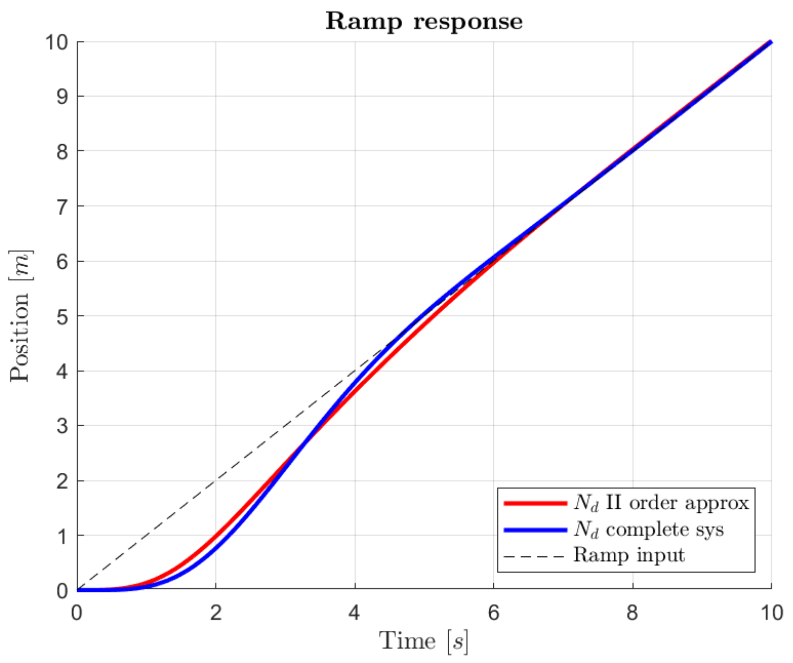

Figure 4.6: Ramp response of the approximated and complete system

allows to point any direction of space (up to limitation of maximum angle rotations of the camera). The advantage of being gimballed comes from the fact that tag detection is not influenced by the attitude of the UAV, because it does not show problems of a downward facing camera which can lose track of the visual target when the drone pitches forward to follow the GV [18].

The two angles used to control the orientation of sight of the camera are the pitch angle θ_c and yaw angle ψ_c of reference frame $\{c\}$ (the roll angle ϕ_c is constantly kept equal to zero). These angles should be as such that the camera keeps constantly track of the tag and they are evaluated from the estimated relative position $\hat{\mathbf{p}}_r$ coming from the EKF. Without loss of generality, it is assumed that point C and D are coincident, as well as point A and T. Recalling that $S(t)$ defined in equation (4.3) represents the horizontal distance between UAV and landing target, the desired camera gimbal pitch and yaw angles can be computed by

$$\theta_c = -\arctan 2(\hat{D}_r, S)$$
$$\psi_c = \arctan 2(\hat{E}_r, \hat{N}_r)$$

An illustration of the camera rotation angles can be found in Figure 4.7.

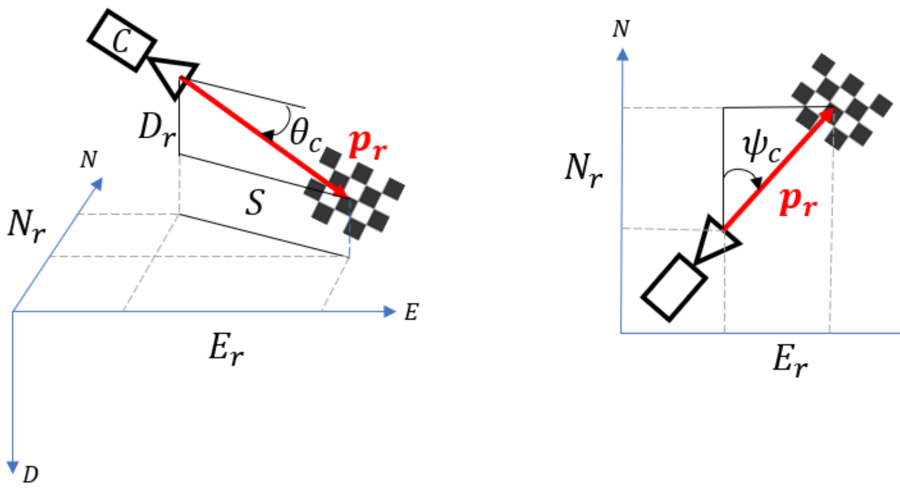

Figure 4.7: Gimballed camera rotation angles

Chapter 5

Simulation results

In this chapter, an overview of the simulation set-up is presented. Then, the results for two different scenarios of the landing approach are provided and discussed.

5.1 Simulation set-up

The period of work of this thesis unfortunately coincided with the worldwide pandemic of Covid-19 which restricted the accessibility to laboratories and compromised the possibility of developing an experimental verification of the theory presented in this work. For this reason, a computer simulation that tries to represent at best a real experimental application is addressed.

The software used was MATLAB [19] and Simulink [20]. In particular, Simulink enables a model-based integration of all the parts involved in the landing problem and the test of the design of the system. The overall simulation architecture results to be a combination of sub-systems represented by single Simulink "blocks" which can be found at the scheme in Figure 5.1 and that are:

1. UAV and ground vehicle model,
2. sensors module,
3. camera and computer vision system,
4. Extended Kalman filter,
5. trajectory control module.

The system was tested and results were analyzed and visualized in MATLAB. In this section, the description of each of sub-system block is presented.

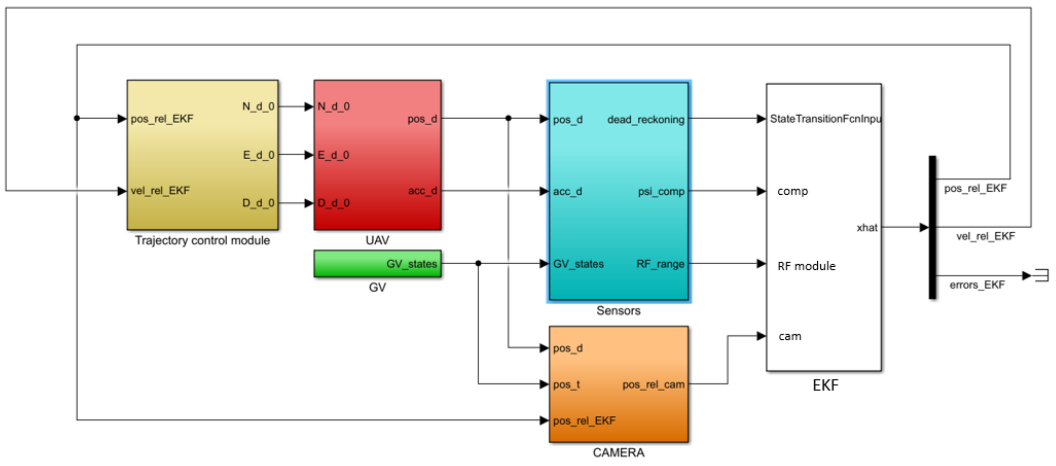

Figure 5.1: Simulink model

Feature	**ANT-R**
Weight	0.73kg
Dimensions	$19 \times 17 \times 8.5$cm
Propellers	3 blades 5045
Motors	Emax RS2205-2300KV

Table 5.1: ANT-R drone characteristics

5.1.1 UAV and ground vehicle

As mentioned in Section 1.5.1, the model used for the simulation of the UAV is a discrete-time linear model obtained by black box identification of the drone in Figure 5.2 (codename ANT-R). It is a smaller drone with respect to the M100 quadcopter which was used by Borowczyk [4] in the main reference experiment for this thesis, however the model of the ANT-R was chosen because it had already demonstrated to be sufficiently accurate from past works [6]. The characteristics of the ANT-R can be found in Table 5.1.1.

The model for the ground vehicle is the same as presented in 1.4.2. Even if it is a simple kinematic model that does not consider the complexity of the dynamics of the vehicle (e.g., mass, engine performance), it is sufficient to describe the trajectory of a generic car or truck. It is important to notice that the GV does not collaborate with the UAV by adjusting its speed to "help" the drone to land, but follows an independent trajectory which is set by imposing front wheel speed and steering angle. Future work could be the study of a possible collaborative situation, where control of the UAV and GV are synchronized [21]. The only parameter of the GV, which is axle length L, was set to $3\,m$.

Figure 5.2: UAV of reference for the simulation

5.1.2 Sensors

The sensors block, schematized in Figure 5.3, simulates the measures coming from all the instruments mounted on the UAV and GV which are affected by noise. It takes as input the real states of the two vehicles, then it adds bias and random errors as described in Section 3.2.2, which makes them deviate from ideal sensors. The values of noise parameters were chosen from commercially available and relative low-cost sensors data-sheets in order to be as realistic as possible.

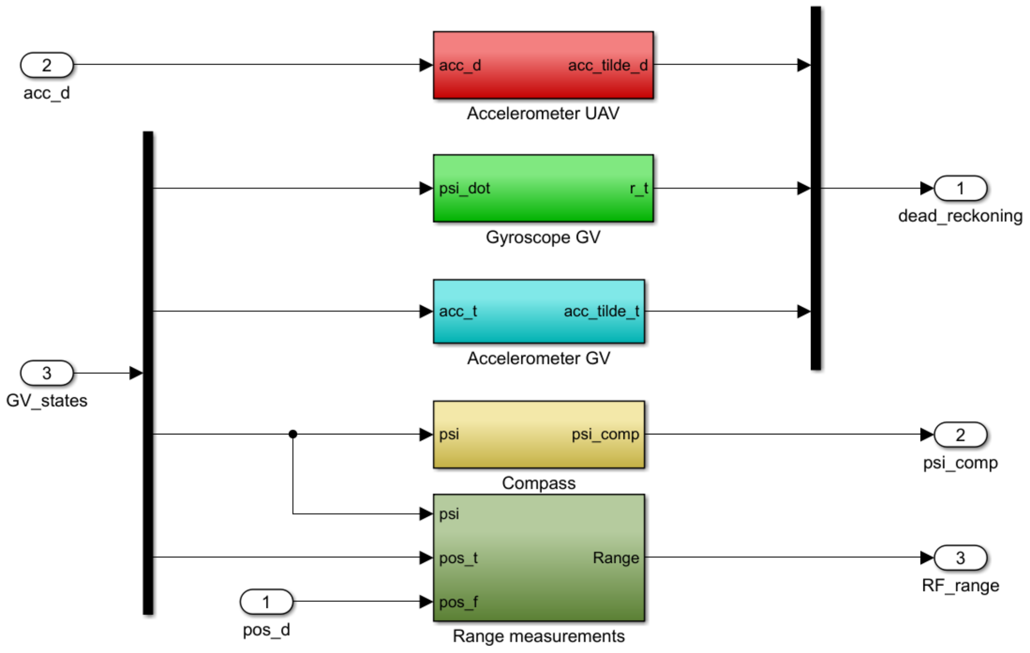

Figure 5.3: Sensors subsystem block

For the IMU installed on the drone, the reference instrument was the Ellipse 2 Micro Series from SBG systems [22], which combines good performance, low weight and cost-effectiveness. The values for this sensor should represent a realistic

Sensor	**Parameter**	**Symbol**	**Technical data**
Accelerometer	Noise density	σ_{a_d}	$57\,\mu g/\sqrt{Hz}$
	Bias stability	σ_{b_d}	$5\,mg$
	Scale factor	σ_{s_d}	$1000\,\text{ppm}$
	Frequency	f_a	$100\,Hz$

Table 5.2: UAV's accelerometer parameters

Sensor	**Parameter**	**Symbol**	**Technical data**
Accelerometer	Noise density	σ_{a_t}	$120\,\mu g/\sqrt{Hz}$
	Bias stability	σ_{b_d}	$50\,mg$
	Scale factor	σ_{s_d}	$1000\,\text{ppm}$
	Frequency	f_a	$100\,Hz$
Gyroscope	Noise density	σ_{w_r}	$0.01^\circ\,/\sqrt{s}$
	Bias stability	σ_{b_r}	$0.6^\circ\,/s$
	Scale factor	σ_{s_r}	$500\,\text{ppm}$
	Frequency	f_a	$100\,Hz$
Compass	Noise density	σ_{comp}	0.1°
	Frequency	f_{comp}	$10\,Hz$
RF range module	Noise density	σ_{RF}	$0.01\,m$
	Frequency	f_{comp}	$20\,Hz$

Table 5.3: GV's sensors noise parameters (from iPhone 6)

benchmark for general IMUs used for small drones. Its specifications are found in Table 5.2.

The measurements of accelerometers, gyroscopes and compass of the GV are assumed to be derived from a commercial smartphone. The advantage of this choice is the low-cost and accessibility to smartphones which already comprise sensors needed, together with a system that can be used to transmit data to the UAV (e.g.,Wi-Fi, Bluetooth). These self-contained sensors have lower grade of performance with respect to drone's one, but we want to demonstrate the effectiveness of the navigation system even if using a simple mobile phone as source of information for the GV. The accelerometer's and gyro of reference are respectively the Bosch BMA280 [23] and an InvenSense MPU-6500 [24] mounted on the Apple iPhone 6 and whose characteristics are found in Table 5.3. The noise density of the compass was arbitrarily set to 0.1°.

A commercially available example for the RF range distance sensor is the DW1000 Module [25]. It was not possible to have directly the value of the noise density, thus its value was assumed to be $\sigma_{RF} = 0.01\,m$.

Parameter	Symbol	Technical data
Focal length	f_{len}	$20\,mm$
Field of view	FOV	$94°$
Horizontal resolution	ρ_u	640 pixels
Vertical resolution	ρ_v	360 pixels
Frequency	f_c	$5\,Hz$

Table 5.4: DJI Zenmuse X3 camera parameters

5.1.3 Camera and computer vision

The *Camera* block is responsible for image simulation and relative position reconstruction between GV and UAV. Its scheme can be found in Figure 5.4.

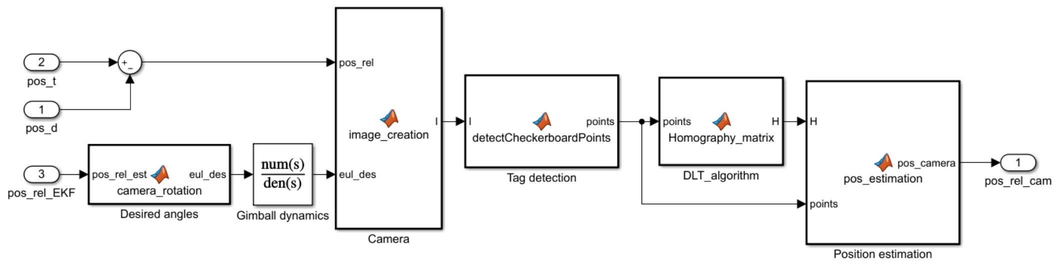

Figure 5.4: Camera Simulink block

First, the camera is pointed in space accordingly to the gimball rotation angles calculated from the estimated relative position $\hat{\mathbf{p}}_r$ and described in Section 4.3. Rotation is subjected to the dynamics of the electric motors that orient the camera and represented by the *Gimball dynamics* block. Then, *Camera* subsystem simulates the frames consistently to the camera center projection model, using the real relative position $\mathbf{p}_r$ and camera angles. A representation of some instants seen from the camera is shown in Figure 5.5.

The camera parameters were chosen from the DJI Zenmuse X3 [26] three-axis gimballed camera, simulating frames at $5\,Hz$ frequency and 640x360 pixels resolution. All the specifications of the camera can be found in Table 5.4.

It should be noticed that, since the start of the simulation, the camera is oriented towards the tag and never loses track of the checkerboard. In a real application it may be possible that the gimballed camera loses track of the tag and, for this, a wide angle downward facing camera could be used to assist the gimballed camera.

The *Tag detection* block comprises the `detectCheckerboardPoints` function that finds the 12 intersection points of the squares of the (4x5) chessboard tag. If the algorithms returns exactly 12 points the image is considered and processed in

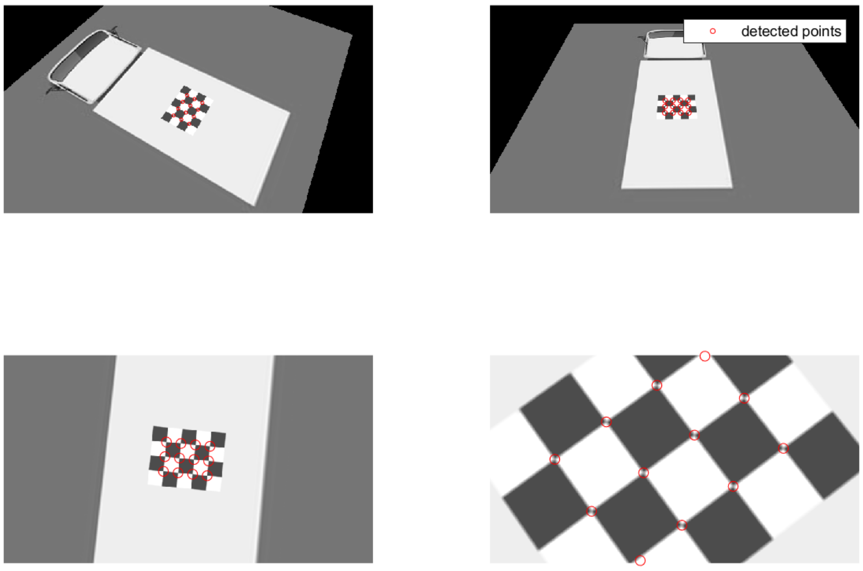

Figure 5.5: Camera frames simulation

the next steps, otherwise the measure is discarded.

The *DLT algorithm* subsystem computes the homography matrix H that is sent as input to the *Position estimation* block that finally calculates the transformation and finds the value of the relative position $\mathbf{p}_{cam}$.

Computer vision performance

The performance of the camera vision algorithm was addressed by capturing some frames of the tag, following a generic trajectory in the North-East plane described by

$$N_t(t) = \begin{cases} v_t\, t & \text{if } t \leq 5\,\mathrm{s} \\ 1 & \text{if } 5\,\mathrm{s} < t \leq 15\,\mathrm{s} \\ 1 - v_t(t-15) & \text{if } t > 15\,\mathrm{s} \end{cases},$$

$$E_t(t) = \begin{cases} 1 & \text{if } t \leq 5\,\mathrm{s} \\ 1 - v_t(t-5) & \text{if } 5\,\mathrm{s} < t \leq 15\,\mathrm{s} \\ -1 & \text{if } t > 15\,\mathrm{s} \end{cases},$$

$$D_t(t) = 0,$$

	North error $[m]$	East error $[m]$	Down error $[m]$
Mean	-0.0019	-0.0027	0.0031
Std. dev.	0.0220	0.0172	0.0167

Table 5.5: Mean and standard deviation of the camera error

where $v_t = 0.2\,\mathrm{m/s}$ is the speed at which the tag moves along a Γ-shape path that can be visualized in Figure 5.6. The camera, initially located at $\mathbf{p}_c(0\,\mathrm{s}) = [0, 0, -1]^T$, points the tag during all the time and moves away from the horizontal plane, describing a trajectory on the vertical axis

$$D_c(t) = -1 - v_c\,t, \tag{5.1}$$

where $v_c = 0.0286\,\mathrm{m/s}$ is the vertical speed.

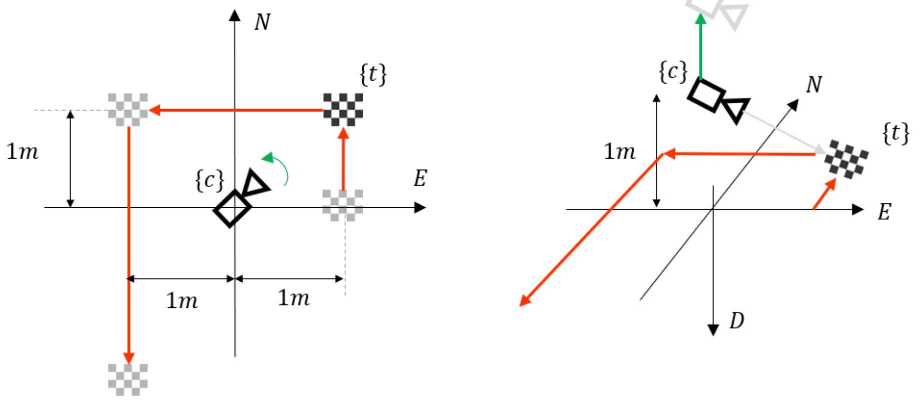

Figure 5.6: Tag and camera movement for vision test

The results of the test are shown in Figure 5.7. We can notice that until $25\,s$, which corresponds at a line-of-sight distance of approximately $2.50\,m$, the camera detects accurately the position of the tag (errors are shown in Figure 5.8). After this distance, the position estimation is abruptly imprecise; this is interpreted as a consequence on the ambiguity of detection of the points from the `detectCheckerboardPoints` algorithm. In reality, a smoother degradation of the results is rather expected, given also from the uncertainties of the camera parameters, which in this work were assumed as precisely known.

The results in terms of error's mean and standard deviation on NED position estimation until $25\,s$ of the test are collected in Table 5.5.

5.1.4 Extended Kalman filter

The integration and sensor fusion of all the measurements was done by the *Extended Kalman filter* block. It is a built-in Simulink subsystem that uses non-

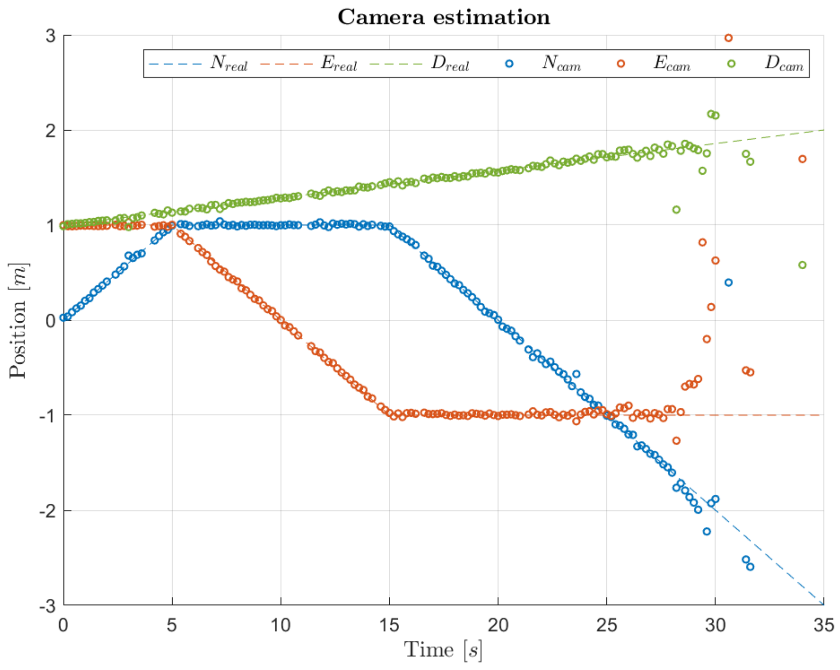

Figure 5.7: Camera position estimation

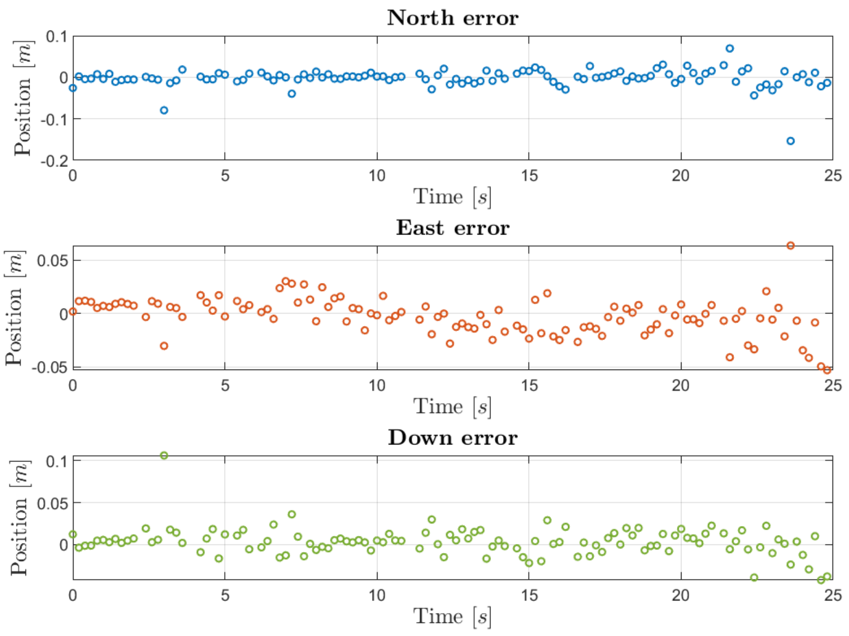

Figure 5.8: Camera position estimation error for distance up to $2.5\,m$

linear state transition and measurements functions and performs an estimation algorithm as presented in Section 3.4. The state model is the one of equation (3.28) and the measurement model is (3.29).

The discrete-time system noise covariance matrix Q_k is

$$Q_k = \begin{bmatrix} \sigma_{a_t}^2 I_3 & & \\ & \sigma_{a_d}^2 I_3 & \\ & & \sigma_{w_r}^2 \end{bmatrix},$$

where I_3 and I_4 are the identity matrices and covariance values on the diagonal were taken from the noise parameters of the accelerometers and gyros described above. The measurement noise covariance matrix is

$$R = \begin{bmatrix} \sigma_{comp}^2 & & \\ & \sigma_{cam}^2 I_3 & \\ & & \sigma_{RF}^2 I_4 \end{bmatrix}$$

where values of covariances of compass and RF range module are the same as the one in Table 5.3. From the camera performance test discussed in the previous section, the camera covariance is set to $\sigma_{cam}^2 = (0.02)^2\, m^2$. The EKF block is capable to manage up to 5 position fixing measurements coming at different sample rates.

A set of conditions are required for the initialization of the EKF algorithm, more specifically initial state and covariance estimates. The initial state estimate $\hat{\mathbf{x}}_0$ is composed by the real position, velocity and GV's yaw, added with noise to simulate the uncertainties on the knowledge of the initial real state (respectively $\tilde{\mathbf{p}}_r$, $\tilde{\mathbf{v}}_r$ and $\tilde{\psi}_t$), whereas for the biases and scale factors, they were initially set as zero. Thus the initial state estimate results to be

$$\hat{\mathbf{x}}_0 = [\tilde{\mathbf{p}}_r \quad \tilde{\mathbf{v}}_r \quad \tilde{\psi}_t \quad 0_{1\text{x}9}]^T.$$

The initial covariance is used to express the confidence in the initial state vector; more we trust in the initial guess and smaller the initial covariance values will be; on the contrary if we don't have prior knowledge about the initial state variables, the values of Σ_0 may be higher [27]. The chosen value for the simulation was

$$\Sigma_0 = 10^{-2} I_{16}. \tag{5.2}$$

5.1.5 Trajectory control module

The PD controller that generates the desired trajectory points $[N_d^o, E_d^o, D_d^o]^T$ follows the strategy presented in 4.1. It comprises also the algorithm that checks if the UAV is inside the safety area for starting the descent and defines if the land on the GV has occurred from evaluating the relative Down position, thus stopping the simulation.

5.2 Results

In this section, the simulation results for the landing approach of the drone on top of the GV is presented. Two different scenarios of possible trajectories are going to be discussed:

1. **Straight line**: from stop position, the GV accelerates until reaching a desired speed, keeping a constant direction of the GV during all the time.

2. **Change of direction**: the GV accelerates, then performs a turn of $90°$.

The first case was chosen to test the overall system performance and the behaviour of the navigation system for a simple situation.

The second case was addressed to validate the robustness of the system even in case of abrupt changes of direction and check if the UAV can still keep track of the GV.

5.2.1 Straight line

In this first case, the landing of the UAV on the GV moving in a straight path is tested. The GV is initially located at position $\mathbf{p}_t = [0, 1, -0.5]^T$, then accelerates with a constant value $a_{sl} = 0.5\, m/s^2$, until reaching the maximum speed of $v_{sl} = 5\, m/s$, keeping during all the procedure a null steering angle and a constant yaw of $\psi_t = 30°$.

The initial location of the UAV is at $\mathbf{p}_d = [0, 0, -2.5]^T$ and it has null velocity, thus meaning it is hovering above the GV in a position not too far from the landing target, which makes the camera capable to detect the position of the tag since the beginning. This work does not consider the approach phase of the UAV which may start from a far distance where the tag is not visible. Further developments can be the implementation of a navigation system that may use other information (e.g., GPS) when vision is not available, simulating an approach of the GV even from far. In this case, other types of control system could be used, for example Proportional-Navigation [4].

The 3D trajectory is visualized in Figure 5.9. Figure 5.10 shows the estimation of NED relative position from the camera that detects the tag until approximately instant $t = 19\, s$, when the checkerboard goes outside the field of view because the camera is too close to it. After that moment, the only position fixing measures comes from the RF range module.

The results for the position estimates of the navigation system are shown in Figure 5.11 and 5.12, with corresponding errors collected in 5.13. We notice that the error in the estimation of relative position remains under the value of approximately $10\, cm$, which seems to be enough for the landing manoeuvre to be performed (in Figure 5.14 the horizontal relative trajectory). As expected, the error decrease when the UAV moves closer to the landing point because the accuracy of the camera grows (the checkerboard is “bigger” in the image, thus with better resolution) and the RF range measurement is added in the EKF. Velocity estimates are shown in Figure 5.15 and 5.16, with corresponding errors in 5.17.

The results show that the descend starts at approximately $t = 13\,s$, when the GV is moving at maximum speed v_{sl} and the UAV has stabilized above the target. The landing is positively tested and the accuracy of the navigation system is validated.

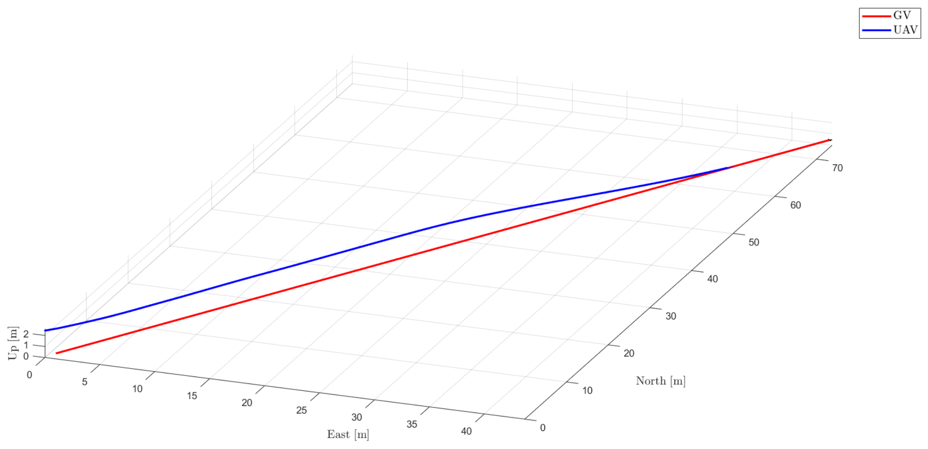

Figure 5.9: 3D straight line landing trajectory in ENU

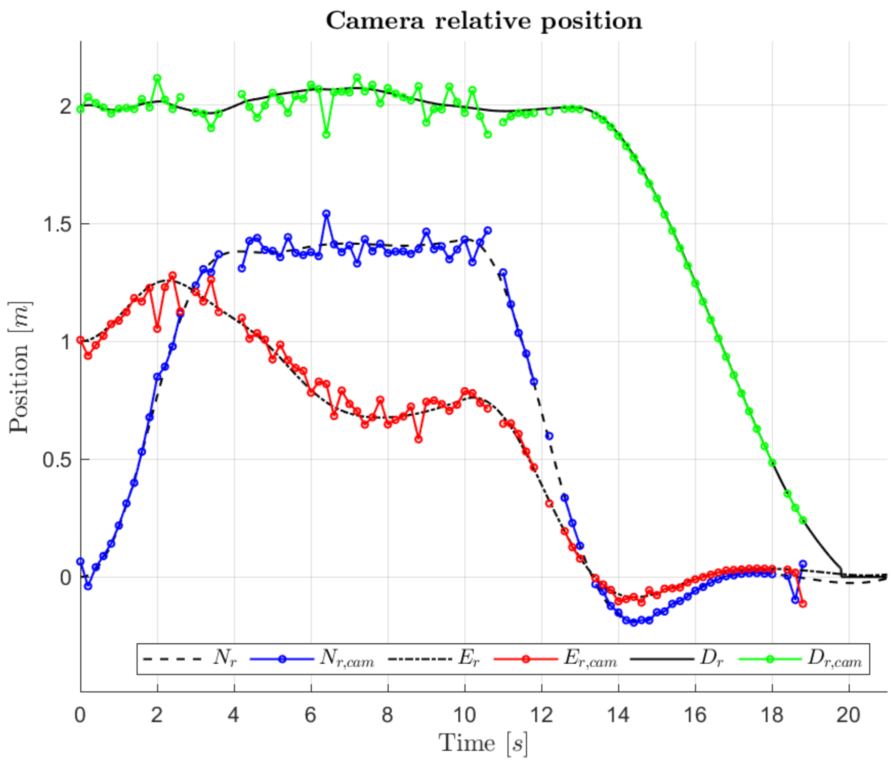

Figure 5.10: Camera position estimation

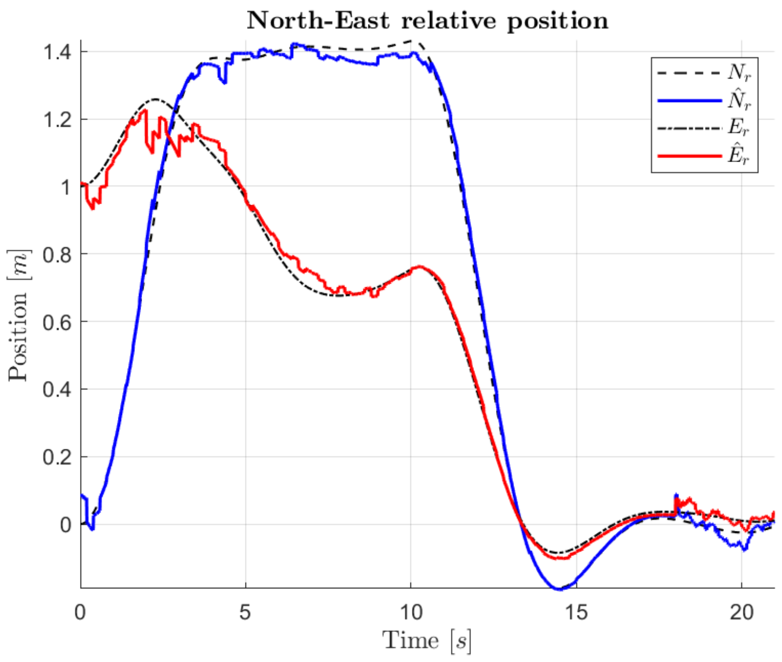

Figure 5.11: North and East relative position estimates

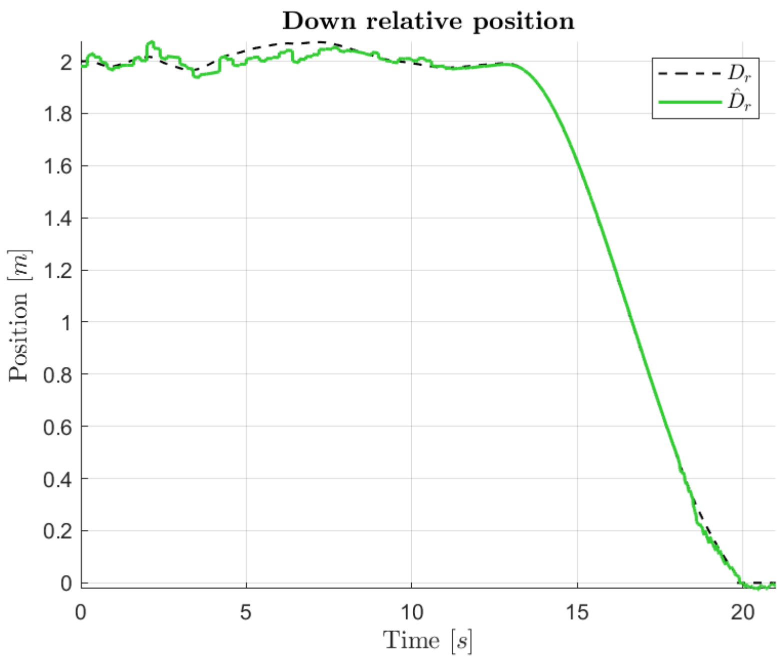

Figure 5.12: Down relative position estimates

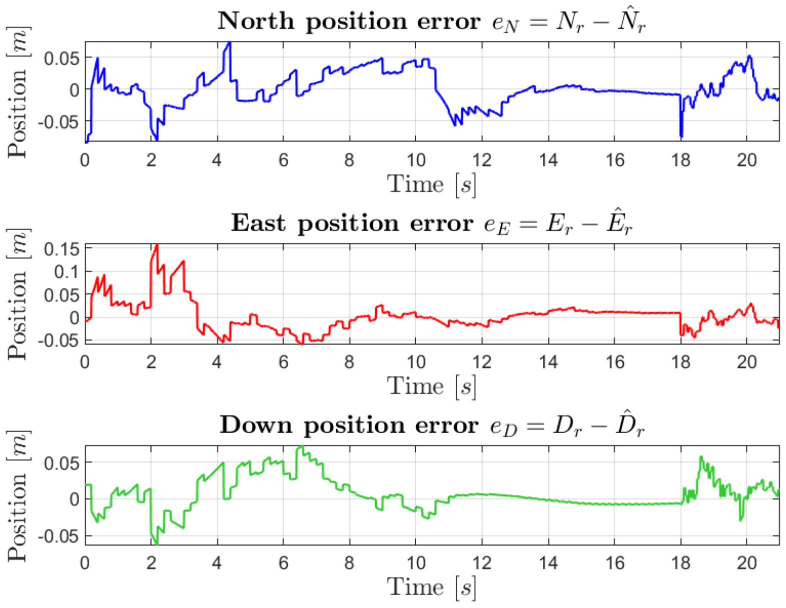

Figure 5.13: NED estimate errors

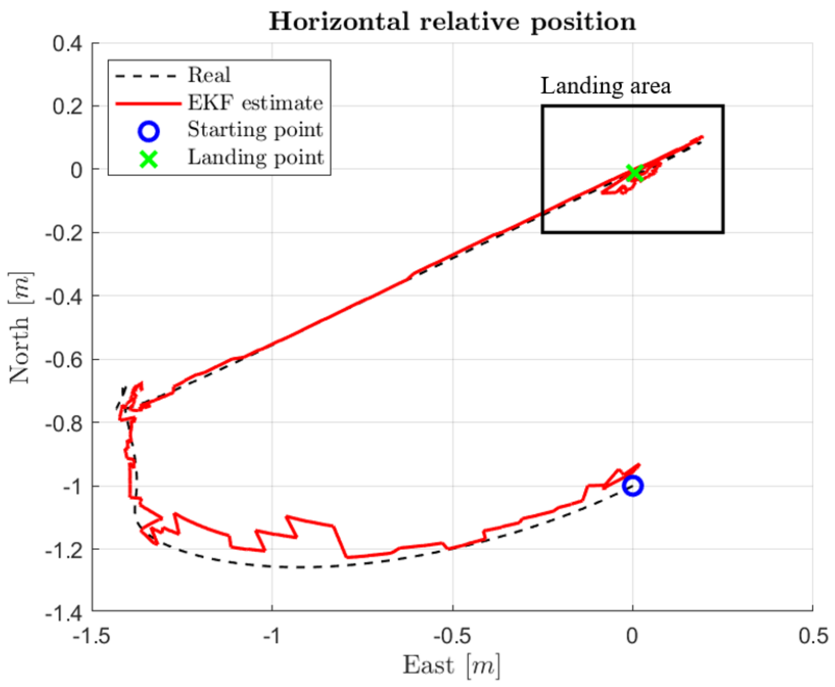

Figure 5.14: Horizontal path estimate

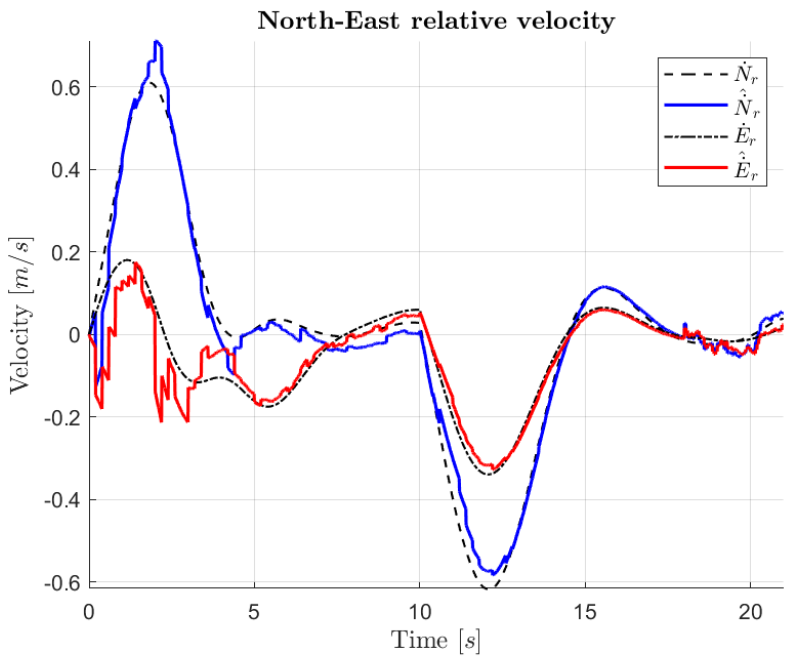

Figure 5.15: North and East relative velocity estimates

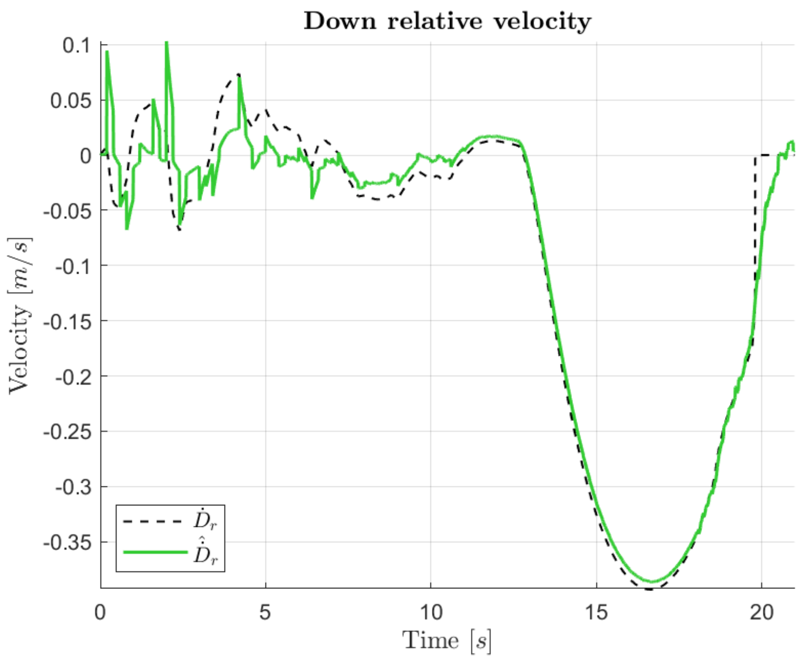

Figure 5.16: Down relative velocity estimates

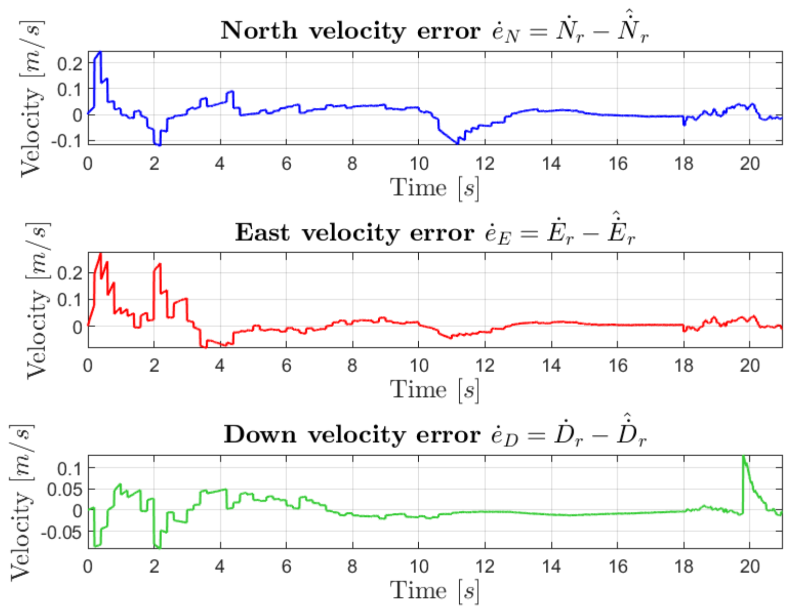

Figure 5.17: NED relative velocity errors

5.2.2 Change of direction

In the second case, the landing of the UAV on the GV which performs a turn of approximately $90°$ is tested. The GV is initially pointing North, located at $\mathbf{p}_t = [0, 1, -0.5]^T$ and accelerates of $a_{cd} = 0.4\,m/s^2$ until reaching a maximum speed of $v_{cd} = 2\,m/s$, . Afterwards, an input of steering angle $\theta_{st} = 40°$ is given from instant $t = 8\,s$ until $t = 13\,s$, which causes a turn of the GV of almost $90°$ and makes it point East. The initial position of the UAV is the same of the previous case.

The 3D trajectory of both vehicles is shown in Figures 5.18 and 5.19. In Figure 5.20 is visualized the trajectory from top-view. We notice that the UAV aligns with the GV in the first moments. When the GV start to turn, the UAV is capable to keep track of it, even if with some misalignment on the horizontal plane. When the GV returns to follow the straight path, the UAV re-aligns and can perform the land.

At approximately $t = 7\,s$ a first attempt of the descend is performed, but the start of the turn of the GV makes the UAV go outside the safety area and return to a relative height of $2\,m$. The drone is not capable to keep a sufficient track of the GV to perform the landing during the turn, but it was demonstrated that it can continue following the target during the moments when it changes direction, then land when the GV returns in a straight line.

In Figure 5.21 and 5.22 are presented the results for the position estimation of the navigation system and in Figure 5.23 the horizontal trajectory estimate. Results in terms of velocity estimates are in Figure 5.24 and 5.25.

The inputs from the trajectory control module in terms of $[N_d^o, E_d^o, D_d^o]^T$ are compared with the real absolute NED positions of the drone and GV in Figures 5.26, 5.27 and 5.28.

Also in this case, the landing is performed and the validation of the system is confirmed.

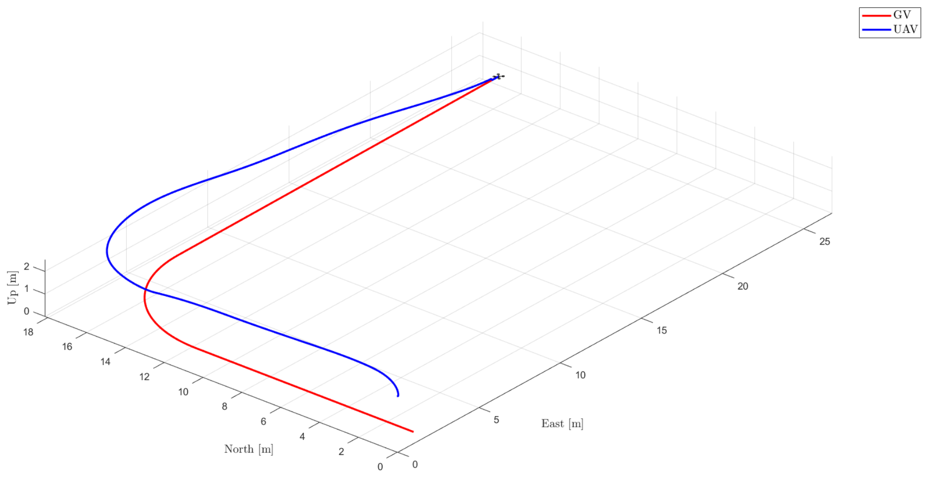

Figure 5.18: 3D landing trajectory in ENU

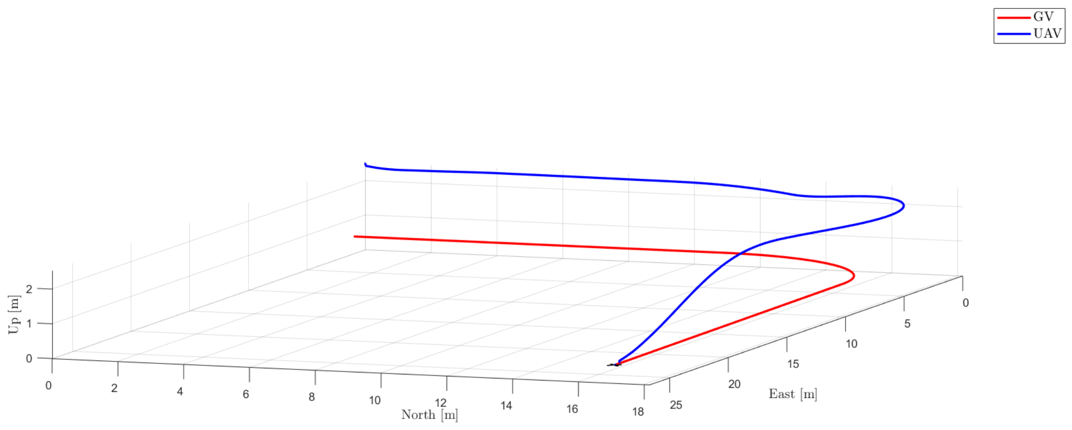

Figure 5.19: 3D landing trajectory in ENU - side view

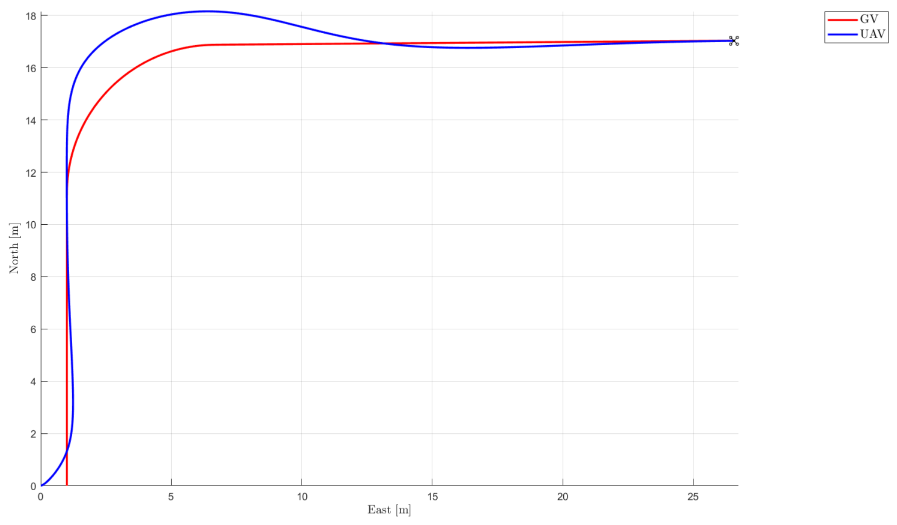

Figure 5.20: Horizontal landing trajectory

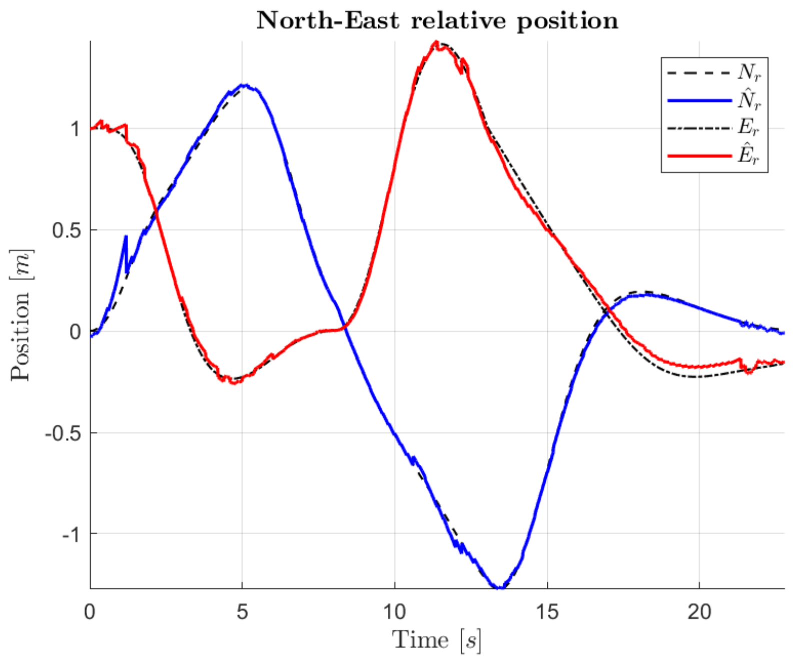

Figure 5.21: North and East relative position estimates

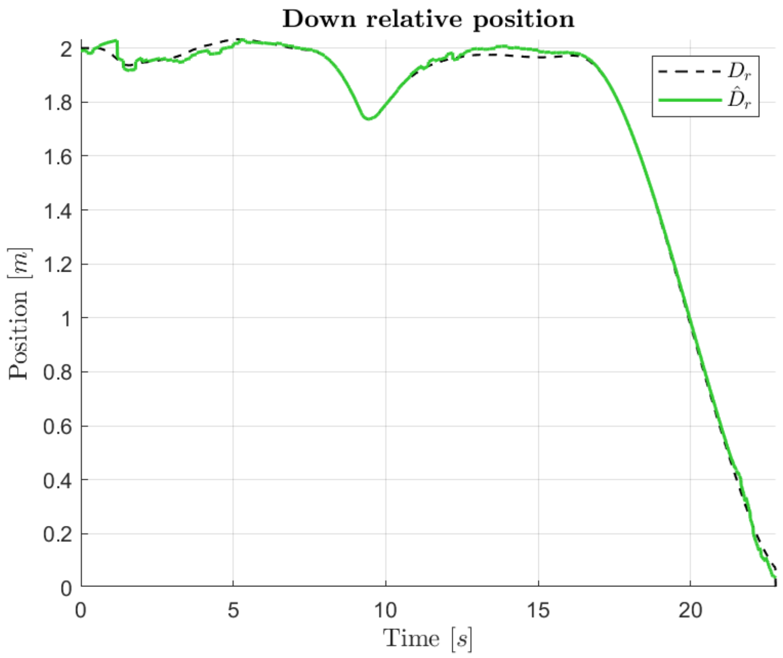

Figure 5.22: Down relative position estimates

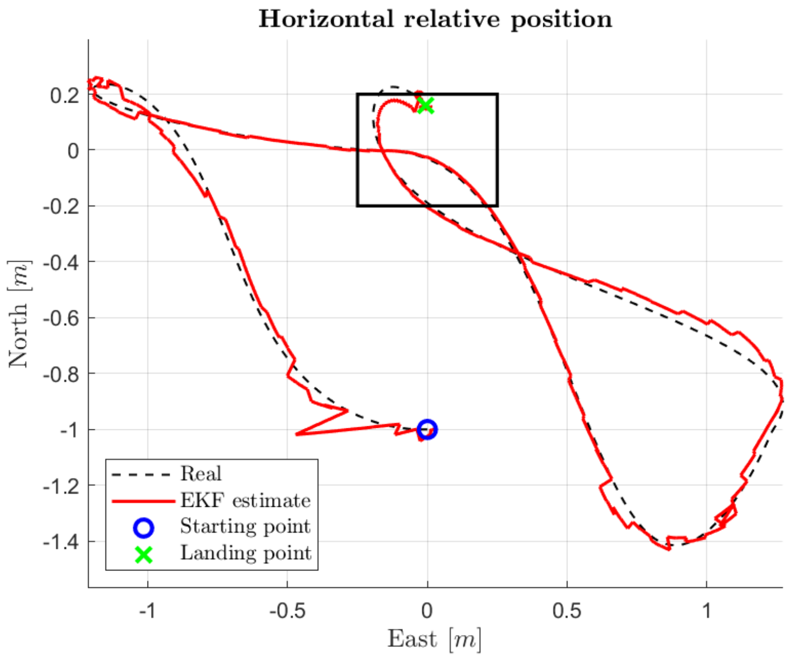

Figure 5.23: Horizontal relative trajectory estimate

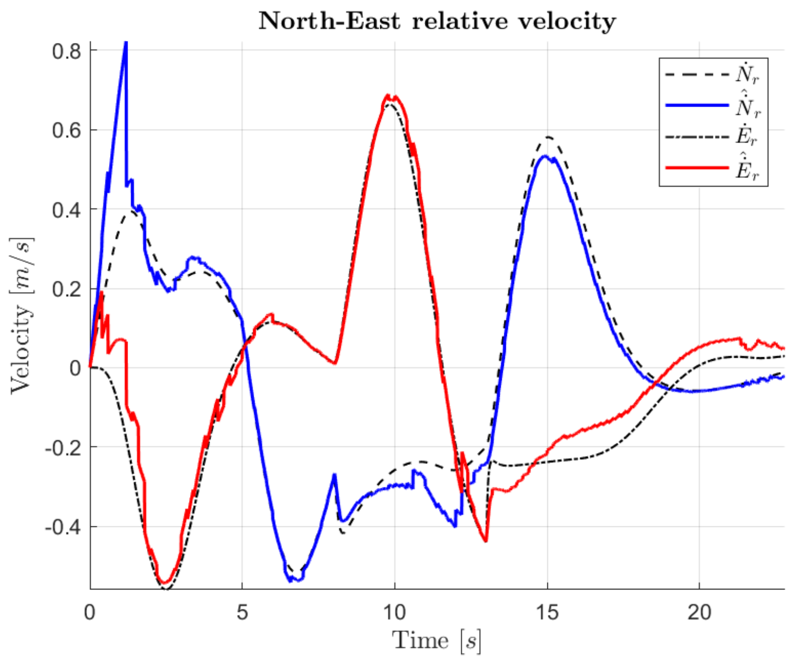

Figure 5.24: North and East relative velocity estimates

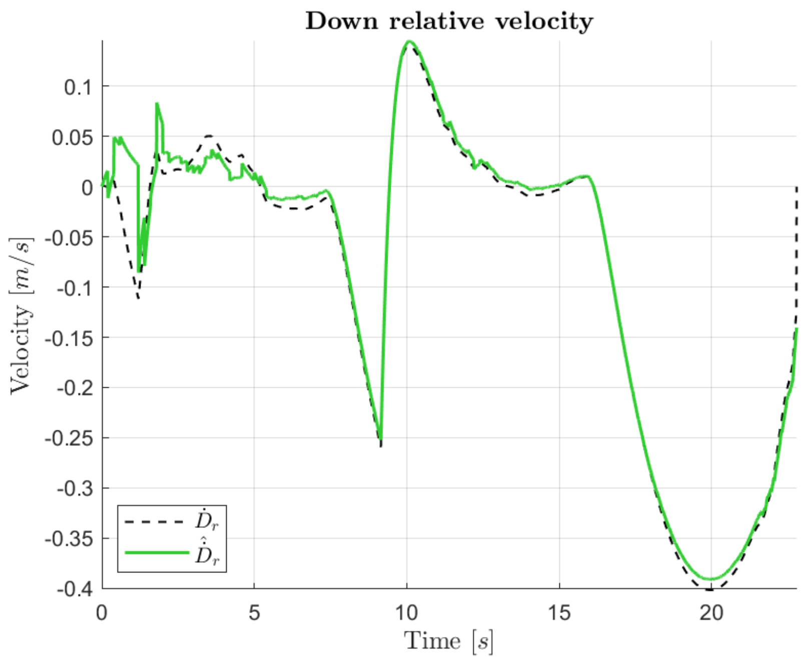

Figure 5.25: Down relative velocity estimates

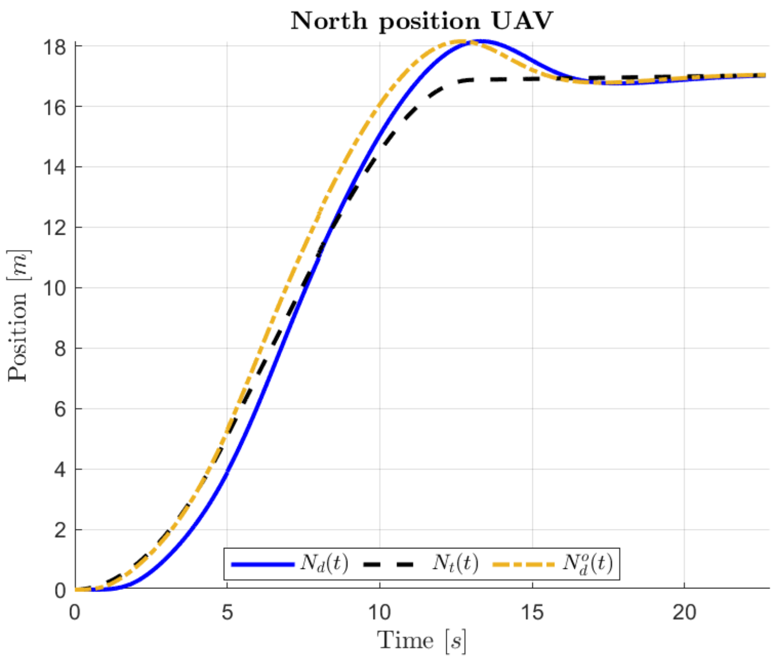

Figure 5.26: North absolute UAV trajectory

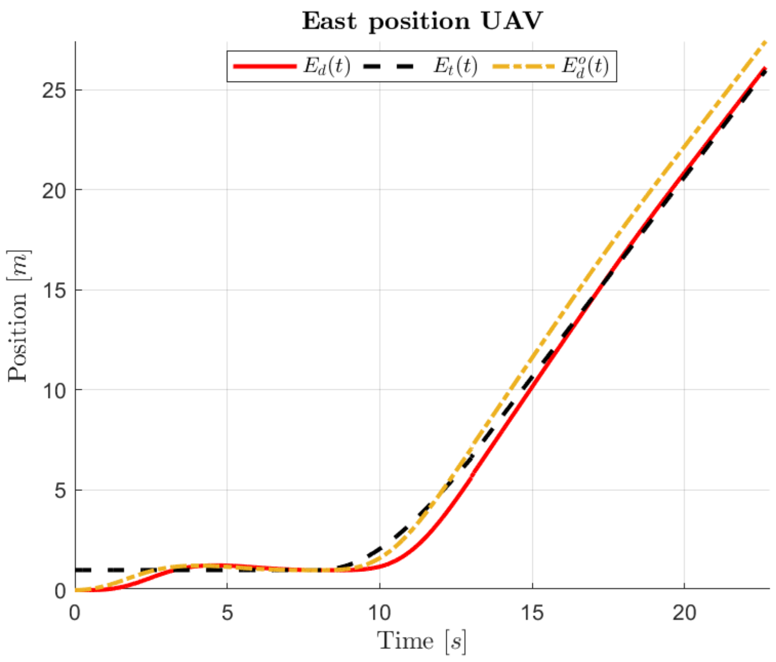

Figure 5.27: East absolute UAV trajectory

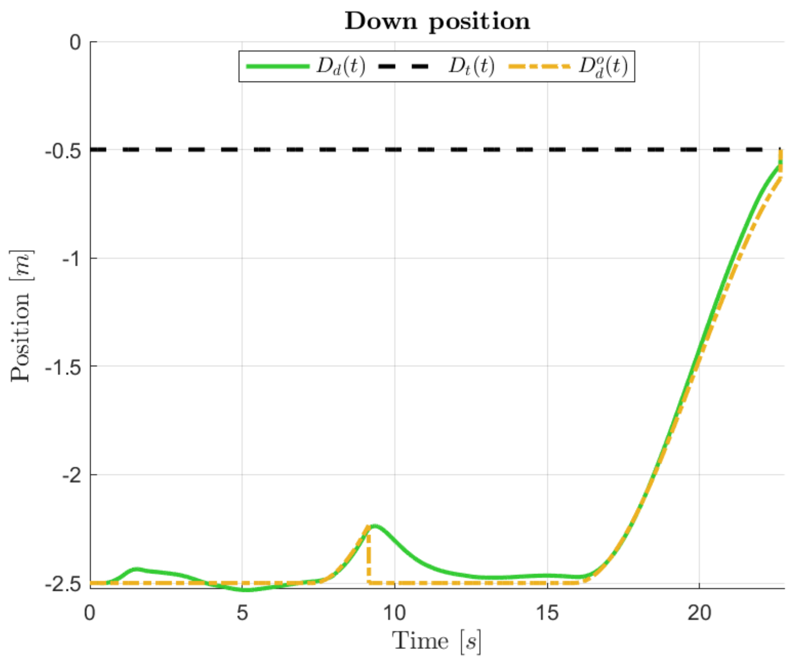

Figure 5.28: Down absolute UAV trajectory

Conclusions

The aim of this thesis was to study the autonomous landing of a multirotor UAV on top of a moving ground vehicle.

The conducted activity is mainly inspired by the work of Borowczyk et al. [4], who demonstrated the possibility to land an UAV on a GV moving at the very high speed of $50\,km/h$. Starting from this, the purpose of the work was to develop a simulation using MATLAB and Simulink for a guidance and navigation system of a drone performing the landing, potentially using low cost and off-the-shelf sensors.

A computer vision algorithm for relative position estimation between UAV and GV was presented; the central projection model was used to simulate the frames seen from the gimballed camera mounted on the UAV. The images showed the fiducial tag consisting in a (4x5) checkerboard of known dimensions which was positioned on top of the vehicle. From these frames, it was possible to reconstruct the position of the GV by means of the calculation of the Homography matrix. A test on the accuracy of the visual information showed a precise estimation of the order of magnitude of centimeters, until approximately $3\,m$ of line-of-sight distance, after that, the vision algorithm was not able anymore to detect the tag.

An Extended Kalman filter, using on a series of sensors mounted both on the UAV (IMU and camera) and the GV (accelerometers, gyro and RF distance module), has been implemented. It was responsible to estimate relative position and velocity based on the non-linear system and measurement models of the dynamics of the vehicles and sensors, eventually managing multi-rate measures. The tuning of the EKF has been executed using the variances of the errors of the involved sensors, which have been found through data-sheets of commercially available sensors. In particular, values of sensors mounted on the GV were taken from a smartphone model's ones. The filter was able to give a continuous state estimate even in the case of unavailability of measures in some instants, like the camera in the last moments of the manoeuvre. The accuracy of the navigation system was demonstrated to be precise enough for the landing to be performed.

A PD trajectory controller, for the three NED directions of motion, was specifically designed to generate set-points to be tracked by the UAV. A third order polynomial descent trajectory was imposed after the UAV aligned with the GV in the horizontal plane. The gains of the controller were found by means of a pole placement approach applied to a second order system dynamics approximation of the UAV.

Conluding, some considerations about the future developments of the simulation could be done:

- other fiducial tag libraries could be used instead of the detection of the checkerboard, such as AprilTag, ArUco or ARToolKit,
- a more precise model of the vehicles could be implemented, possibly considering the aerodynamic interaction between the moving GV and the UAV,
- a distant approach phase could be considered, where vision may not be available because the UAV too far from the GV, thus using GNSS to locate the landing target until the camera could finally detect the landing pad.

Bibliography

[1] UPS - Drone meets delivery truck. https://www.ups.com/us/en/services/knowledge-center/article.page?kid=cd18bdc2.

[2] R Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004.

[3] Amazon Prime Air. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011.

[4] 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.

[5] J. Wang and E. Olson. AprilTag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[6] G. Gozzini. Uav autonomous landing on moving aerial vehicle. Master's thesis, Politecnico di Milano, 2019.

[7] V. M. Gonçalves, R. McLaughlin, and G. A. S. Pereira. Precise landing of autonomous aerial vehicles using vector fields. *IEEE Robotics and Automation Letters*, 5(3):4337–4344, 2020.

[8] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38 – 47, 2018.

[9] 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, 2014.

[10] J. Le Ny. *ELE6209A Navigation systems - Lecture notes*. Polytechnique Montréal, 2020.

[11] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 125–134, 2008.

[12] M. Fiala. Artag, a fiducial marker system using digital techniques. page 590–596, 2005.

[13] MATLAB - computer vision toolbox. `https://www.mathworks.com/products/computer-vision.html`.

[14] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster. Automatic camera and range sensor calibration using a single shot. In *2012 IEEE International Conference on Robotics and Automation*, pages 3936–3943, 2012.

[15] P.D. Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*. GNSS technology and applications series. Artech House, 2008.

[16] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME — Journal of Basic Engineering, Vol. 82, No. 1*, pages 35 – 45, 1960.

[17] MATLAB - Control System Toolbox. `https://www.mathworks.com/products/control.html`.

[18] K. Ling. Precision landing of a quadrotor uav on a moving target using low-cost sensors. Master's thesis, Univ. of Waterloo, Waterloo, ON, Canada, 2014.

[19] MATLAB. `https://www.mathworks.com/products/matlab.html`.

[20] Simulink. `https://www.mathworks.com/products/simulink.html`.

[21] T. Muskardin, G. Balmer, S. Wlach, K. Kondak, M. Laiacker, and A. Ollero. Landing of a fixed-wing uav on a mobile ground vehicle. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1237–1242, 2016.

[22] SBG systems ellipse 2 micro series. `https://www.sbg-systems.com/products/ellipse-micro-series/`.

[23] Bosh accelereration sensor bma280. `https://www.bosch-sensortec.com/products/motion-sensors/accelerometers/bma280.html`.

[24] TDK mpu-6500 six-axis (gyro + accelerometer) mems motiontracking. `https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6500/`.

[25] DecaWave dwm1000 module. `https://www.decawave.com/product/dwm1000-module/`.

[26] DJI zenmuse x3. `https://www.dji.com/it/zenmuse-x3`.

[27] S. Musacchio. Optimal and robust uav state estimation based on gps and optical flow. Master's thesis, Politecnico di Milano, 2018.