POLITECNICO DI TORINO

Master of Science In Mechatronic Engineering

Master's Degree Thesis

UAV precise ATOL techniques using UWB technology

Supervisors
Prof. Marcello Chiaberge
Dott. Ing. Giovanni Fantin

Candidate
Gennaro Scarati

December 2021

Summary

Autonomous landing of a UAV on a mobile platform is currently one of the most explored research areas. It emerges as a powerful solution in many sophisticated civil and military applications where human intervention is not always available or sufficiently responsive. Examples of this are continuous flight tasks or long distances to be covered, where mobile charging stations are needed. It is therefore clear that in all these operations the position accuracy of both UAV and mobile platform is of vital importance.

This thesis examines the development of an autonomous landing system based on ultrawide-band ranging sensors. A pose estimation filter and a robust control algorithm are proposed, enabling precise tracking and autonomous landing both on the stationary or moving platform. Firstly, they are tested in a large number of simulated scenarios, where it is possible to model both UAV and UGV dynamics and all sensors with their realistic noise. Finally, the algorithms are implemented on the real embedded system, allowing a landing accuracy from 5cm to 10cm.

Ultrawide-band is a suitable technology for service robotics because of its high ranging precision, obstacle penetration capabilities and robustness against interference. It is in fact possible to accurately compute the relative position of both UAV and UGV by installing UWB ranging sensors on the two systems. Positioning is achieved through a least squares multilateration algorithm, which takes as input the distances given by the UWB devices and returns the relative position of the two systems in the rover reference frame. This information is very noisy and needs to be rotated in the drone reference frame for control purposes. Therefore, it is then fused with UAV sensors and UGV compass data in a loosely coupled Kalman filter, allowing up to 5cm accuracy when the drone is within 1m from the rover.

The filtered relative position estimate is then passed to a gain scheduling PID speed controller, which ensures fast tracking and acceptable overshoot in both chase and landing phases. The first step is handled by a proportional control algorithm, while the second by a proportional-integrative-derivative one. Since the main problem of the proposed landing system is the misalignment between the drone and rover compasses, this control algorithm is designed to be robust with respect to orientation errors, allowing successful landings with errors up to

40 degrees. Finally, a predictive control variant is proposed, in which the indirect UGV speed estimate is used to compute at a 10Hz rate the optimal control setpoint and allow autonomous landings at speeds above 3km/h.

Acknowledgements

I would first like to thank professor Chiaberge and the entire PIC4SeR, for giving me the opportunity to work on such a challenging yet so interesting project. In particular, special thanks go to my friend Cosimo Conte, who made working on this thesis much more enjoyable and without whose help this achievement would probably not have been possible.

My heartfelt thanks to Davide for all these years of unforgettable experiences, I am sure there will be many more to come. I wish to extend my thanks to the Compagnia D'Alta Quota for all the amazing moments we lived together.

Particular gratitude goes to my girlfriend Paola, with whom I shared months of incomparable adventure and complicity.

I would like to acknowledge Damiano Scibilia, Edoardo Fioriti and Stefan Tiberius who made each lecture of these years unbelievably entertaining.

I would like to thank Gianmarco Carrabba, Marco Ratta, Edoardo Saggio and Lorenzo Torcello for being with me during all these years of university, and particular gratitude goes to Marco, for his valuable help in writing my thesis. I also thank Sergio Scalabrino, for the countless moments spent together in these five years and Elisa Serafini, for all the laughs of the last months.

Finally, I wish to extend my special thanks to my family, who have unconditionally supported me in my studies and most importantly, in all my choices.

Contents

List of Figures

Acronyms

UAV
Unmanned Aerial Vehicle

UGV
Unmanned Ground Vehicle

UWB
Ultra-wideband

ATOL
Autonomous Take-off and Landing

KF
Kalman Filter

EKF
Extended Kalman Filter

ROS
Robot Operating System

RTPS
Real Time Publish Subscribe

PID
Proportional Integral Derivative

SLAM
Simultaneous Localization and Mapping

VO

Visual Odometry

FOV

Field of View

FCC

Federal Communications Commission

IEEE

Institute of Electrical and Electronics Engineers

LOS

Line-of-Sight

NLOS

Non-Line-of-Sight

GDOP

Geometric Dilution of Precision

TOA

Time of Arrival

TOF

Time of Flight

NED

North East Down

ENU

East North Up

FRD

Forward Right Down

FLU

Forward Left Up

Chapter 1

Introduction

1.1 Thesis objective

The goal of this work is to propose an autonomous landing system of a UAV on a moving UGV based on ultrawide-band technology and validate it through software-in-the-loop simulations and experimental testing. Therefore, the estimation algorithm, the control algorithm and the state machine regulating the UAV behavior are defined first. Then, they are simulated using the SITL package provided by PX4 autopilot, ROS2 and the Gazebo simulator. Finally, the same code is implemented on the embedded platform and tested on real flights.

1.2 Thesis organization

The thesis is divided in six main chapters, here concisely summarised.

1. **PX4 Autopilot** This first chapter introduces PX4 Autopilot, a powerful open source autopilot flight stack which is used to perform both the simulations in Gazebo and the experimental tests. Indeed, it provides a software-in-the-loop package and all the necessary APIs to interact with the simulated vehicle in Gazebo via ROS2. The same code is then implemented on a Raspberry Pi 4, which is responsible of sending the computed setpoints to the PX4 Autopilot board, that controls and stabilizes the drone.

2. **Ultrawide-band** This section briefly describes the ultrawide-band technology and introduces the concept of multilateration. Moreover, the Least Squares positioning method is presented and discussed in detail.

3. **System state estimation algorithm** In this chapter, the entire system state estimation algorithm is explained and further investigated. To this end,

first a brief overview of the Kalman Filter is given, then the entire system is mathematically modeled using discrete differential equations.

4. **System state machine** The system state estimate is then managed by a complex state machine and a suitably designed controller in order to define the setpoints to be given to the autopilot.

5. **Software-in-the-loop simulations** In this chapter, the autonomous landing system is verified by means of software-in-the-loop simulations using ROS2 and Gazebo. Furthermore, the effect of some parameters on the system response is illustrated and discussed.

6. **Experimental testing** In this last chapter, the proposed autonomous landing system is validated through several experimental tests, that show excellent results in both fixed and moving target landings.

Chapter 2

PX4 Autopilot

2.1 Overview

PX4 is a powerful open source autopilot flight stack that consists of many guidance, navigation and control algorithms. In the proposed configuration, the UAV is equipped with a board running the autopilot, that is in charge of estimating, stabilizing and controlling its state.

Its relevant characteristics are:

- Many types of supported vehicles.
- Large number of interfaceable devices and hardware selection.
- Highly configurable and several flight modalities.

The PX4 board is then connected to a companion computer, in this case a Raspberry Pi 4, which executes all the various flight algorithms and sends the commands to the autopilot. Therefore, PX4 is mainly responsible for managing the received commands and controlling the drone, as well as stabilizing it.

2.2 System architecture

The proposed architecture can be seen in picture 2.1.

It is noticeable that the system is composed of:

- Telemetry radios, necessary to communicate with the ground station.
- A power module, in charge of controlling the motors.
- Several sensors.

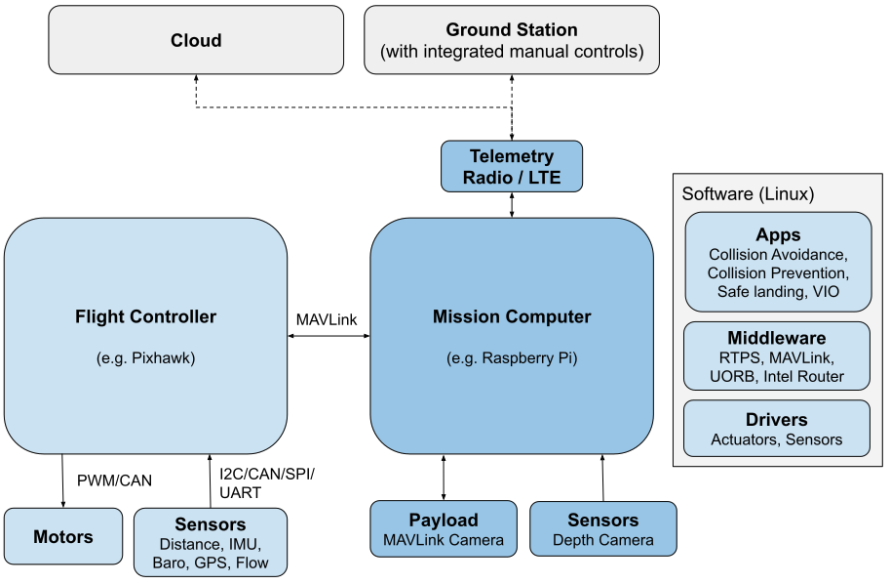

Figure 2.1: PX4 system architecture[1].

- A controller board running the PX4 flight stack, equipped with a compass, an IMU and a barometer.
- Radio control antennas, in order to manually operate the UAV.

In the proposed solution, the ground station is a Linux computer that can communicate (via wifi) with the companion computer. The latter is connected to the flight control board through UART protocol.

The most important software that runs on the flight stack consists of estimators and controllers. It includes an Extended Kalman Filter (EKF) that uses data from several sensors to estimate the UAV state. In our case, the used sensor are: a GPS-RTK module, a distance sensor, an accelerometer, a gyroscope, a barometer and a compass.

2.2.1 Controller architecture

The PX4 UAV control system can be seen in figure 2.2. This control architecture is composed of several PID controllers arranged in a cascade control fashion, where the feedback estimates (internal to each block) are computed by an Extended Kalman Filter. As can be seen in figures 2.3 and 2.4, the drone can be controlled in both position and speed. It is noticeable that the velocity control scheme shows the derivative term applied directly to the measurement, rather than to the error, in order to avoid the derivative kick. Indeed, when the setpoint changes, the error

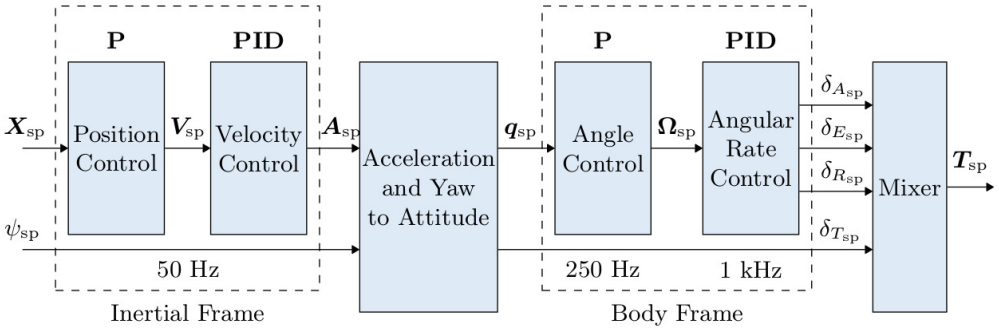

Figure 2.2: PX4 controller architecture[2].

derivative could reach extremely high values. Finally, the mixer is in charge of transforming force commands into individual motor inputs, while properly taking into account signals saturation.

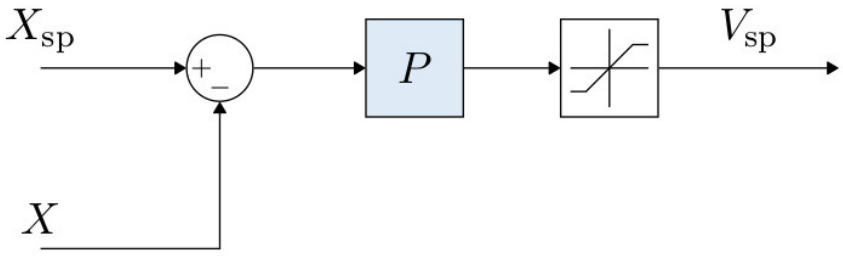

Figure 2.3: PX4 position controller[2].

2.3 Offboard control via ROS2

PX4 provides several flight and control modes that can be managed via offboard control. It is in fact possible to control the drone attitude, position, speed, and acceleration or even manage the thrust of each motor.

The offboard control consists in controlling the drone using software that runs on a companion computer, ground station or cloud. It requires that at least one positioning method is available (GPS, Visual Odometry, etc), with an accuracy above a minimum threshold. It is also possible to interface ROS2 and PX4 via PX4-ROS2 bridge, necessary to translate ROS2 commands into the corresponding UORB messages.

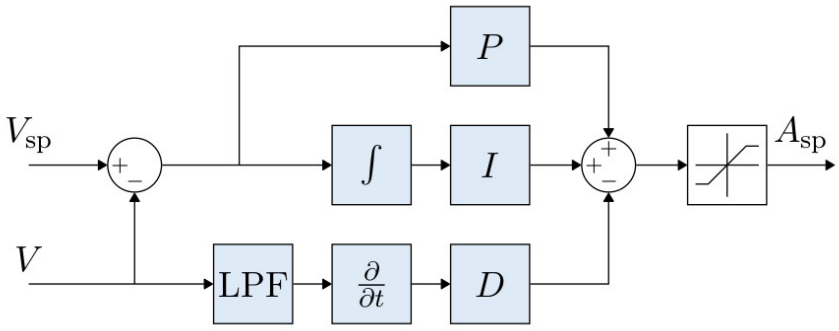

Figure 2.4: PX4 velocity controller[2].

The offboard control mode requires that the drone is armed and that the autopilot receives a continuous stream of messages, with a frequency higher than $2Hz$. Otherwise, a forced exit will occur. Moreover, any RC command will cause the offboard control to be interrupted.

2.4 Reference frames

PX4 uses FRD (X Forward, Y Right and Z Down) and NED (X North, Y East, Z Down) reference frames. The former is used for the body local frame, while the latter for the fixed inertial frame. This one is aligned with the magnetic axis, given by the compass, and is created by the autopilot at drone startup. Moreover, PX4 uses the NED convention to send and receive position and control commands in the offboard control mode. Therefore, they must be first converted to this frame.

In contrast, Gazebo and ROS use FLU (X Forward, Y Left, Z Up) and ENU (X East, Y North, Z Up) frames. This concept is summarized in table 2.1.

Reference frame	**PX4**	**ROS/Gazebo**
Body	FRD	FLU
World	NED	ENU

Table 2.1: Reference frames in ROS/Gazebo and PX4[3].

By looking at the figure 2.5, it is trivial deriving the rotation matrix to switch between ENU and NED frames. If both of them are aligned with the magnetic axis, the rotation matrix can be written as:

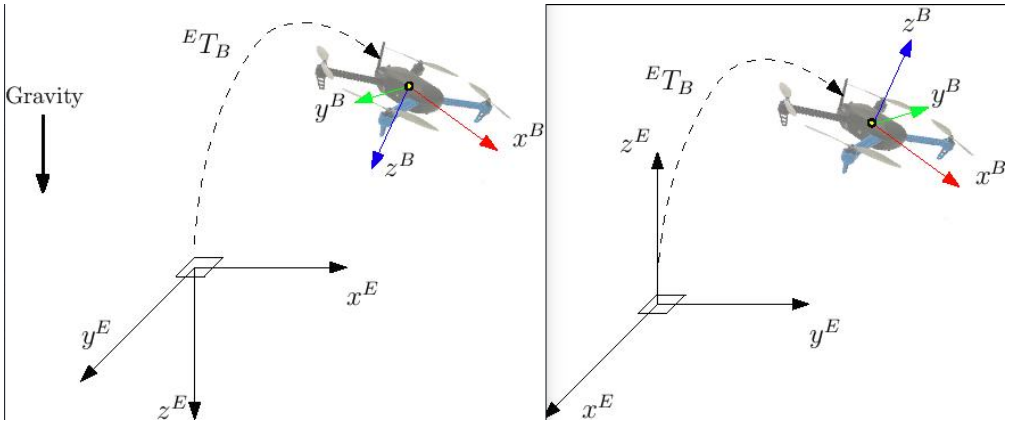

Figure 2.5: NED and FRD frames on the left. ENU and FLU frames on the right[3].

$$R_{NED}^{ENU} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{2.1}$$

The estimated UAV pose is posted on ROS2 topics by the autopilot. In particular, it provides the drone position $\hat{\boldsymbol{p}}_{\boldsymbol{UAV}}^{\boldsymbol{NED}}$ and attitude $\hat{\boldsymbol{R}}_{\boldsymbol{UAV}}^{\boldsymbol{NED}}$.

Chapter 3

Ultrawide-band

3.1 Overview

3.1.1 Definition

As defined in the IEEE 802.15.4a standard and FCC regulations, an Ultrawide-band transmitter is any transmitter that has a fractional bandwidth equal to or greater than 0.20 or a UWB bandwidth equal to or greater than $500MHz$, whatever the fractional bandwidth[4].

The UWB bandwidth is the frequency band bounded by the lower frequency f_L and the upper frequency f_H in which the signal has a power spectral density of $10dB$ below its maximum.

This can be mathematically summarised as:

$$\begin{gathered} Bandwidth >= 500MHz \\ or \\ FractionalBandwidth = 2\frac{f_H - f_L}{f_H + f_L} >= 0.2 \end{gathered} \tag{3.1}$$

The UWB technology exploits a wider frequency band (from 3.1 to $10.6GHz$) at lower power density than other technologies, which use narrow bandwidths and high power densities. This can be noticed in image 3.1.

3.1.2 Advantages

The main reasons why it is decided to adopt UWB technology in this work are range, accuracy, robustness, size and cost. In particular:

- UWB ranging sensors have been proven to work to ranges up to $> 100m$[6].

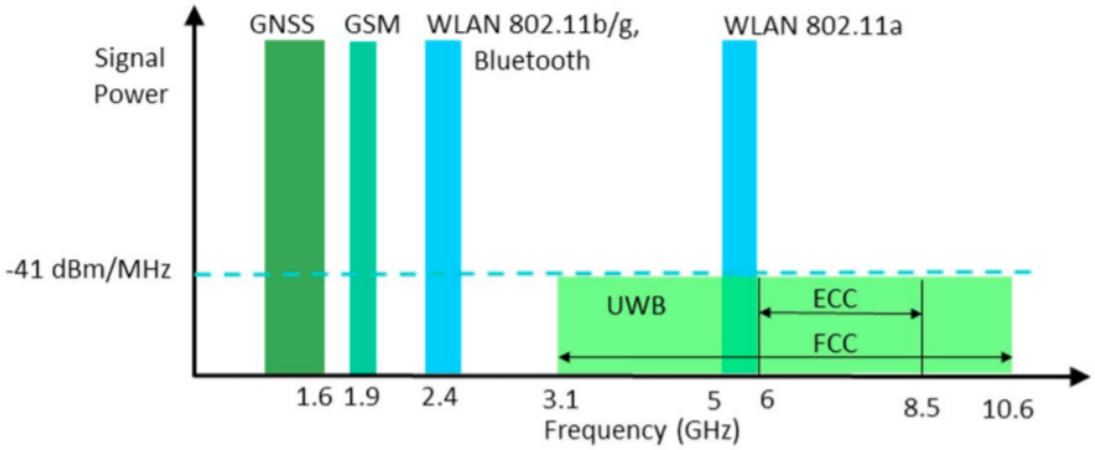

Figure 3.1: Frequency interval of UWB compared to other radio communication protocols [5, Figure 3.2].

- It allows a ranging accuracy around $10cm$[7]. This depends very much on the application (indoor or outdoor), the presence of obstacles and interference, and on the tuning of the ranging antennas used.
- Its wide bandwidth ensures robustness against multi-path propagation, allows obstacle penetration and high data rate transmission (up to 1Gbps).
- The low spectral power increases its resistance agains interference.
- This technology requires space-saving hardware with low power consumption.
- The cost of a single device ranges from around 12 to 200 euros.

UWB offers one of the best solutions when it comes to finding a compromise between accuracy and coverage. This can be seen in the picture 3.2.

3.2 Localization theory

UWB ranging devices can be divided into tags or anchors.

- Anchors are emitters of known position, needed to determine the position of each tag. In this landing system, all four anchors are mounted at the vertices of the UGV.
- Tags are the devices to be tracked, installed on the target. In this specific case the tag is fixed on the UAV underneath.

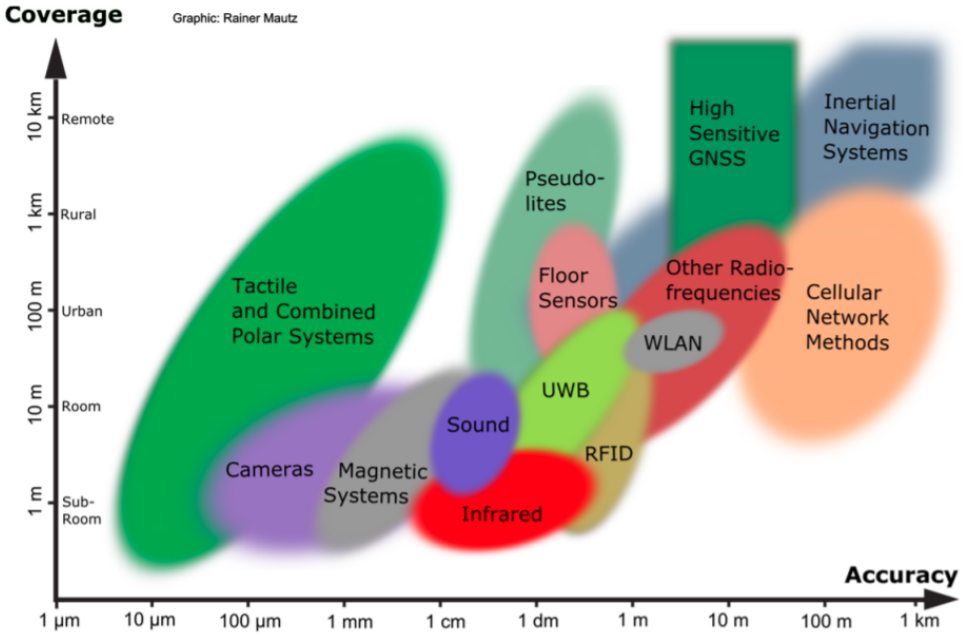

Figure 3.2: List of indoor localization technologies with respect to accuracy and coverage [8, Figure 1.1].

In this work, the proposed device is the Decawave EVB1000 board, which can be easily programmed either as a tag or as an anchor. The detailed specifications can be found in section 7.1.1. The firmware has been set to output the distance of the tag from each anchor, at a given settable frequency.

If some conditions are met, the precise target position can therefore be calculated by means of true range multilateration methods.

Number of anchors *Let d be the number of physical tag dimensions to be determined, and m the number of available anchors, it is required that $m \geq d + 1$*

Thus, at least three anchors are required to obtain the 2D position of the tag, while a minimum of four are needed to achieve 3D positioning.

As the number of anchors increases, the redundancy of the system and therefore the positioning accuracy improves. In general, it is advisable to have more anchors than the minimum required in order to limit NLOS (non-line-of-sight) effects.

Anchors placement *In order to achieve a precise tracking, anchors should never be placed on the same line, in the case of 2D positioning, or on the same plane, in the case of 3D positioning.*

Let's introduce the concept of Geometric Dilution of Precision (GDOP). It is defined as:

$$GDOP = \frac{\Delta(Output\ Location)}{\Delta(Measured\ Data)} \tag{3.2}$$

It expresses how much an error in the calculated distances affects the error in the target position. The higher the GDOP, the lower the reliability of a calculated position.

In particular, *it tends to infinity when placing the anchors on the same line, in the case of 2D positioning, or on the same plane, in the case of 3D positioning*[9].

Another case of interest is when *the UAV is not within the area delimited by the anchors* (e.g. in the chase phase). The greater the distance from it, the higher the GDOP.

This means that in the presence of real measurements (i.e. non-ideal measurements), the resulting positioning error will be extremely high.

In the proposed landing system it is impossible to install the anchors at significantly different heights. For this reason, the UWB technology has only been used to compute the relative 2D position between the UAV and the UGV, as its height estimate z_D would be extremely unreliable. Nevertheless, 4 anchors have been chosen instead of 3 in order to allow positioning even if one device stops working and increase the estimation accuracy.

3.3 Multilateration

Once the coordinates of the anchors and the distances between them and the tag are known, it is possible to estimate the position of the tag. Let us consider the 2D case with 3 anchors for simplicity, as the solution can easily be extended to higher dimensions and larger number of UWB ranging devices.

In the ideal case (fig. 3.3), where the distances are not affected by errors, the solution is given by the intersection of the circles whose centre is the position of the anchors and whose radius is the distance between them and the tag. If more anchors are available, the system is overdetermined and the solution does not change. In the 3D case it is sufficient to intersect the spheres obtained following the same procedure.

Unfortunately, in the real case (fig. 3.4) all measurements are affected by error. This means that the circles will not intersect precisely at one point. A valid solution consists in the minimization of a given cost function that depends on the error between the estimated and true position. In this case, an increase in the number of anchors (i.e. having overdetermined systems) leads to a more precise solution.

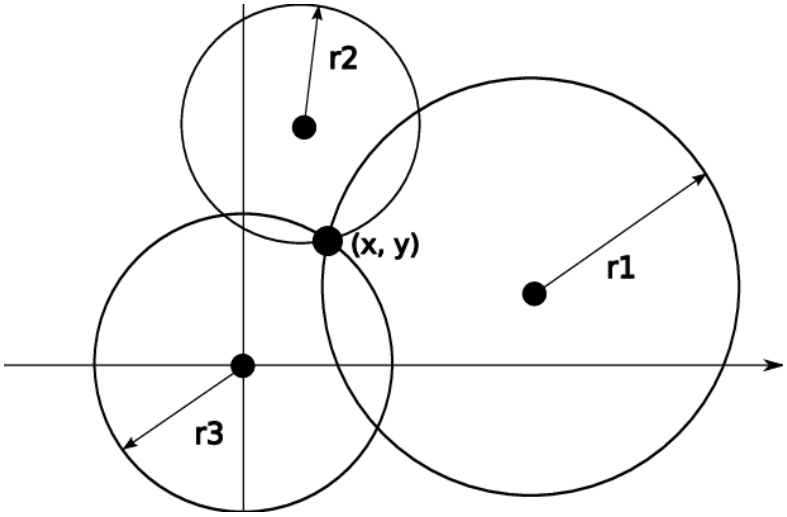

Figure 3.3: Ideal 2D positioning[10].

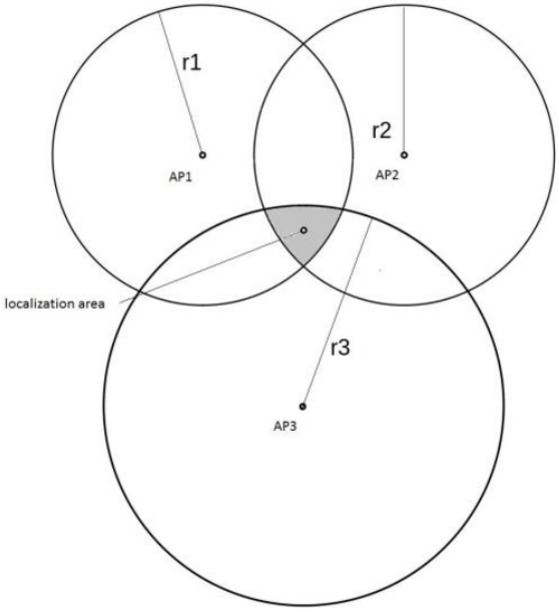

Figure 3.4: Real 2D positioning[11].

Specifically, the multilateration algorithm that has been implemented in our system is the Linear Least Squares (LS).

In the proposed landing system, the UGV is equipped with 4 anchors, while the UAV mounts 1 tag. The mentioned algorithm provides the relative position $\hat{\boldsymbol{p}}_{\boldsymbol{UWB}}^{\boldsymbol{UGV}}$ between the UAV and the UGV in the UGV mobile frame, given the ranges between the anchors and the tag. A simplified representation of the proposed

multilateration system can be seen in image 3.5. A detailed explanation can be found in the next paragraph.

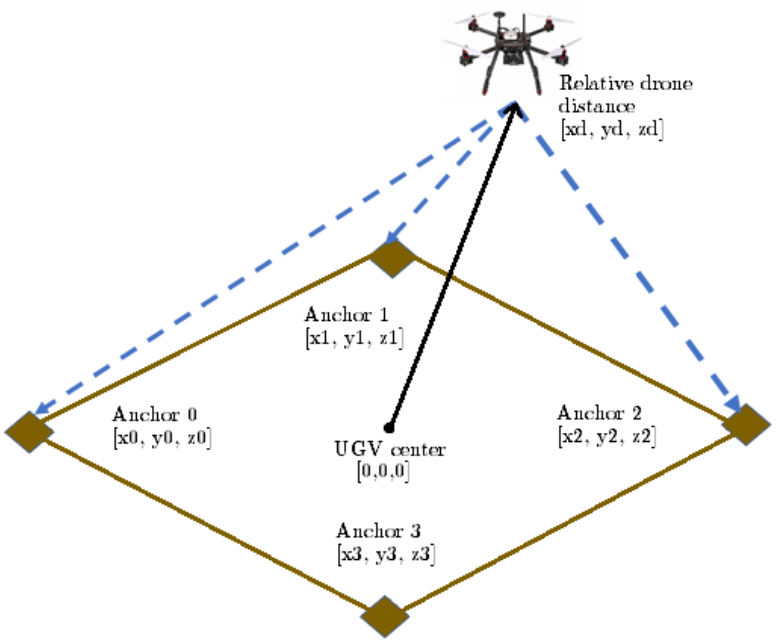

Figure 3.5: Proposed multilateration system.

3.3.1 Linear Least Squares

In this section, the 3D case algorithm is shown and explained. It is trivial obtaining the 2D solution by removing the terms that depend on the z component. The accurate steps necessary to derive the formula can be found in [12]-[13].

Let $(\hat{x}_T, \hat{y}_T, \hat{z}_T)$ *be the estimated coordinates of the tag in the frame* $\mathcal{R}$.
Let (x_i, y_i, z_i) *with* $i = 1..m$ *the known coordinates of the anchors in the frame* $\mathcal{R}$.
Let d_i *with* $i = 1..m$ *the known distance between the anchor* i *and the tag* T.

It follows that:

$$\boldsymbol{A\hat{x}} = \boldsymbol{b} \tag{3.3}$$

Where:

$$\boldsymbol{A} = \begin{bmatrix} 1 & -2x_1 & -2y_1 & -2z_1 \\ 1 & -2x_2 & -2y_2 & -2z_3 \\ 1 & -2x_3 & -2y_3 & -2z_3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & -2x_m & -2y_m & -2z_m \end{bmatrix}, \tag{3.4}$$

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{x}_T^2 + \hat{y}_T^2 + \hat{z}_T^2 \\ \hat{x}_T \\ \hat{y}_T \\ \hat{z}_T \end{bmatrix}, \tag{3.5}$$

$$\boldsymbol{b} = \begin{bmatrix} d_1^2 - x_1^2 - y_1^2 - z_1^2 \\ d_2^2 - x_2^2 - y_2^2 - z_2^2 \\ d_3^2 - x_3^2 - y_3^2 - z_3^2 \\ \vdots \\ d_m^2 - x_m^2 - y_m^2 - z_m^2 \end{bmatrix} \tag{3.6}$$

The solution $\hat{\boldsymbol{x}}$ can be found as:

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{b} \tag{3.7}$$

The estimated tag position $(\hat{x}_T, \hat{y}_T, \hat{z}_T)$ in the frame $\mathcal{R}$ is therefore given by the last 3 components of $\hat{\boldsymbol{x}}$.

Similarly, in the proposed landing system we have:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} (\hat{x}_{UWB}^{UGV})^2 + (\hat{y}_{UWB}^{UGV})^2 + (\hat{z}_{UWB}^{UGV})^2 \\ \hat{x}_{UWB}^{UGV} \\ \hat{y}_{UWB}^{UGV} \\ \hat{z}_{UWB}^{UGV} \end{bmatrix} \tag{3.8}$$

Where $\begin{bmatrix} \hat{x}_{UWB}^{UGV} \\ \hat{y}_{UWB}^{UGV} \end{bmatrix} = \hat{\boldsymbol{p}}_{\boldsymbol{UWB}}^{\boldsymbol{UGV}}$ is the estimated relative position vector between the UAV and the UGV in the UGV frame.

The advantages of this algorithm are:

- its computational lightness. The only costly operation is the inversion of the matrix ($\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$) (eq. 3.7), which depends exclusively on the positions of the anchors (fixed in the considered reference system), and therefore can be performed just once.
- The stability of the algorithm in the presence of outliers and various starting points.

The disadvantage is given by:

- the lower accuracy (with respect to other algorithms) of the obtained estimate, since this method exploites a linearized version of the system.

Chapter 4

System state estimation algorithm

In this section, a linear Kalman Filter approach is proposed as a valid method to estimate the relative position and velocity of the UAV with respect to the UGV in the NED reference frame. Exploiting the system dynamic model and noise information and fusing together the information coming from UWB antennas, PX4 sensors and UGV compass, it is possible to compute an estimate that is better than each of the collected measurements. The complete estimation scheme can be found in figure 4.1, note that the orange blocks are provided by PX4 Autopilot. The proposed Kalman Filter takes as input the multilaterated relative position between the UAV and UGV in the UGV frame $\hat{\boldsymbol{p}}_{\boldsymbol{UWB}}^{\boldsymbol{UGV}}$, the UGV compass heading $\hat{\theta}_{compass}$ and the UAV pose in the NED frame $\hat{\boldsymbol{x}}_{\boldsymbol{UAV}}^{\boldsymbol{NED}}$, and outputs the relative position $\hat{\boldsymbol{p}}_{\boldsymbol{rel}}^{\boldsymbol{NED}}$ and velocity $\hat{\boldsymbol{v}}_{\boldsymbol{rel}}^{\boldsymbol{NED}}$ estimates between the drone and the rover expressed in the NED frame.

The Least Squares algorithm shown in 3.3 allows the computation of the relative position of the drone with respect to the rover in its mobile reference frame. Hence, it needs to be rotated in the NED reference frame for control purposes. For this reason, the UGV is equipped with a compass that outputs the orientation of the drone with respect to the NED reference frame. Unfortunately, the noise affecting both orientation and position makes the directly rotated vector, and hence its derivative, useless. The Kalman filter solves these problems: position and angle are filtered separately, in a loosely coupled fashion. Each time the multilaterated position is computed, it is firstly rotated in the NED frame using the smoothed angle, and subsequently used to update the position estimate. A detailed explanation of this algorithm can be found in this chapter.

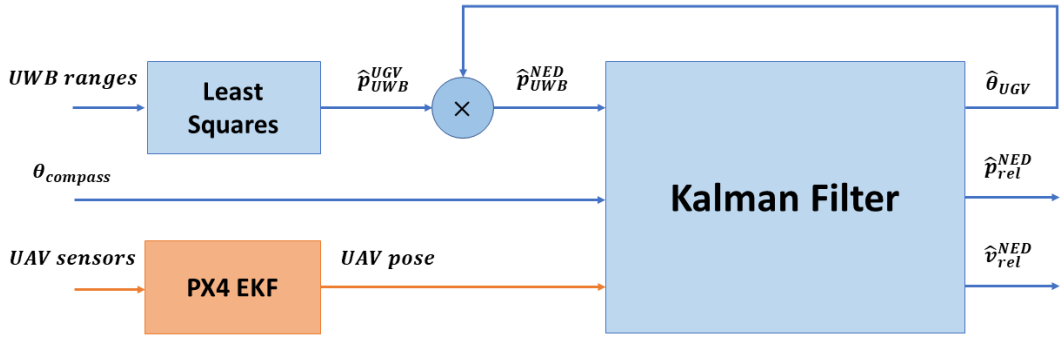

Figure 4.1: Proposed system state estimation algorithm.

4.1 Kalman Filter theory

Let's describe our system with the following state space model[14]:

$$\begin{cases} \boldsymbol{x_k} = \boldsymbol{F_k x_{k-1}} + \boldsymbol{B_k u_k} + \boldsymbol{w_k} \\ \boldsymbol{z_k} = \boldsymbol{H_k x_k} + \boldsymbol{v_k} \end{cases} \tag{4.1}$$

Then:

- $\boldsymbol{x_k}$ is the system state.
- $\boldsymbol{F_k}$ is the state transition model.
- $\boldsymbol{B_k}$ is the control-input model.
- $\boldsymbol{w_k}$ is the process noise.
- $\boldsymbol{z_k}$ is the observation.
- $\boldsymbol{H_k}$ is the observation model.
- $\boldsymbol{v_k}$ is the observation noise.

Let's assume that $\boldsymbol{w_k}$ belongs to a normal distribution $\mathcal{N}$ with covariance $\boldsymbol{Q_k}$ and $\boldsymbol{v_k}$ to be a Gaussian white noise with covariance $\boldsymbol{R_k}$.
This can be expressed in the following way:

$$\boldsymbol{Q_k} : \boldsymbol{w_k} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q_k}) \tag{4.2}$$

$$\boldsymbol{R_k} : \boldsymbol{v_k} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R_k}) \tag{4.3}$$

The sensor fusion algorithm is composed of two steps, which can occur successively or asynchronously. This depends on when the sensor data are available.

Prediction In the prediction phase, the system model, the inputs $\boldsymbol{u_k}$ and the estimated states $\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}}$ are used to predict the value of the states at the next instant $\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}$. Usually this step is performed at the fastest sensor rate, taken as input, or at a fixed rate, hereafter referred to as dt.

Update In the update phase the states $\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}$ are corrected thanks to the observation $\boldsymbol{z_k}$. In this way, a more accurate estimate of $\boldsymbol{x_k}$ can be computed. Moreover, the number of update phases is the same as the number of asynchronous sensors, each one with its specific observation model.

4.1.1 Prediction

Firstly, the predicted a priori state estimate $\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}$ and a priori estimate covariance $\boldsymbol{P}_{\boldsymbol{k|k-1}}$ are computed.

$$\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} = \boldsymbol{F_k}\hat{\boldsymbol{x}}_{\boldsymbol{k-1|k-1}} + \boldsymbol{B_k}\boldsymbol{u_k} \tag{4.4}$$

$$\boldsymbol{P}_{\boldsymbol{k|k-1}} = \boldsymbol{F_k}\boldsymbol{P}_{\boldsymbol{k-1|k-1}}\boldsymbol{F_k^T} + \boldsymbol{Q_k} \tag{4.5}$$

Here, the starting estimate covariance $\boldsymbol{P_0}$ and the state covariance matrix $\boldsymbol{Q_k}$ must be appropriately chosen. This is usually done through a trial and error procedure or thanks to a deep knowledge about the real system.

4.1.2 Update

Lastly, the updated state estimate $\hat{\boldsymbol{x}}_{k|k}$ and updated covariance $\boldsymbol{P}_{k|k}$ are computed exploiting the sensor data $\boldsymbol{z_k}$.

$$\tilde{\boldsymbol{y}}_{\boldsymbol{k}} = \boldsymbol{z_k} - \boldsymbol{H_k}\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} \tag{4.6}$$

$$\boldsymbol{S_k} = \boldsymbol{H_k}\boldsymbol{P}_{\boldsymbol{k|k-1}}\boldsymbol{H_k^T} + \boldsymbol{R_k} \tag{4.7}$$

$$\boldsymbol{K_k} = \boldsymbol{P}_{\boldsymbol{k|k-1}}\boldsymbol{H_k^T}\boldsymbol{S_k^{-1}} \tag{4.8}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k|k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} + \boldsymbol{K_k}\tilde{\boldsymbol{y}}_{\boldsymbol{k}} \tag{4.9}$$

$$\boldsymbol{P}_{\boldsymbol{k|k}} = \left(\boldsymbol{I} - \boldsymbol{K_k}\boldsymbol{H_k}\right)\boldsymbol{P}_{\boldsymbol{k|k-1}} \tag{4.10}$$

Here, the observation covariance matrix $\boldsymbol{R_k}$ must be appropriately chosen.

4.2 System model

Given the dynamics of the system, the UAV can be described by a constant acceleration model, while the UGV by a constant velocity one. Since the landing platform moves mainly in a straight line, the yaw speed can be approximated to zero. These measurements are asynchronously published on different ROS2 topics. Therefore, it is necessary to model several observation models. The prediction phase is executed each timestep dt, while each update phase is executed whenever the correspoding data are available.

4.2.1 Prediction model

Based on the assumptions listed above, the UAV and the UGV can be modeled by the following sets of differential equations:

$$\begin{cases} \frac{d}{dt}x_{UAV}(t) = \dot{x}_{UAV}(t) \\ \frac{d}{dt}\dot{x}_{UAV}(t) = \ddot{x}_{UAV}(t) \\ \frac{d}{dt}\ddot{x}_{UAV}(t) = 0 \\ \frac{d}{dt}y_{UAV}(t) = \dot{y}_{UAV}(t) \\ \frac{d}{dt}\dot{y}_{UAV}(t) = \ddot{y}_{UAV}(t) \\ \frac{d}{dt}\ddot{y}_{UAV}(t) = 0 \end{cases} \tag{4.11}$$

$$\begin{cases} \frac{d}{dt}x_{UGV}(t) = \dot{x}_{UGV}(t) \\ \frac{d}{dt}\dot{x}_{UGV}(t) = 0 \\ \frac{d}{dt}y_{UGV}(t) = \dot{y}_{UGV}(t) \\ \frac{d}{dt}\dot{y}_{UGV}(t) = 0 \\ \frac{d}{dt}\theta_{UGV}(t) = 0 \end{cases} \tag{4.12}$$

They can be discretized through the Euler method, as follows:

$$\begin{cases} x_{UAV,k} = x_{UAV,k-1} + \dot{x}_{UAV,k-1}dt + \frac{1}{2}\ddot{x}_{UAV,k-1}dt^2 \\ \dot{x}_{UAV,k} = \dot{x}_{UAV,k-1} + \ddot{x}_{UAV,k-1}dt \\ \ddot{x}_{UAV,k} = \ddot{x}_{UAV,k-1} \\ y_{UAV,k} = y_{UAV,k-1} + \dot{y}_{UAV,k-1}dt + \frac{1}{2}\ddot{y}_{UAV,k-1}dt^2 \\ \dot{y}_{UAV,k} = \dot{y}_{UAV,k-1} + \ddot{y}_{UAV,k-1}dt \\ \ddot{y}_{UAV,k} = \ddot{y}_{UAV,k-1} \\ x_{UGV,k} = x_{UGV,k-1} + \dot{x}_{UGV,k-1}dt \\ \dot{x}_{UGV,k} = \dot{x}_{UGV,k-1} \\ y_{UGV,k} = y_{UGV,k-1} + \dot{y}_{UGV,k-1}dt \\ \dot{y}_{UGV,k} = \dot{y}_{UGV,k-1} \\ \theta_{UGV,k} = \theta_{UGV,k-1} \end{cases} \tag{4.13}$$

The constant acceleration state transition model can be written as:

$$\boldsymbol{F_1} = \begin{bmatrix} 1 & dt & \frac{1}{2}dt^2 \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

On the other hand, the constant velocity one is given by:

$$\boldsymbol{F_2} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{4.15}$$

Therefore, our system can be written in matrix form as:

$$\begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ \theta_{UGV,k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F_1} & \mathbf{0}_{3x3} & \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{0}_{3x1} \\ \mathbf{0}_{3x3} & \boldsymbol{F_1} & \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{0}_{3x1} \\ \mathbf{0}_{2x3} & \mathbf{0}_{2x3} & \boldsymbol{F_2} & \mathbf{0}_{2x2} & \mathbf{0}_{2x1} \\ \mathbf{0}_{2x3} & \mathbf{0}_{2x3} & \mathbf{0}_{2x2} & \boldsymbol{F_2} & \mathbf{0}_{2x1} \\ \mathbf{0}_{1x3} & \mathbf{0}_{1x3} & \mathbf{0}_{1x2} & \mathbf{0}_{1x2} & 1 \end{bmatrix} \begin{bmatrix} x_{UAV,k-1} \\ \dot{x}_{UAV,k-1} \\ \ddot{x}_{UAV,k-1} \\ y_{UAV,k-1} \\ \dot{y}_{UAV,k-1} \\ \ddot{y}_{UAV,k-1} \\ x_{UGV,k-1} \\ \dot{x}_{UGV,k-1} \\ y_{UGV,k-1} \\ \dot{y}_{UGV,k-1} \\ \theta_{UGV,k-1} \end{bmatrix} \tag{4.16}$$

Let's choose the UGV and UAV pose in the **NED** frame as the system states.

$$\boldsymbol{x} = \begin{bmatrix} x_{UAV} \\ \dot{x}_{UAV} \\ \ddot{x}_{UAV} \\ y_{UAV} \\ \dot{y}_{UAV} \\ \ddot{y}_{UAV} \\ x_{UGV} \\ \dot{x}_{UGV} \\ y_{UGV} \\ \dot{y}_{UGV} \\ \theta_{UGV} \end{bmatrix} \tag{4.17}$$

Therefore, the state can be described by the following model:

$$\boldsymbol{x_k} = \boldsymbol{F_k}\boldsymbol{x_{k-1}} + \boldsymbol{w_k} \tag{4.18}$$

Where the state transition model is equal to:

$$\boldsymbol{F_k} = \begin{bmatrix} \boldsymbol{F_1} & \mathbf{0}_{3x3} & \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{0}_{3x1} \\ \mathbf{0}_{3x3} & \boldsymbol{F_1} & \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{0}_{3x1} \\ \mathbf{0}_{2x3} & \mathbf{0}_{2x3} & \boldsymbol{F_2} & \mathbf{0}_{2x2} & \mathbf{0}_{2x1} \\ \mathbf{0}_{2x3} & \mathbf{0}_{2x3} & \mathbf{0}_{2x2} & \boldsymbol{F_2} & \mathbf{0}_{2x1} \\ \mathbf{0}_{1x3} & \mathbf{0}_{1x3} & \mathbf{0}_{1x2} & \mathbf{0}_{1x2} & 1 \end{bmatrix} \tag{4.19}$$

And the control-input model is:

$$\boldsymbol{B_k} = \mathbf{0} \tag{4.20}$$

Adopting a discrete constant white noise model[15], we have:

$$\boldsymbol{Q_1} = q_{UAV} \begin{bmatrix} \frac{1}{4}dt^4 & \frac{1}{2}dt^3 & \frac{1}{2}dt^2 \\ \frac{1}{2}dt^3 & dt^2 & dt \\ \frac{1}{2}dt^2 & dt & 1 \end{bmatrix} \tag{4.21}$$

$$\boldsymbol{Q_2} = q_{UGV} \begin{bmatrix} \frac{1}{4}dt^4 & \frac{1}{2}dt^3 \\ \frac{1}{2}dt^3 & dt^2 \end{bmatrix} \tag{4.22}$$

$$Q_3 = q_\theta \tag{4.23}$$

The model covariance matrix can therefore be written as:

$$\boldsymbol{Q_k} = \begin{bmatrix} \boldsymbol{Q_1} & \mathbf{0}_{3x3} & \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{0}_{3x1} \\ \mathbf{0}_{3x3} & \boldsymbol{Q_1} & \mathbf{0}_{3x2} & \mathbf{0}_{3x2} & \mathbf{0}_{3x1} \\ \mathbf{0}_{2x3} & \mathbf{0}_{2x3} & \boldsymbol{Q_2} & \mathbf{0}_{2x2} & \mathbf{0}_{2x1} \\ \mathbf{0}_{2x3} & \mathbf{0}_{2x3} & \mathbf{0}_{2x2} & \boldsymbol{Q_2} & \mathbf{0}_{2x1} \\ \mathbf{0}_{1x3} & \mathbf{0}_{1x3} & \mathbf{0}_{1x2} & \mathbf{0}_{1x2} & Q_3 \end{bmatrix} \tag{4.24}$$

Whereas $\boldsymbol{P_0}$ is modeled as follows:

$$\boldsymbol{P_0} = p_0 \boldsymbol{I}_{11x11} \tag{4.25}$$

q_i represents how much one trusts the accuracy of the respective model, while p_0 how much reliable the initial state $\boldsymbol{x_0}$ is. Therefore, increasing p_k leads to giving less weight to the mathematical model and more importance to the measurements $\boldsymbol{z_k}$. Increasing p_k results in higher uncertainty about the initial state $\boldsymbol{x_0}$. These parameters are chosen through a trial and error approach.

4.2.2 Observation models

As explained before, each observation model can be written as:

$$\boldsymbol{z_k} = \boldsymbol{H_k x_k} + \boldsymbol{v_k} \tag{4.26}$$

We will have 3 different observation models, one for each sensor.

- **Compass observation model**

  The compass mounted on the UGV directly returns the angle of the rover with respect to the North.

$$z_{compass,k} = \theta_{UGV,k} + v_{UGV,k} \tag{4.27}$$

$$z_{compass,k} = \begin{bmatrix} \mathbf{0_{1x10}} & 1 \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ \theta_{UGV,k} \end{bmatrix} + v_{UGV,k} \tag{4.28}$$

  Hence:

$$\boldsymbol{H_{compass,k}} = \begin{bmatrix} \mathbf{0_{1x10}} & 1 \end{bmatrix} \tag{4.29}$$

- **PX4 sensors observation model**

  The PX4 Autopilot provides the entire UAV pose on a ROS2 topic.

$$\boldsymbol{z_{PX4,k}} = \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \end{bmatrix} + \boldsymbol{v_{PX4,k}} \tag{4.30}$$

$$\boldsymbol{z_{PX4,k}} = \begin{bmatrix} \boldsymbol{I_{6x6}} & \boldsymbol{0_{6x5}} \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ \theta_{UGV,k} \end{bmatrix} + \boldsymbol{v_{PX4,k}} \tag{4.31}$$

  Therefore:

$$\boldsymbol{H_{UAV,k}} = \begin{bmatrix} \boldsymbol{I_{6x6}} & \boldsymbol{0_{6x5}} \end{bmatrix} \tag{4.32}$$

- **UWB position observation model**

  The multilaterated position $\hat{\boldsymbol{p}}^{\boldsymbol{UGV}}_{\boldsymbol{UWB,k}}$ is in the rover moving reference frame $\mathcal{R}^{UGV}$, therefore it needs to be rotated in the NED one before it can be used to update the system state.

$$\hat{\boldsymbol{p}}^{\boldsymbol{UGV}}_{\boldsymbol{UWB,k}} = \begin{bmatrix} \Delta\hat{x}^{UGV}_{UWB,k} \\ \\ \Delta\hat{y}^{UGV}_{UWB,k} \end{bmatrix} \tag{4.33}$$

  This position can be rotated by left-multiplying the rotation matrix $\hat{\boldsymbol{R}}^{\boldsymbol{NED}}_{\boldsymbol{UGV}}$.

$$\hat{\boldsymbol{p}}^{\boldsymbol{NED}}_{\boldsymbol{UWB,k}} = \hat{\boldsymbol{R}}^{\boldsymbol{NED}}_{\boldsymbol{UGV}} \hat{\boldsymbol{p}}^{\boldsymbol{UGV}}_{\boldsymbol{UWB,k}} = \boldsymbol{p}^{\boldsymbol{NED}}_{\boldsymbol{UWB,k}} + \boldsymbol{v_{UWB,k}} \tag{4.34}$$

$$\hat{\boldsymbol{p}}^{\boldsymbol{NED}}_{\boldsymbol{UWB,k}} = \begin{bmatrix} x_{UAV,k} - x_{UGV,k} \\ y_{UAV,k} - y_{UGV,k} \end{bmatrix} + \boldsymbol{v_{UWB,k}} \tag{4.35}$$

Where $\hat{\boldsymbol{R}}_{\boldsymbol{UGV}}^{\boldsymbol{NED}}$ is directly constructed using the angle estimate $\hat{\theta}_{UGV,k}$, returned by the filter at each instant k.

$$\hat{\boldsymbol{R}}_{\boldsymbol{UGV}}^{\boldsymbol{NED}} = \begin{bmatrix} \cos\left(\hat{\theta}_{UGV,k}\right) & -\sin\left(\hat{\theta}_{UGV,k}\right) & 0 \\ \sin\left(\hat{\theta}_{UGV,k}\right) & \cos\left(\hat{\theta}_{UGV,k}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.36}$$

$$\boldsymbol{z}_{\boldsymbol{UWB,k}}^{\boldsymbol{NED}} = \boldsymbol{p}_{\boldsymbol{UWB,k}}^{\boldsymbol{NED}} + \boldsymbol{v}_{\boldsymbol{UWB,k}} \tag{4.37}$$

$$\boldsymbol{z}_{\boldsymbol{UWB,k}}^{\boldsymbol{NED}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ \theta_{UGV,k} \end{bmatrix} + \boldsymbol{v}_{\boldsymbol{UWB,k}} \tag{4.38}$$

Hence:

$$\boldsymbol{H}_{\boldsymbol{UWB,k}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{4.39}$$

As a final step, the covariance matrices $\boldsymbol{R}_{\boldsymbol{PX4,k}}$, $\boldsymbol{R}_{\boldsymbol{UWB,k}}$ and $R_{compass,k}$ should be suitably chosen based on the accuracy of each sensor. Increasing R leads to less trust in the considered measure, while decreasing it results in giving it more weight.

4.2.3 Outputs

The outputs returned by the Kalman Filter are composed of the UAV filtered position, velocity and acceleration, the filtered UGV yaw angle and the indirect

estimate of the UGV position and velocity.

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{x}_{UAV} \\ \hat{\dot{x}}_{UAV} \\ \hat{\ddot{x}}_{UAV} \\ \hat{y}_{UAV} \\ \hat{\dot{y}}_{UAV} \\ \hat{\ddot{y}}_{UAV} \\ \hat{x}_{UGV} \\ \hat{\dot{x}}_{UGV} \\ \hat{y}_{UGV} \\ \hat{\dot{y}}_{UGV} \\ \hat{\theta}_{UGV} \end{bmatrix} \tag{4.40}$$

Then, the relative position estimate between the UAV and the UGV in the NED frame is equal to:

$$\hat{\boldsymbol{p}}_{rel,k} = \begin{bmatrix} \hat{x}_{UGV,k} - \hat{x}_{UAV,k} \\ \hat{y}_{UGV,k} - \hat{y}_{UAV,k} \end{bmatrix} \tag{4.41}$$

And the relative velocity is given by:

$$\hat{\boldsymbol{v}}_{rel,k} = \begin{bmatrix} \hat{\dot{x}}_{UGV,k} - \hat{\dot{x}}_{UAV,k} \\ \hat{\dot{y}}_{UGV,k} - \hat{\dot{y}}_{UAV,k} \end{bmatrix} \tag{4.42}$$

Finally, it is possible to indirectly estimate the velocity of the UGV in the NED frame. This is equal to:

$$\hat{\boldsymbol{v}}_{UGV,k} = \begin{bmatrix} \hat{\dot{x}}_{UGV,k} \\ \hat{\dot{y}}_{UGV,k} \end{bmatrix} \tag{4.43}$$

These three vectors are used by the proposed control algorithm to make the UAV autonomously land on the UGV.

Chapter 5

Autonomous takeoff and landing algorithm

5.1 System state machine

The proposed landing system can be divided into 4 main phases, as shown in figure 5.1. Specifically, it is composed of: Takeoff, Chase, Descent and Landing. These steps are not consequential and may alternate, depending on state variable values.

During the takeoff phase the drone is firstly armed, then it climbs to an altitude called $HovHeight$. If a valid relative position $\hat{p}_{rel}$ and velocity $\hat{v}_{rel}$ estimate of the UGV with respect to the UAV is available, then it is possible to switch to the chase step. Here, a gain scheduling PID speed control algorithm takes care of bringing the UAV close to the target, while keeping its height constant. The descent can begin only if the relative position $\|\hat{\boldsymbol{p}}_{\boldsymbol{rel}}\|_2$ is less or equal than a certain distance $DescDist(\hat{z}_{rel})$ and if the relative velocity $\|\hat{\boldsymbol{v}}_{\boldsymbol{rel}}\|_2$ is less or equal than a certain velocity $DescVel$. If one of the two conditions is not met, the descent is aborted and the drone returns to the chase mode. Finally, if the relative height $\hat{z}_{rel}$ of the drone with respect to the rover is less than a certain distance $LandHeight$, the UAV attempts a landing by turning off the motors.

5.1.1 Takeoff phase

During this phase the drone is controlled in position. As explained in the paragraph 2.3, it is possible to give the UAV position setpoints via offboard control. These are managed by the autopilot control algorithm shown in figure 2.3. By giving the

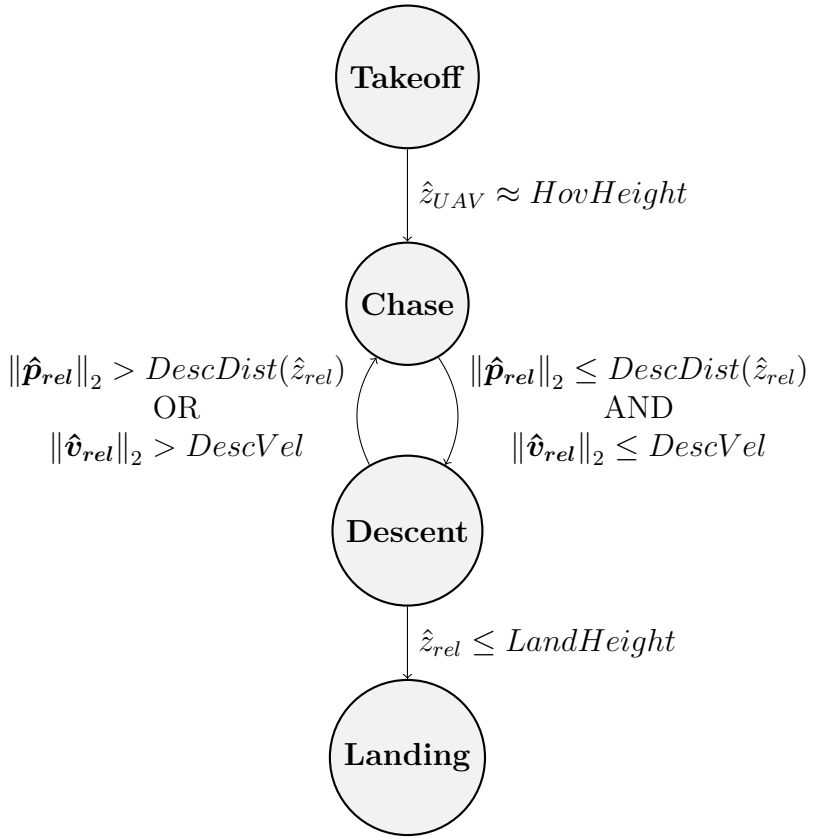

Figure 5.1: Landing system state machine.

takeoff command, the UAV receives a position setpoint equal to:

$$\begin{cases} x_{UAV,k} = 0 \\ y_{UAV,k} = 0 \\ z_{UAV,k} = HovHeight \end{cases} \tag{5.1}$$

Where the drone height estimate $\hat{z}_{UAV}$ is returned by PX4 Autopilot. As soon as the UAV reaches the desired height, it switches to the chase step.

5.1.2 Chase phase

During this phase a gaining scheduling PID speed control algorithm controls $\dot{x}_{UAV}$ and $\dot{y}_{UAV}$, while a position control algorithm keeps z_{UAV} constant and equal to $HovHeight$. The control algorithm switches from a proportional control to a proportional-integral-derivative one depending on the relative position of the UAV and UGV, in a gain scheduling fashion. At high distance, when $\|\boldsymbol{p}_{\boldsymbol{rel}}\|_2 >$

$SwitchContrDist$, the control is purely proportional, whereas at close distance, when $\|\boldsymbol{p_{rel}}\|_2 \leq SwitchContrDist$, the control algorithm follows a proportional-integral-derivative law. Therefore, the setpoint given to the PX4 Autopilot in this step is equal to:

$$\begin{cases} \dot{x}_{UAV,k} = P \cdot \hat{x}_{rel,k} & \text{if } \|\boldsymbol{p_{rel}}\|_2 > SwitchContrDist \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{rel,k} & \text{if } \|\boldsymbol{p_{rel}}\|_2 > SwitchContrDist \\ \dot{x}_{UAV,k} = P \cdot \hat{x}_{rel,k} + I \cdot \sum_0^k \hat{x}_{rel,k} dt + D \cdot \hat{\dot{x}}_{rel,k} & \text{if } \|\boldsymbol{p_{rel}}\|_2 \leq SwitchContrDist \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{rel,k} + I \cdot \sum_0^k \hat{y}_{rel,k} dt + D \cdot \hat{\dot{y}}_{rel,k} & \text{if } \|\boldsymbol{p_{rel}}\|_2 \leq SwitchContrDist \\ z_{UAV} = HovHeight \end{cases} \tag{5.2}$$

Coupled with an anti-windup scheme, this control algorithm ensures precise tracking and fast convergence with a really small overshoot. A detailed explanation can be found in paragraph 5.2.1. If the UAV is close enough to the platform and the relative velocity estimate is fairly small, the drone switches to the descent phase.

5.1.3 Descent phase

In order to switch to this state, it is required that $\|\hat{\boldsymbol{p}}_{\boldsymbol{rel}}\|_2 \leq DescDist(\hat{z}_{rel})$ and $\|\hat{\boldsymbol{v}}_{\boldsymbol{rel}}\|_2 \leq DescVel$. This to ensure that the drone starts descending only when close to the target, and avoid descents in the presence of overshoots or sudden stops of the rover, which could lead to crashes or instability. It is noticeable that $DescDist(\hat{z}_{rel})$ depends on $\hat{z}_{rel}$, indeed the allowed area of descent is given by a cone, as can be seen in image 5.2. After several simulations and real flights it has been noticed that the UAV struggled to start its descent at high altitudes, while it had no problems when closer to the target. This is due to a higher variance in the relative position estimate due to the intrinsic geometry of the problem (i.e anchors and tag). A cone-like descent constraint area is sufficient to solve this problem. The constraint can be mathematically expressed as:

$$\|\hat{\boldsymbol{p}}_{\boldsymbol{rel}}\|_2 \leq DescDist(\hat{z}_{rel}) \tag{5.3}$$

With:

$$\begin{cases} DescDist = DescDistCyl + ConeSlope \cdot (\hat{z}_{rel} - CylHeight) & \text{if } \hat{z}_{rel} > CylHeight \\ DescDist = DescDistCyl & \text{if } \hat{z}_{rel} \leq CylHeight \end{cases} \tag{5.4}$$

Where:

$$\hat{z}_{rel} = \hat{z}_{UAV} - \hat{z}_{UGV} \tag{5.5}$$

Note that z_{UGV} can be approximated to a constant value, equal to the rover height. If this constraint is not met, the descent is aborted and the drone returns to the chase step.

The setpoint given to the UAV is equal to:

$$\begin{cases} \dot{x}_{UAV,k} = P \cdot \hat{x}_{rel,k} + I \cdot \sum_0^k \hat{x}_{rel,k} dt + D \cdot \hat{\dot{x}}_{rel,k} \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{rel,k} + I \cdot \sum_0^k \hat{y}_{rel,k} dt + D \cdot \hat{\dot{y}}_{rel,k} \\ \dot{z}_{UAV} = DescVel \end{cases} \tag{5.6}$$

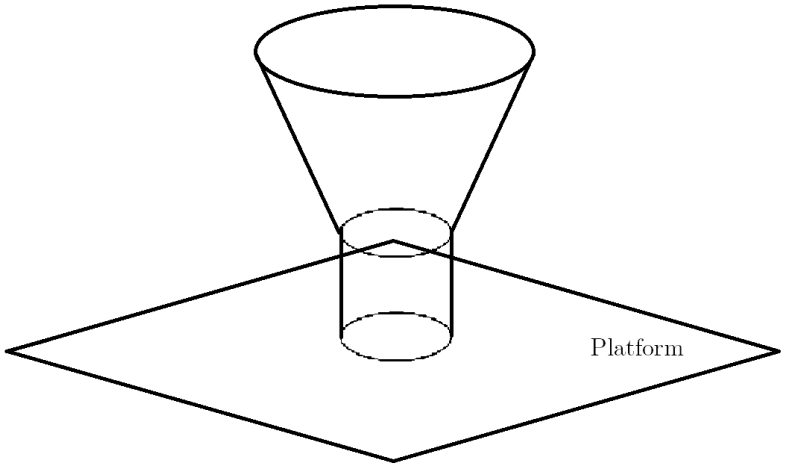

Figure 5.2: Descent constraint area.

5.1.4 Landing phase

As soon as the relative height between the UAV and the UGV $\hat{z}_{rel}$ decreases under a certain value $LandHeight$, then the landing command is sent to PX4 Autopilot, making the drone turn off its motors. If the landing pad moves too fast (i.e more than 3km/h), it is possible that the drone misses the platfrom during the free fall. In order to solve this problem a predictive control variant is proposed, as can be seen in subsection 5.2.2.

5.2 Proposed control architecture

Given $\hat{\boldsymbol{p}}_{\boldsymbol{rel}}^{\boldsymbol{NED}}$ and $\hat{\boldsymbol{v}}_{\boldsymbol{rel}}^{\boldsymbol{NED}}$, the proposed tracking control algorithm computes, at a dt rate, the velocity setpoint to be passed to the autopilot. The complete scheme can be seen in image 5.3. Note that the orange blocks are already provided by PX4 Autopilot and shown in pictures 2.2-2.4.

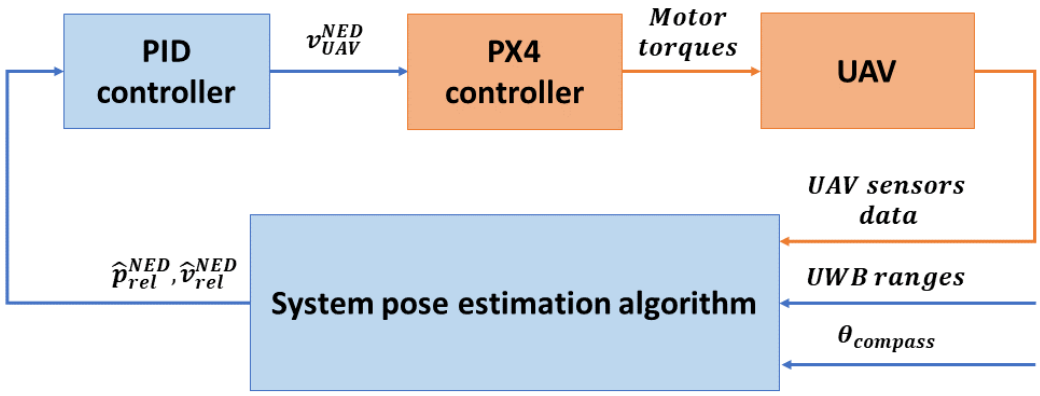

Figure 5.3: Proposed system control algorithm.

5.2.1 Control algorithm

The control algorithm switches from a proportional to a proportional-integral-derivative one, as can be seen in equation 5.2 and in state machine 5.4.

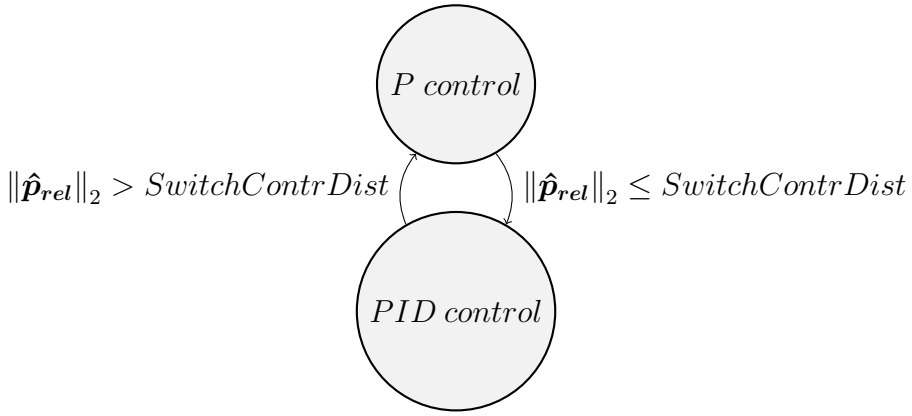

Figure 5.4: Gain scheduling PID logic.

The proportional control is given by:

$$\begin{cases} \dot{x}_{UAV,k} = P \cdot \hat{x}_{rel,k} \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{rel,k} \end{cases} \tag{5.7}$$

While the PID one is given by:

$$\begin{cases} \dot{x}_{UAV,k} = P \cdot \hat{x}_{rel,k} + I \cdot S(\hat{x}_{rel,k}) + D \cdot \hat{\dot{x}}_{rel,k} \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{rel,k} + I \cdot S(\hat{y}_{rel,k}) + D \cdot \hat{\dot{y}}_{rel,k} \end{cases} \tag{5.8}$$

Where $S(\hat{x}_{rel,k})$, $S(\hat{y}_{rel,k})$ are the relative position discrete integrals over all time instants k.

$$S(\hat{x}_{rel,k}) = \sum_{0}^{k} \hat{x}_{rel,k}dt = \sum_{0}^{k-1} \hat{x}_{rel,k}dt + \hat{x}_{rel,k}dt = S(\hat{x}_{rel,k-1}) + \hat{x}_{rel,k}dt$$
$$S(\hat{y}_{rel,k}) = \sum_{0}^{k} \hat{y}_{rel,k}dt = \sum_{0}^{k-1} \hat{y}_{rel,k}dt + \hat{y}_{rel,k}dt = S(\hat{y}_{rel,k-1}) + \hat{y}_{rel,k}dt$$

The smooth transition between the proportional control and the PID one is ensured by the continuity of the function in the derivative, integral and proportional values. Indeed, the proportional gain is the same in both control schemes, while the integral and derivative ones start from 0 when the switch occurs. Note that the integral contributions $S(\hat{x}_{rel,k})$, $S(\hat{y}_{rel,k})$ are reset everytime the control scheme changes, i.e. from proportional to PID and viceversa.

In order to prevent high overshoots an anti-windup scheme is implemented. The proposed solution consists in stopping the integration whenever the UAV speed saturates to its maximum value. This can be mathematically expressed as:

$$\begin{cases} S(\hat{x}_{rel,k}) = S(\hat{x}_{rel,k-1}) & \text{if } \left|\hat{\dot{x}}_{UAV}\right| > 0.8|\dot{x}_{UAV,max}|,\ S(\hat{x}_{rel,k-1}) \cdot \hat{x}_{rel,k}dt > 0 \\ S(\hat{x}_{rel,k}) = S(\hat{x}_{rel,k-1}) + \hat{x}_{rel,k}dt & \text{if } \left|\hat{\dot{x}}_{UAV}\right| \leq 0.8|\dot{x}_{UAV,max}| \end{cases} \tag{5.9}$$

Note that the condition $S(\hat{x}_{rel,k-1}) \cdot \hat{x}_{rel,k}dt > 0$ makes sure that integration is denied only when the integral contribution and the new value have the same sign. In this way, the integral contribution cannot saturate but only decrease.

5.2.2 Predictive control algorithm

One of the problems of landing at high speeds (i.e. more than 3km/h) is the possibility of the UAV falling off the platform. Indeed, during the landing phase the drone shuts down the motors and falls vertically, often causing the UAV to miss the rover.

The proposed system state estimation algorithm shown in chapter 4 returns an indirect estimate of the UGV velocity. This information can be used to predict, at a dt rate, the position of the rover over the following time instants. Hence, the above problem is solved by making the UAV chase the position given by the one of the UGV center **plus** the predicted displacement of the rover over the drone fall time. This control scheme can be mathematically expressed as:

$$\begin{cases} \hat{x}_{pred,k} = \hat{x}_{rel,k} + \dfrac{LandHeight}{LandVel} \cdot \hat{\dot{x}}_{UAV,k} \\ \\ \hat{y}_{pred,k} = \hat{y}_{rel,k} + \dfrac{LandHeight}{LandVel} \cdot \hat{\dot{y}}_{UAV,k} \end{cases} \tag{5.10}$$

Note that it corresponds to shifting the descent constraint cone along the moving direction of the landing pad, as shown in figure 5.5.

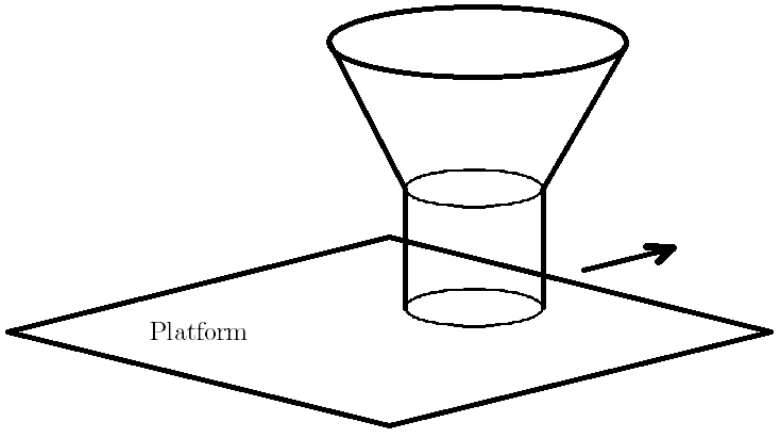

Figure 5.5: Predictive descent constraint area.

Now, the proportional control is given by:

$$\begin{cases} \dot{x}_{UAV,k} = P \cdot \hat{x}_{pred,k} \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{pred,k} \end{cases} \tag{5.11}$$

While the PID one is given by:

$$\begin{cases} \dot{x}_{UAV,k} = P \cdot \hat{x}_{pred,k} + I \cdot S(\hat{x}_{pred}) + D \cdot \hat{\dot{x}}_{pred,k} \\ \dot{y}_{UAV,k} = P \cdot \hat{y}_{pred,k} + I \cdot S(\hat{y}_{pred}) + D \cdot \hat{\dot{y}}_{pred} \end{cases} \tag{5.12}$$

Chapter 6

Software-in-the-loop simulations

Simulations are the only safe way to conduct preliminary tests of such systems and perform a detailed risk and fault analysis. PX4 provides a SITL (Software-In-The-Loop) package that allows to run the flight stack on a computer. Interaction with the simulator and the simulated model is possible thanks to an offboard API, with the advantage of using exactly the same code both in simulation and on the real embedded system. In our case the chosen simulator is Gazebo, a powerful 3D simulation environment that is typically used with ROS and ROS2.

Simulations allow us to stress the proposed system to failure, in order to understand its limitations and test its robustness. This is done by conservatively adding some offset to the sensors data or increasing their noise well above the actual one, adding randomness to the UGV motion, or purposely removing one or more anchors.

In order to simulate the landing system, the UAV, UGV and UWB antennas models must be created. A realistic simulation requires that the right noise is added to each sensor data, and that the simulated dynamics and scenarios are similar to those in which the real system will be tested. For this reason, several plugins are implemented so that the drone and the rover are equipped with all necessary sensors.

6.1 Models

In this section we illustrate how to model and implement in simulation all necessary sensors and devices.

6.1.1 UWB model

Modelling the ultrawide-band technology is essential to obtain a valid simulation and safely test all localization algorithms. To this end, both tag and anchor models have been created. A specifically created plugin allows the objects defined as anchors to send their position to the tag at a given rate, and the tag to publish on a ROS2 topic the distance from each of them whenever it is received. Moreover it is possible to add noise to these measurements.
The tag is mounted on left leg of the UAV (white box in figure 6.1), while 4 UWB anchors are placed at the four vertices on the UGV surface (small white prisms in figure 6.2).

6.1.2 UAV model

In order to simulate the UAV, we use a model already provided by the PX4 SITL package. In particular, the proposed quadcopter is the 3DR IRIS (figure 6.1), selected because of its similarity to the one used in real flights (section 7.1.2). Indeed, its main features are: weight equal to $1300g$ and a $550mm$ wingspan.
This model has been further customized to include two legs, a UWB tag (the white box on the left leg), a camera (the white box between the black propellers) and a range sensor (the black box between the blue propellers).

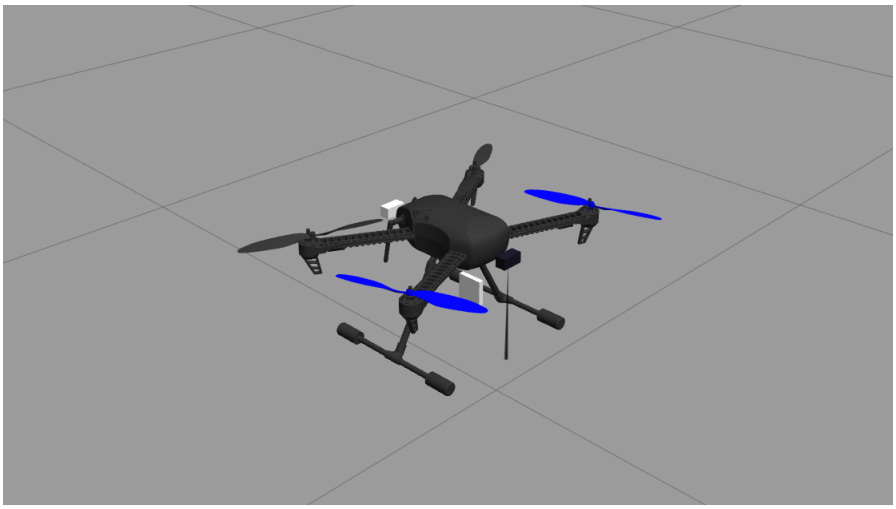

Figure 6.1: Simulated UAV model.

6.1.3 UGV model

The UGV is modeled as a three-wheel vehicle with differential drive. It is composed of a square frame measuring $1m$ x $1m$, equipped with 4 UWB anchors, a compass and an AprilTag marker (figure 6.2). Its dimensions are the same as the UGV used in real tests (section 7.1.3). Moreover, it can be controlled using ROS2 thanks to a differential drive plugin.
The implemented compass publishes on a ROS2 topic the heading data at a given rate. Additionally, there is the possibility to chose the noise affecting this information, so that the sensor behaves as realistically as possible.

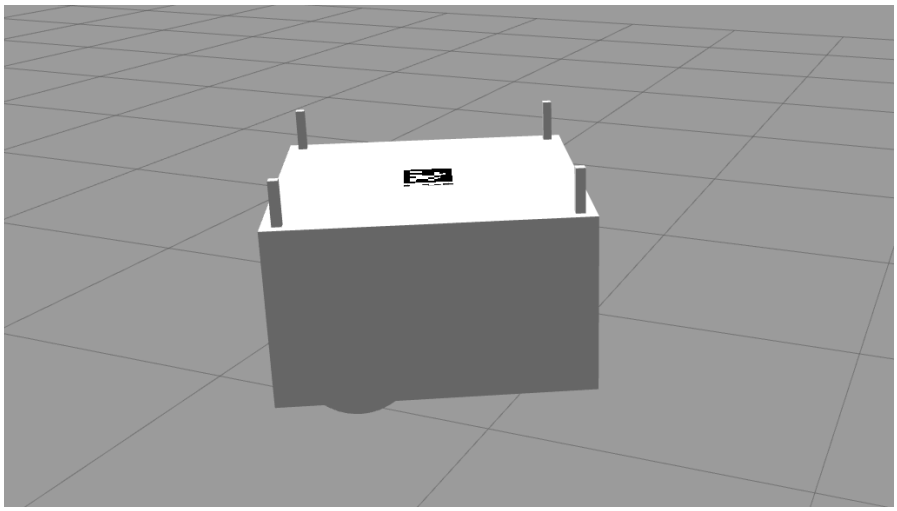

Figure 6.2: Simulated UGV model.

6.2 Landing system simulation

The performed simulations aim to replicate as much as possible the behavior of the real system. For this reason, noise and performance characteristics have been taken from datasheets and experimental tests and implemented on the simulated models. Specifically:

- **UWB antennas** Each distance measurement is affected by white noise with standard deviation $\sigma = 0.1m$, that is nearly twice as much as the real one[7]. This is done to test the system in the worst case scenario.

- **UGV compass** After measuring the real ones, an offset $\Delta\hat{\theta}_{compass} = 25°$ and a standard deviation $\sigma = 0.75°$ have been selected.

- **UGV model** The UAV landing has been simulated in three scenarios: stationary UGV, UGV moving at a $1m/s$ (maximum speed of the real rover (section 7.1.3), UGV moving randomly. In this last case, both linear and steering velocities are affected by noise. This means that the rover steers, accelerates and decelerates randomly, with a top speed of $1m/s$.

Simulation results can be found in paragraph 6.2.3, whereas 6.2.1 and 6.2.2 illustrate the UAV and UGV compasses misalignment problem and its solution respectively. All control and estimation algorithms run at a frequency of $10Hz$.

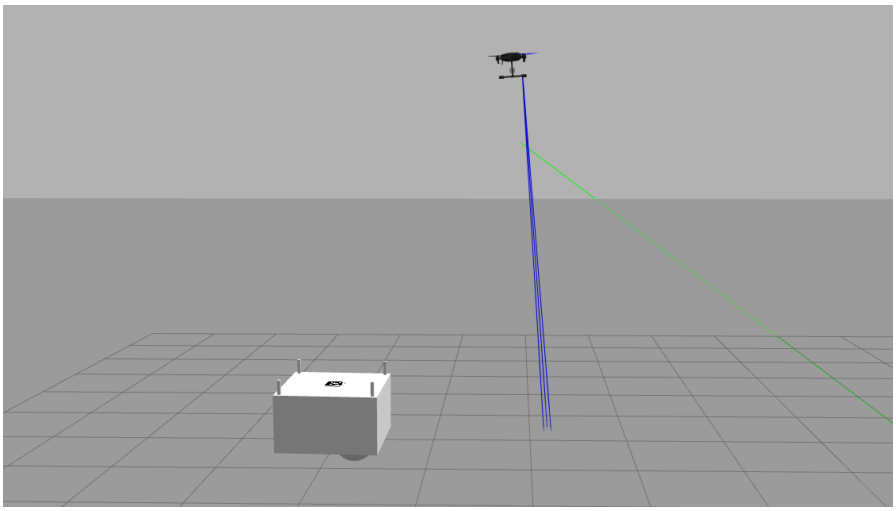

Figure 6.3: UAV chasing the UGV in simulation.

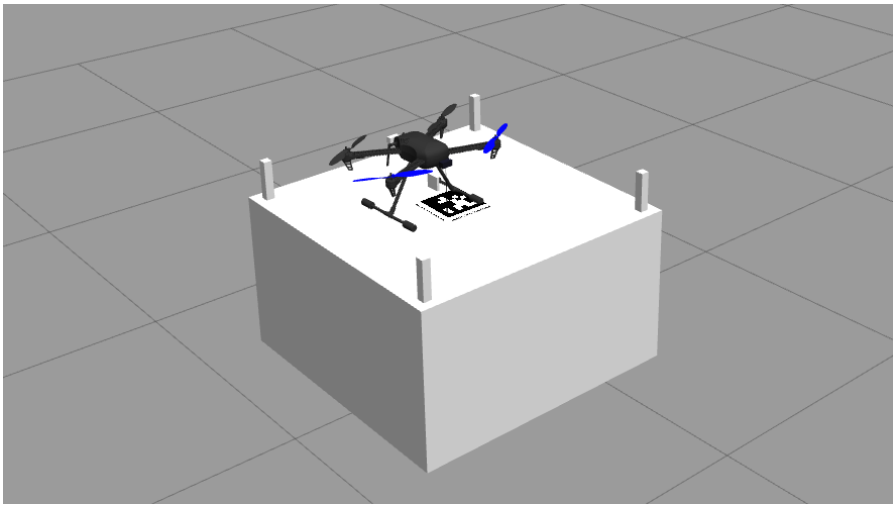

Figure 6.4: UAV landed on the UGV in simulation.

6.2.1 UAV and UGV compasses misalignment effect

The main problem of the proposed landing system is the misalignment $\Delta\hat{\theta}$ between UAV and UGV compasses, which cannot be estimated using the implemented estimation algorithm. Since all position information and control commands must be referenced with respect to the NED frame, consistency between the two compasses is a critical issue. In this paragraph, we illustrate the effect of $\Delta\hat{\theta}$ on the autonomous landing performance.

Let $\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}=0}}$ be the relative position estimate with $\Delta\hat{\theta} = 0$ and $\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}\neq 0}}$ the relative position estimate when $\Delta\hat{\theta} \neq 0$, then:

$$\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}\neq 0}} = \boldsymbol{R}(\boldsymbol{\Delta\hat{\theta}})\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}=0}} \tag{6.1}$$

Where $\boldsymbol{R}(\boldsymbol{\Delta\hat{\theta}})$ is the rotation matrix around the z axis constructed using $\Delta\hat{\theta}$.
The positioning error contribution given by $\Delta\hat{\theta}$ is equal to:

$$e_{pos,\Delta\hat{\theta}} = \left\|\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}\neq 0}} - \hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}=0}}\right\|_2 = 2 \cdot \left\|\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}=0}}\right\|_2 \cdot \left|\sin\left(\frac{\Delta\hat{\theta}}{2}\right)\right| \tag{6.2}$$

It means that the further away the drone and rover are, the greater is the positioning error contribution given by the misalignment error.
The above two formulas can be better understood by looking at figure 6.5. Note that the point $\boldsymbol{O_{UAV}}$ is the center of the UAV, while $\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}=0}}$ is the position of the UGV when $\Delta\hat{\theta} = 0$. The positioning error caused by $\Delta\hat{\theta}$ is given by the length of the line connecting $\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}=0}}$ and $\hat{\boldsymbol{p}}_{\boldsymbol{rel},\boldsymbol{\Delta\hat{\theta}\neq 0}}$.

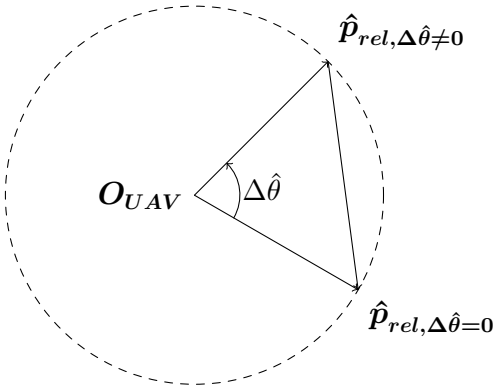

Figure 6.5: Positioning error given by compasses misalignment.

In order to investigate how it affects the mission performance, we simulate the autonomous landing of the UAV on a stationary platform placed at a relative

distance of $x = 5m,\ y = 0m$. For simplicity, the misalignment can be modeled as an offset affecting the UGV compass heading $\theta_{compass}$.

The ideal scenario is shown in pictures 6.6-6.7, where $\Delta\hat{\theta}$ is equal to 0. Indeed, the system shows a fast convergence without oscillations (figure 6.6), with the total positioning error (figure 6.7) being extremely low and reaching a maximum of $30cm$ when the drone is far from the rover and approximately $10cm$ when it is within the area delimited by the anchors. It is also noticeable that the least squares position estimate is really noisy when the UAV and the UGV are far away, and becomes smoother when they get closer. This behavior highly depends on the geometry of the system (equation 3.2).

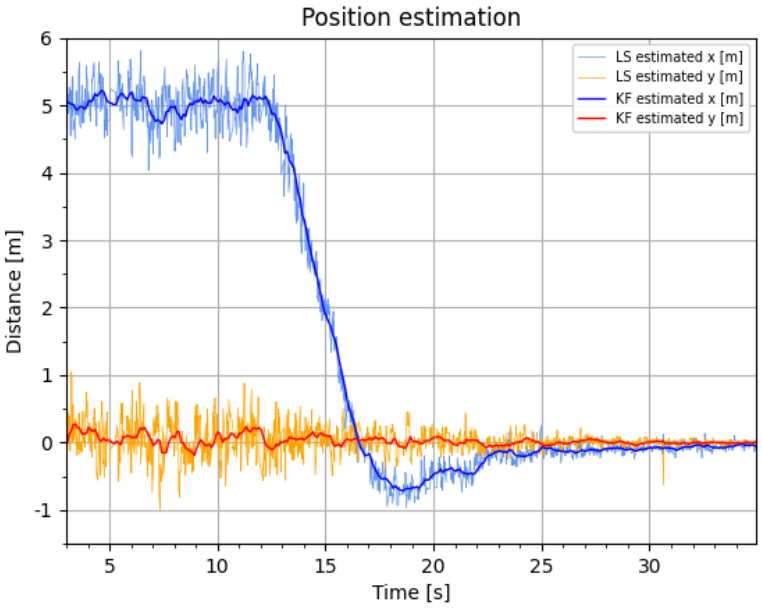

Figure 6.6: System response with $\Delta\hat{\theta} = 0°$ and $D = 0.3$.

Increasing $\Delta\hat{\theta}$ to 25° leads to wider oscillations (figure 6.8) with amplitude $\approx 1m$ and a really high total positioning error (figure 6.9). It is noticeable that this error decreases rapidly as the UAV approaches the UGV (chase phase), since the $\Delta\hat{\theta}$ contribution highly depends on the relative distance between the two (equation 6.2). It can also be seen that the initial position is rotated by 25° from the true one.

Finally, a $\Delta\hat{\theta}$ equal to 40° (6.10-6.11) causes the above described issues to be even more pronounced, while still allowing a successful autonomous landing. In the next paragraph (6.2.2) we will see how this is not obvious at all and highly

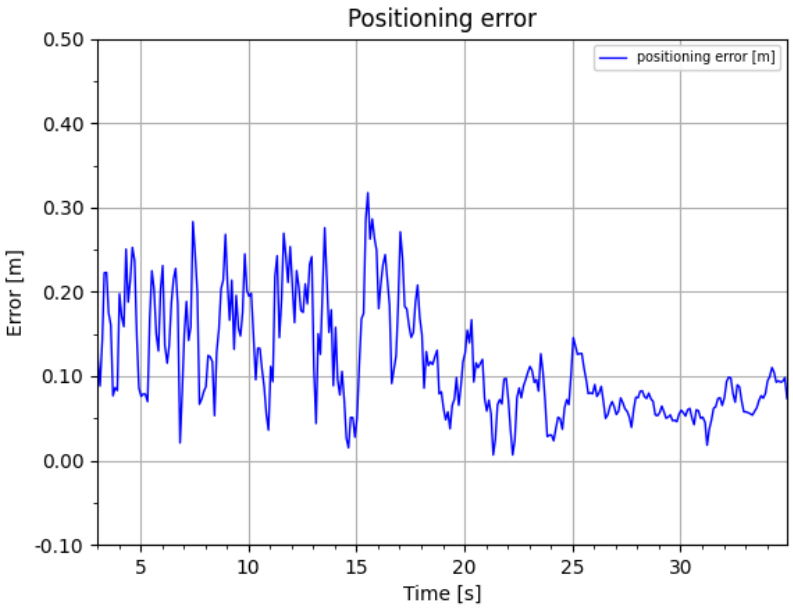

Figure 6.7: Positioning error with $\Delta\hat{\theta} = 0°$ and $D = 0.3$.

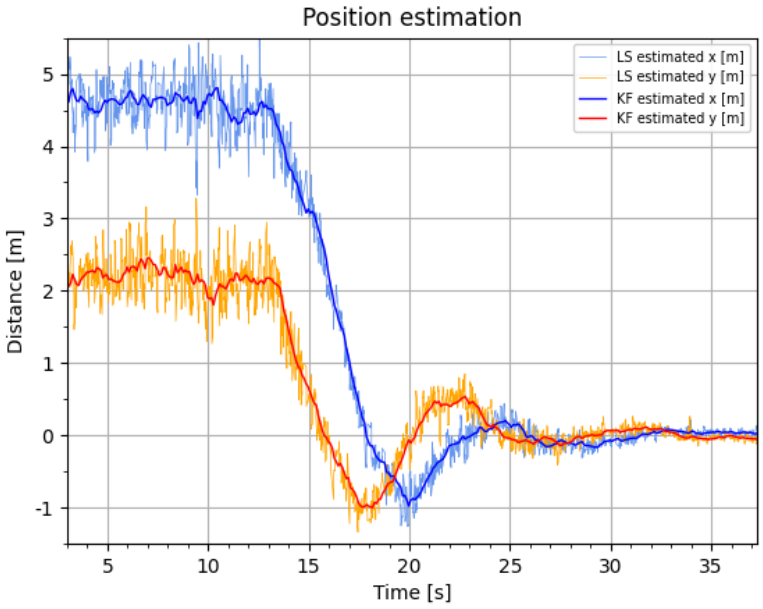

Figure 6.8: System response with $\Delta\hat{\theta} = 25°$ and $D = 0.3$.

Figure 6.9: Positioning error with $\Delta\hat{\theta} = 25°$ and $D = 0.3$.

depends on the derivative term of the controller.

6.2.2 Derivative term effect

The robustness of the proposed landing algorithm with respect to the compasses misalignment largely depends on the derivative term D of the controller (section 5.2.1). Increasing this constant leads to less oscillations and a faster convergence time, but makes the system more sensitive to noise. Hence, D must be chosen as a tradeoff between speed and noise rejection.

The effect of the derivative term on the system response is clearly noticeable in pictures 6.12-6.13-6.14. The chosen parameters are: $\Delta\hat{\theta} = 40°$ and $D = 0$, $D = 0.1$ and $D = 0.3$ respectively.

When $D = 0$ (figure 6.12), the system is characterized by very large oscillations and a long settling time. It can be seen that the landing accuracy (last seconds of the graph) is quite poor.

Increasing D to 0.1 (figure 6.13) largely improves the above descripted behaviour, but the oscillations still cause a fairly inaccurate landing.

Finally, a D equal to 0.3 (figure 6.14) leads to an acceptable control response, while providing a decent noise rejection. This is conservatively chosen as a valid tradeoff, since the real system position estimate noise and spikes may be even more

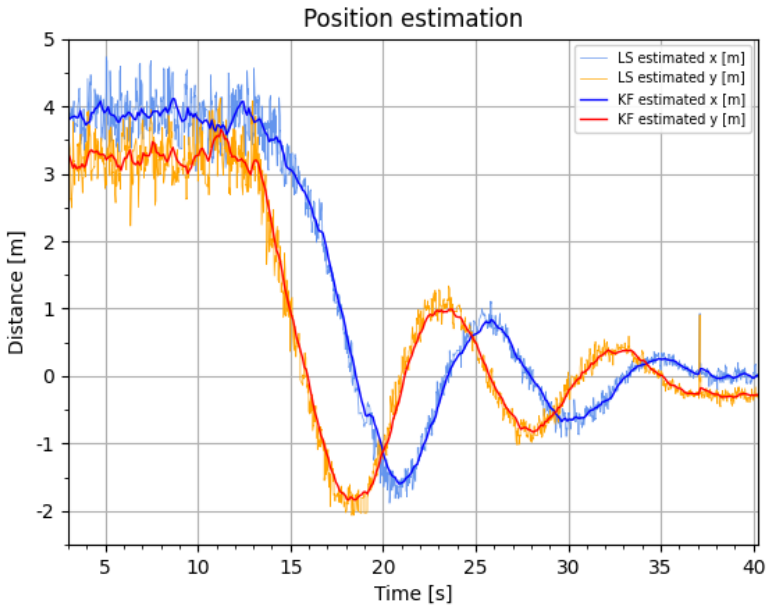

Figure 6.10: System response with $\Delta\hat{\theta} = 40°$ and $D = 0.3$.

Figure 6.11: Positioning error with $\Delta\hat{\theta} = 40°$ and $D = 0.3$.

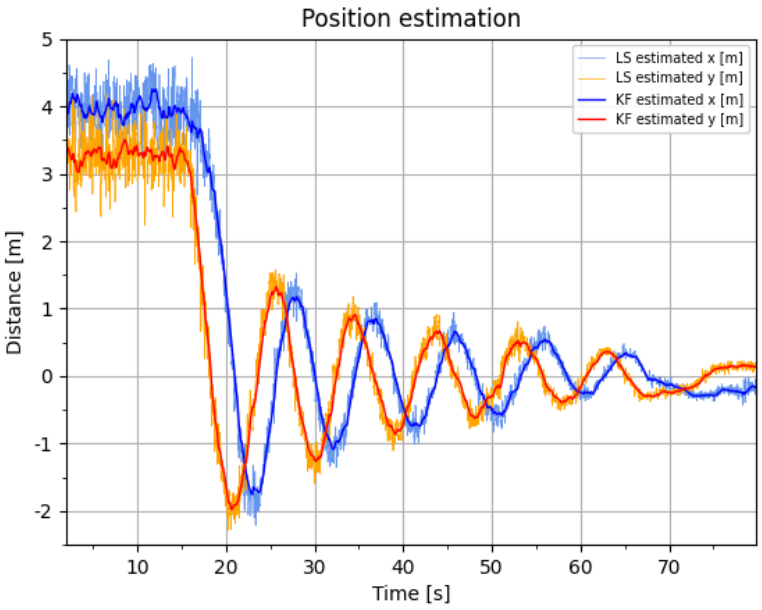

Figure 6.12: System response with $D = 0.0$ and $\Delta\hat{\theta} = 40°$.

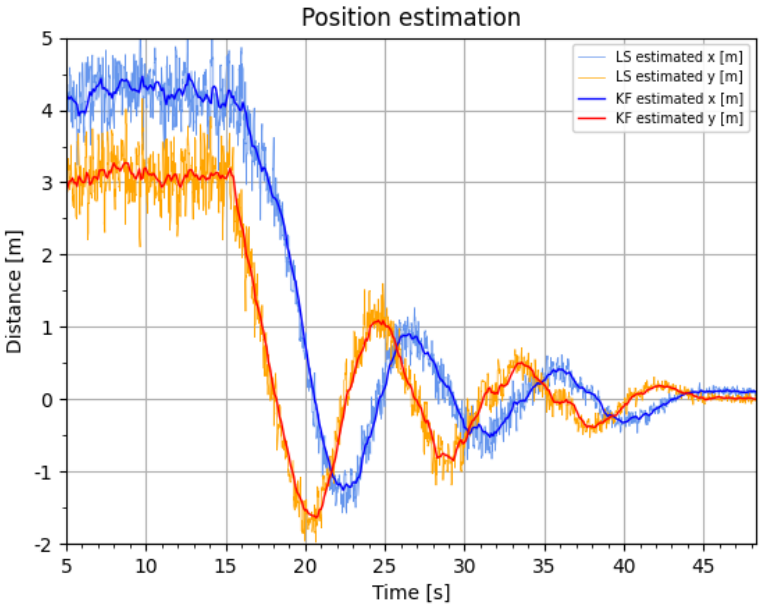

Figure 6.13: System response with $D = 0.1$ and $\Delta\hat{\theta} = 40°$.

pronounced. Also, a further increase in D does not lead to a noticeable performance improvement, but rather makes the system more nervous.

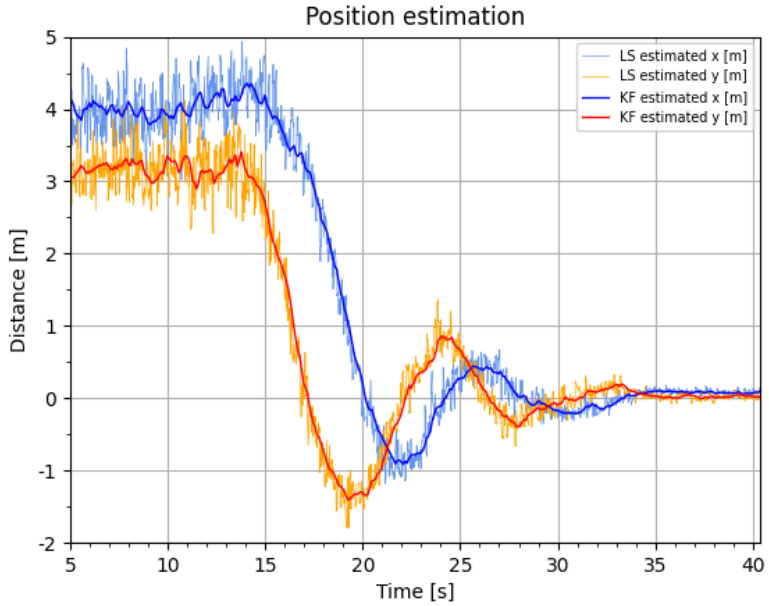

Figure 6.14: System response with $D = 0.3$ and $\Delta\hat{\theta} = 40°$.

6.2.3 Results

The simulated system shows very good performance in both stationary and moving target landings, behaving similarly to the real one (section 7.2) and reaching a landing accuracy of approximately $5cm$.

The simulated landing on the stationary UGV is shown in picture 6.15. The blue, red and green backgrounds represent takeoff, chase and descent phases respectively, while the blue, red and black lines the relative x, y and absolute z estimates. It can be also seen how the drone turns off its motors at $t = 34s$ and lands on the rover with an accuracy of about $5cm$.

In the constantly moving target case (figure 6.16) the descent phase is less smooth than the one in the first scenario. This is due to the UAV attempting the descent only when the relative velocity and the relative distance are small enough. Nevertheless, the convergence speed and the landing accuracy are excellent.

The last scenario is shown in picture 6.17. Here, the rover accelerates, decelerates, and steers unpredictably. It is noticeable how the UAV takes longer to reach the UGV and the descent is even slower, but the performance is still remarkably good.

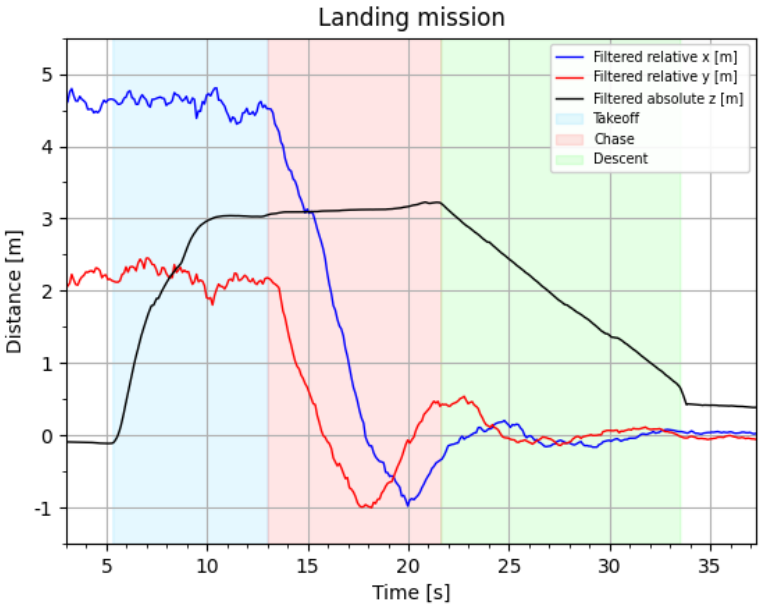

Figure 6.15: Simulated UAV autonomous landing on a stationary UGV.

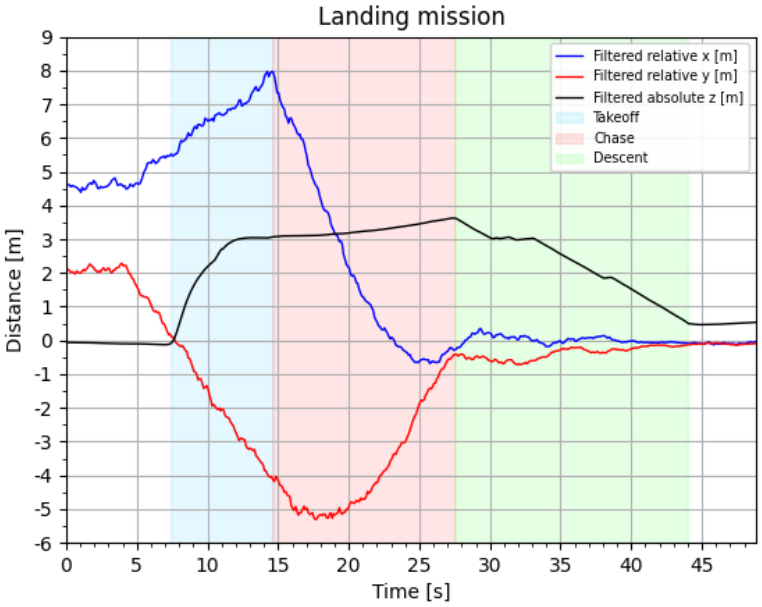

Figure 6.16: Simulated UAV autonomous landing on a constant velocity moving UGV.

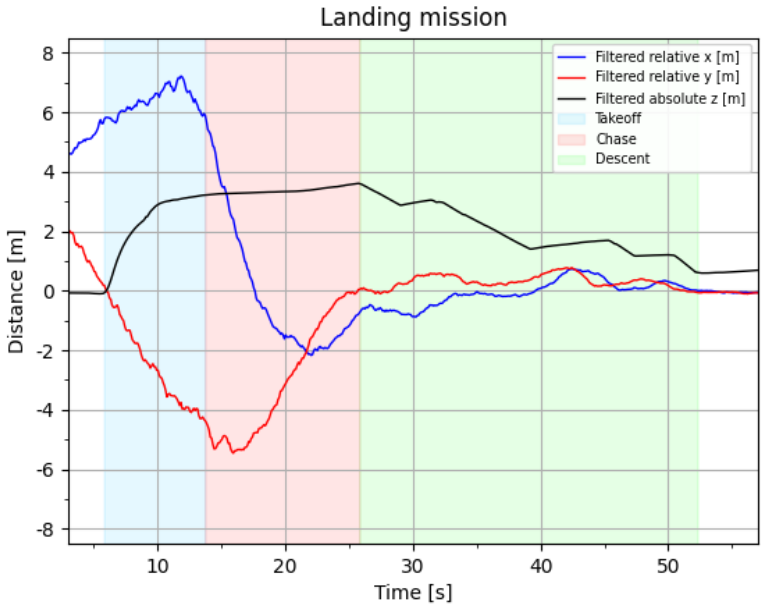

Figure 6.17: Simulated UAV autonomous landing on a randomly moving UGV.

Chapter 7

Experimental testing

In this chapter, the proposed landing algorithm is validated by means of experimental tests. First, we report all the devices used and their specifications. Finally, the experimental test data are shown and analyzed in detail.

7.1 Instrumentation

7.1.1 UWB

The chosen ultrawide-band device is the Decawave EVB1000 board (fig. 7.1). It is one of the most precise devices on the market and can be programmed to work both as an anchor or a tag. Furthermore, it has small dimensions ($\approx 7cm$ x $12cm$ x $1.3cm$) and weight ($\approx 40g$), and the low power consumption ($< 1W$) allows it to be powered with small batteries and work for days just with a regular power bank. It has a ranging precision approximately equal to $2cm$ and an accuracy up to $5cm$ in Line of Sight, with a maximum ranging distance nearly equal to $\approx 150m$ in the absence of obstacles[16].

7.1.2 UAV

The used UAV has been self assembled using the Holybro X500 kit (fig. 7.2) and further equipping it with a Raspberry Pi 4, GPS RTK, Lightware SF11/C range sensor and a Decawave EVB1000 board programmed as a tag. The Raspberry Pi 4 is in charge of executing all the various flight algorithms (chapters 4 and 5) and sending the commands to the autopilot. It is therefore connected via UART to the PX4 flight control board, which manages the received setpoints and controls the drone. Moreover the Raspberry Pi 4 is connected to a wifi network, which allows it to receive commands (e.g. "land", "takeoff", "start autonomous landing") and post and retrieve data (such as the UGV compass heading) from ROS2 topics. The

Figure 7.1: Decawave EVB1000 evaluation board[17].

UWB tag is mounted on the left leg of the drone and connected via USB to the Raspberry Pi 4. It returns the distance from the 4 anchors with a frequency of about $20Hz$.

Figure 7.2: Holybro X500 Kit[18].

7.1.3 UGV

The used rover is the Husky UGV by Clearpath Robotics, shown in picture 7.3. It has a maximum speed of $1m/s$, and dimensions equal to $990mm$ x $670mm$ x

390mm. The huge allowed payload of 75kg and the high torque powertrain allow it to be used in all types of environment[19]. It is further equipped with a 1m x 1m custom wooden landing platform mounted on its top. This is composed of 4 UWB anchors installed on its corners and an electronic compass connected to a Raspberry Pi 4, all powered by several powerbanks. Orientation data are sent to the UAV via wifi, by posting them on a ROS2 topic. Finally, the rover can be controlled either via bluetooth with a joystick or via wifi using a computer.

Figure 7.3: Husky UGV[20].

7.2 Results

Experimental tests succesfully validate the proposed landing system with excellent results, also exhibiting strong similarity to the simulated scenarios. More precisely, the described autonomous takeoff and landing scheme ensures an accuracy up to 5cm, both on fixed and mobile targets. In this paragraph we show three significant tests, carried out within two weeks of each other. These ones have been necessary to further define and improve both the state machine and the estimation and control algorithms. Note that all estimation and control algorithms run at a frequency of 10Hz.

The first autonomous landing on the stationary platform is shown in figure 7.4. It is noticeable that the descent is not smooth at all and the system response is characterized by really large oscillations (amplitude $\approx 2m$). Moreover, the landing accuracy is quite poor, being equal to approximately 30cm. This is due to a whole different autonomous landing algorithm. Its critical issues were:

- a faulty landing detection;

- the derivative value of the controller being much lower than the optimal one ($D = 0.05$ versus $D = 0.3$);
- a different state machine that allowed the drone to climb again as soon as the descent conditions were no longer met;

By subsequently analyzing these data and comparing them with those of the simulations, it was noted that the oscillations were caused by a high misalignment between the drone and rover compass (paragraph 6.2.1), and that the solution was to increase the value of the derivative term of the controller (paragraph 6.2.2). Figure 7.5 shows the position estimated with the Least Squares method as well as

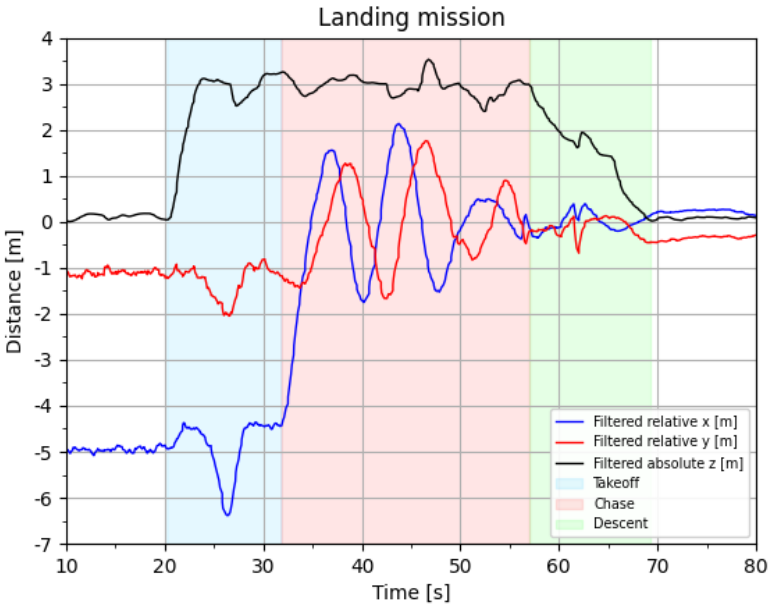

Figure 7.4: First UAV autonomous landing on the stationary UGV - Complete mission chart.

the one subsequently filtered by the Kalman Filter. As can be seen, the latter is much smoother and robust with respect to outliers, while still showing no delay compared to the raw multirated one. Moreover, the variance of the Least Squares estimate is very high at the beginning, when the drone and rover are far away, and visibly decreases when they get closer ($t \geq 32s$). As explained earlier, this depends very much on the system geometry and the GDOP value (equation 3.2).

The performance of the improved version of the autonomous landing algorithm can be seen in figure 7.6. It is noticeable that both the descent phase and the chase one

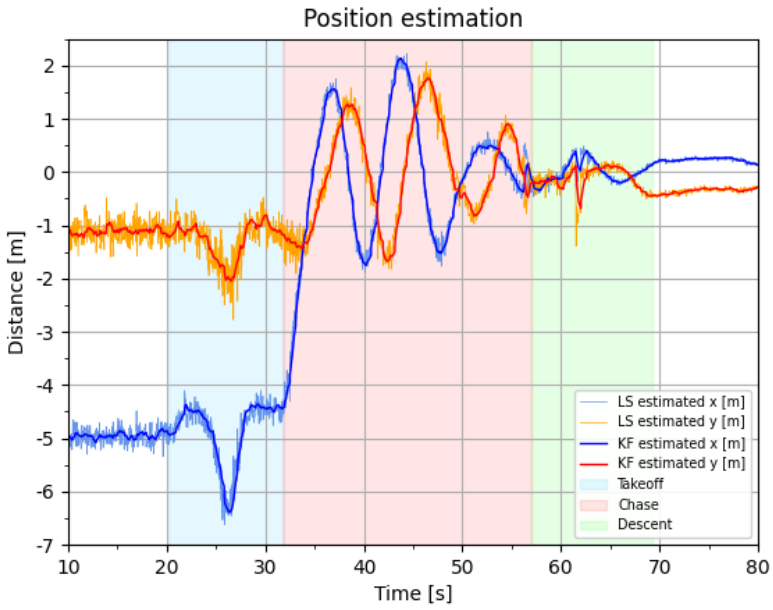

Figure 7.5: First UAV autonomous landing on the stationary UGV - Relative position estimation.

are visibly better than those of the first test, leading to a landing accuracy under $10cm$. The increased value of the derivative term ensures very limited and quickly damped oscillations, allowing the relative position to quickly converge to zero. The descent is much smoother, but the landing detection remains flawed, causing the drone to shut down its motors ($t \approx 112s$) only a long time after landing (t $\approx 100s$). This problem has been subsequently solved, leading to the state machine, controller and estimator described in this thesis.

Finally, from image 7.7 it can be seen that the spikes at the time instants $t \approx 75s$ and $t \approx 86s$ are given by an interruption in the position multilateration. This happens when the drone loses the signal of at least two anchors, since 2D position computation is possible even without one of them.

The last experimental test is shown in figure 7.8. Unlike the previous ones, here the UAV autonomously lands on the UGV moving at approximately $v_{UGV} = 1km/h$. It is noticeable that the results are quite satisfactory: the descent phase is smooth, the relative position quickly converges to zero without oscillations, and the landing detection algorithm works properly, allowing a well-timed shutdown of the motors ($t \approx 45s$) with a landing accuracy under $10cm$.

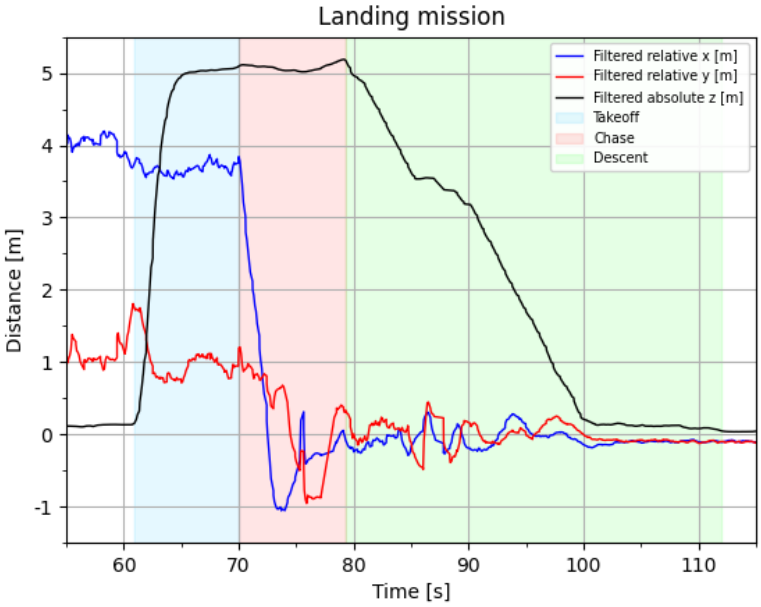

Figure 7.6: First UAV autonomous landing on the stationary UGV - Complete mission chart.

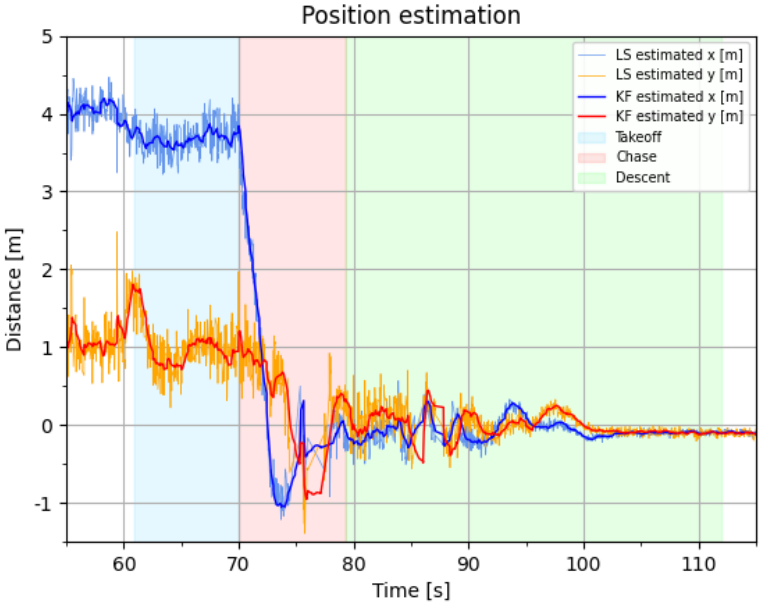

Figure 7.7: Second UAV autonomous landing on the stationary UGV - Relative position estimation.

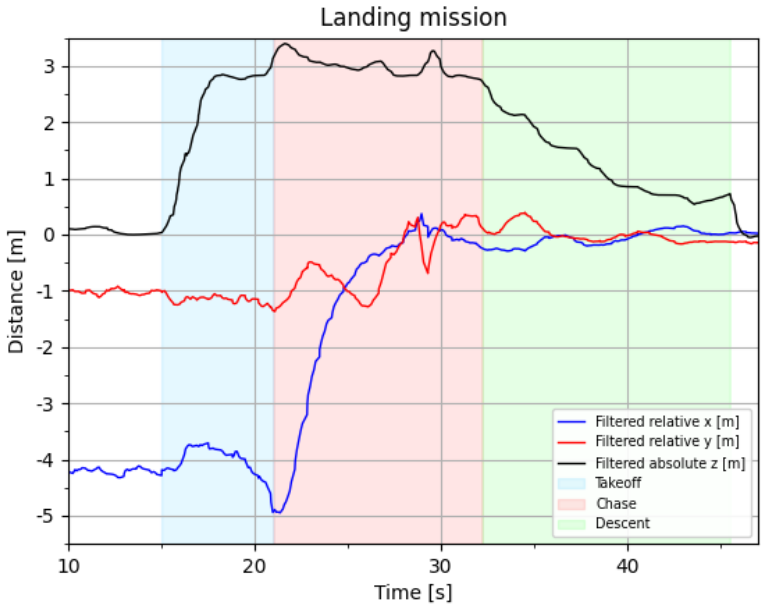

Figure 7.8: UAV autonomous landing on the moving UGV - Complete mission chart.

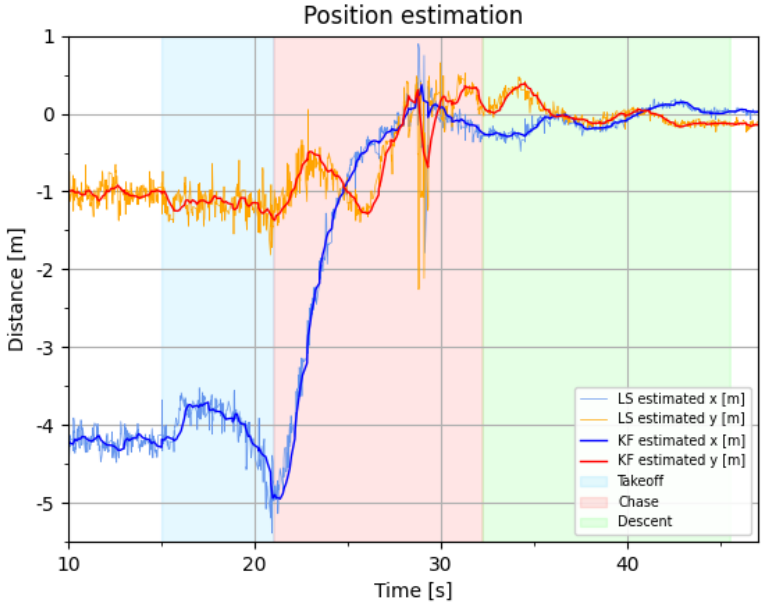

Figure 7.9: UAV autonomous landing on the moving UGV - Position estimation.

Chapter 8

Conlusions

In this thesis, an autonomous landing system based on ultrawide-band technology has been proposed. Software-in-the-loop simulations and experimental tests validate this work with excellent results, ensuring a landing accuracy below $10cm$, both on a stationary and moving platform. More specifically, UWB ranges, UGV compass, and UAV sensors data are fused together by means of loosely coupled linear Kalman Filter. The estimated relative position is then passed to a gain scheduling PID controller that computes the setpoint to be given to the autopilot. The innovation of this work therefore lies in its simplicity and in the low computational power required to achieve an accurate autonomous landing.

8.1 Future work

This work is certainly a great starting point for the development of more robust and high-performance autonomous landing systems. Positioning redundancy could be increased by fusing camera, odometry and GPS data, in order to address the current UWB signal loss or high noise issues. In addition, it may be beneficial using more accurate UAV and UGV models together with a tightly coupled nonlinear filter, to allow autonomous landings at higher speeds. Finally, more sophisticated control architectures, such as nonlinear model predictive control, could be implemented in order to predict the rover position and compute the optimal drone trajectory.

Bibliography

[1] *PX4 Autopilot System Architecture*. https://docs.px4.io/master/en/concept/px4_systems_architecture.html (cit. on p. 4).

[2] *PX4 Autopilot Controller Diagrams*. https://docs.px4.io/master/en/flight_stack/controller_diagrams.html (cit. on pp. 5, 6).

[3] *PX4 Autopilot Reference Frames*. https://docs.px4.io/v1.12/en/ros/external_position_estimation.html (cit. on pp. 6, 7).

[4] G. Breed. «A summary of FCC rules for ultra wideband communications». In: *High Frequency Electronics* (Aug. 2005), pp. 42–44 (cit. on p. 9).

[5] F. Lensund and M. Sjöstedt. *Local positioning system for mobile robots using ultra wide-band technology*. 2018. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-232506 (cit. on p. 10).

[6] Antonio Jiménez and Fernando Seco. «Comparing Decawave and Bespoon UWB location systems: Indoor/outdoor performance analysis». In: Oct. 2016, pp. 1–8. DOI: 10.1109/IPIN.2016.7743686 (cit. on p. 9).

[7] Marko Malajner, Peter Planinšič, and Dušan Gleich. «UWB ranging accuracy». In: *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2015, pp. 61–64. DOI: 10.1109/IWSSIP.2015.7314177 (cit. on pp. 10, 37).

[8] R. Mautz. *Indoor positioning technologies*. 2012. URL: https://www.research-collection.ethz.ch/handle/20.500.11850/54889 (cit. on p. 11).

[9] Binghao Li, Andrew Dempster, and Jian Wang. «3D DOPs for Positioning Applications Using Range Measurements». In: *Wireless Sensor Network* 3 (Jan. 2011), pp. 343–349. DOI: 10.4236/wsn.2011.310037 (cit. on p. 12).

[10] Emanuele Goldoni, Alberto Savioli, Marco Risi, and Paolo Gamba. «Experimental analysis of RSSI-based indoor localization with IEEE 802.15.4». In: May 2010, pp. 71–77. DOI: 10.1109/EW.2010.5483396 (cit. on p. 13).

[11] Maxim Shchekotov. «Indoor Localization Method Based on Wi-Fi Trilateration Technique». In: 2014 (cit. on p. 13).

[12] Abdelmoumen Norrdine. «An Algebraic Solution to the Multilateration Problem». In: (Apr. 2015). DOI: 10.13140/RG.2.1.1681.3602 (cit. on p. 14).

[13] G. Fantin. *UWB localization system for partially GPS-denied robotic applications*. 2019. URL: https://webthesis.biblio.polito.it/10888/ (cit. on p. 14).

[14] Kalman Filter. *Kalman Filter — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Kalman_filter (cit. on p. 18).

[15] «Estimation for Kinematic Models». In: *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Ltd. Chap. 6, pp. 267–299. ISBN: 9780471221272. DOI: https://doi.org/10.1002/0471221279.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0471221279.ch6. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471221279.ch6 (cit. on p. 22).

[16] *Decawave EVB1000 Evaluation Board Datasheet*. https://www.decawave.com/sites/default/files/resources/evb1000-product-brief_3.pdf (cit. on p. 49).

[17] *Decawave EVB1000 Evaluation Board Image*. https://www.decawave.com/product/evk1000-evaluation-kit/ (cit. on p. 50).

[18] *Holybro X500 Kit*. https://shop.holybro.com/x500-kit_p1180.html (cit. on p. 50).

[19] *Husky UGV Overview*. https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/ (cit. on p. 51).

[20] *Husky UGV Image*. https://www.generationrobots.com/en/402177-husky-a200-ugv-mobile-base.html (cit. on p. 51).