POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Robust autonomous landing of a UAV on a high-speed platform

Supervisors

Prof. Marcello CHIABERGE

Dott. Ing. Marco AMBROSIO

Candidate

Leoluca RIGOGLIUSO

December 2022

Summary

Research on autonomous landing methods of UAVs on a moving platform has experienced rapid growth in recent years and found applications in civil and military sectors. The extreme precision of drone landing is required to overcome problems related to the low battery autonomy of drones, through landing in mobile charging stations, but at the same time also finding applications in the most varied sectors ranging from parcel delivery to rescue operations.
This work implements an autonomous algorithm that allows for the landing onto a vehicle that is moving at high speed, through the use of different sensors, chosen depending on the relative drone-rover position. The result is a robust three state machine that makes use of GPS measurements when the drone-rover distance is large, UWB when the rover is nearby and the fusion of information from the camera and the UWB when the drone is landing. The relative position, computed from the UWB sensors with a Least Square algorithm, must be rotated from the rover's mobile system to the NED reference frame. Therefore, a correct estimate of the orientation of the rover and a consistency between the UAV and UGV compasses is of vital importance for an autonomous landing at high speed. This limit is overcome by mounting a camera on the drone that computes the orientation of the apriltag with extreme precision and this information replaces the noisy one of the rover compass. A Kalman filter manages the information coming from the various sensors and generates an estimate of the relative position and relative speed. These are then passed to a PID speed controller that allows accurate and fast tracking and landing on the moving target. Through a purely proportional control over long distances of the rover and a proportional-integrative-derivative control when UAV and UGV are close together, the drone speed value is computed and this is passed to the autopilot which in turn generates the correct thrust of the motors corresponding to that speed. Since the rover landing occurs with a vertical descent after the engines are turned off, a predictive control must be implemented so that the drone predicts the progression of the rover in the next timesteps. The adoption of a predictive control system, the introduction of new sensors and the correction of the misalignment between the compasses of the drone and rover made it possible to reach landing speeds above 30 km/h.

Acknowledgements

Firstly, I would like to start by thanking Professor Chiaberge for giving me the opportunity to challenge myself and work on such a stimulating project. I feel very privileged to have worked in such a motivating and inspiring enviroment like that of the PIC4ser. In particular, I would like to give a special thanks to Marco and Gianluca for their great preparation and availability, without which it would have been impossible to achieve the results obtained.

Thanks to Michelangelo and Gabriele who have been my travelling companions in this thesis. Thank you Michelangelo for the last minute flight.

Thank you to my friends Sciscetto, Riccardo, Ivo, Michele, Federico, Marco, Davide, Samuele, Roberta and Luca who have always helped to make these five years of study more enjoyable, relieving stress in the most intense moments.

Thanks to Claudia, my girlfriend, for her immeasurable support and love. Thank you for the long video calls and interesting talks that have always been a motivating and propulsive push for advancement in my work. Thank you for always being there no matter what.

Thank you to my parents, my brother and my whole family for always pushing me to do better and for rejoicing with me in each of my achievements.

Table of Contents

List of Tables

List of Figures

Acronyms

UWB
Ultra-Wideband

UAV
Unmanned Aerial Vehicle

UGV
Unmanned Ground Vehicle

PID
Proportional Integral Derivative

KF
Kalman Filter

ROS
Federal Communications Commission

GPS
Global Positioning System

RTK
Real Time Kinematic

TOA
Time Of Arrival

TDOA
Time Differential Of Arrival

TOF

Time Of Flight

AOA

Angle Of Arrival

RSSI

Received Signal Strength Indicator

LOS

Line-of-Sight

NLOS

Non-Line-Of-Sight

NFR

Near-Field Ranging

GDOP

Geometric Diluition Process

NED

North East Down

ENU

East North Up

FRD

Forward Right Down

FLU

Forward Left Up

Chapter 1

Introduction

1.1 Objective of the thesis

This work implements an autonomous algorithm that allows for the landing onto a vehicle that is moving at high speed, through the use of different sensors, chosen depending on the relative drone-rover position. The result is a robust three state machine that makes use of GPS measurements when the drone-rover distance is large, UWB when the rover is nearby and the fusion of information from the camera and the UWB when the drone is landing. The correct functionality of the proposed algorithm was first evaluated in a simulated environment provided by Gazebo and then through real tests.

1.2 Organization of the thesis

The thesis consists of six main chapters:

1. **PX4 Autopilot:** This chapter offers an overview of PX4 autopilot, an open source autopilot flight stack that provides many guidance, navigation and control algorithms. This section presents the communication protocol between PX4 and ROS2, the types of messages supported, the different reference systems used and how to control the flight and landing of the drone through the code that runs on a companion computer.

2. **Positioning techniques:** This section provides a theoretical background of the common positioning technique. Then, the functionality of the main sensors used to localise the drone and rover are illustrated. In the last section, a comparison of the accuracy and coverage of various positioning technologies is reported.

3. **Filtering and sensor fusion:** This chapter focuses on the design of the implemented Kalman filter, which handles the information coming from the various sensors and generates an estimate of the relative position and relative velocity. In the last part, the positioning error, due to the incorrect estimation of the rover's orientation, is examined. This limit is overcome by mounting a camera on the drone that computes the orientation of the apriltag with extreme precision . This information replaces the noisy information of the rover's compass.

4. **System control algorithm:** Here, the control algorithm is presented. A PID speed controller, reinforced with a predictive control algorithm, allows accurate and fast tracking and landing on the moving target. The problem of motor shutdown and free fall of the drone when above the platform is deeply investigated.

5. **Simulation:** In the first part of this chapter Gazebo models and plugins are illustrated. In the final part, the ways by which the simulation was conducted is described. The system is stressed by adding errors to the various sensors. Finally, the simulation results are reported.

6. **Experimental analysis:** In this last part the instrumentation used during the tests is described. Finally, the results are reported proving the correct functionality of the proposed solution.

Chapter 2

PX4 Autopilot

2.1 System overview

PX4 is an open source autopilot flight stack that provides many guidance, navigation and control algorithms and several flight mode for different types of vehicles. Autonomous flight modes are completely controlled by the autopilot (takeoff and landing in this work). The user can control the flight via the radio controller (RC)[1]. A companion computer communicates with the PX4-board through a UART to USB adapter. The companion computer used is a Raspberry-pi 4, which in turn communicates via wifi with a ground station running ROS2.
The set of guidance, navigation and control algorithms constitutes one of the main layers of PX4, known as the flight stack. The other fundamental layer of PX4 is the middleware that allows you to communicate with the embedded sensors and with the outside world.

2.1.1 Flight stack and Middleware

The flight stack represents the estimation and control system of PX4. This work will try to improve the estimation and control system to achieve our goal, also through the introduction of external sensors that are not part of PX4. The flight stack represents the living part of PX4, from sensors to engines, with all the intermediate layers of guidance, navigation and control.

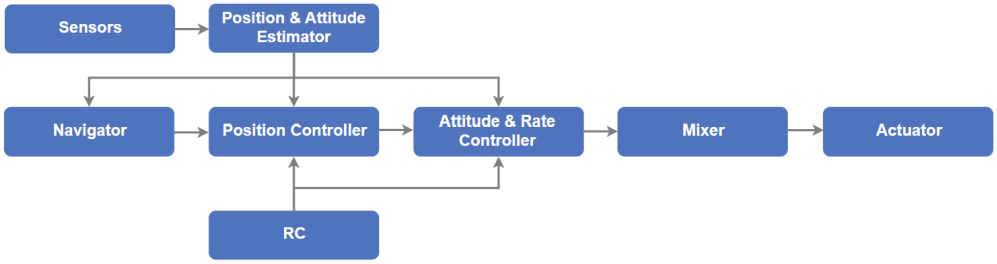

Figure 2.1: Flight Stack[2]

The middleware used by PX4 are basically two. The first one concerns the interface with the companion computer; on the other hand, the second (*simulation layer*) allows you to simulate the PX4 flight code on a desktop computer.
The middleware includes all the functions allowing PX4 to communicate with ROS2. The message types supported within PX4 are UORB type, unlike ROS2 which uses DDS message type. The microRTPS bridge allows the two systems to communicate, through the translation of the two different types of messages, and therefore the real-time publication and subscription of data.

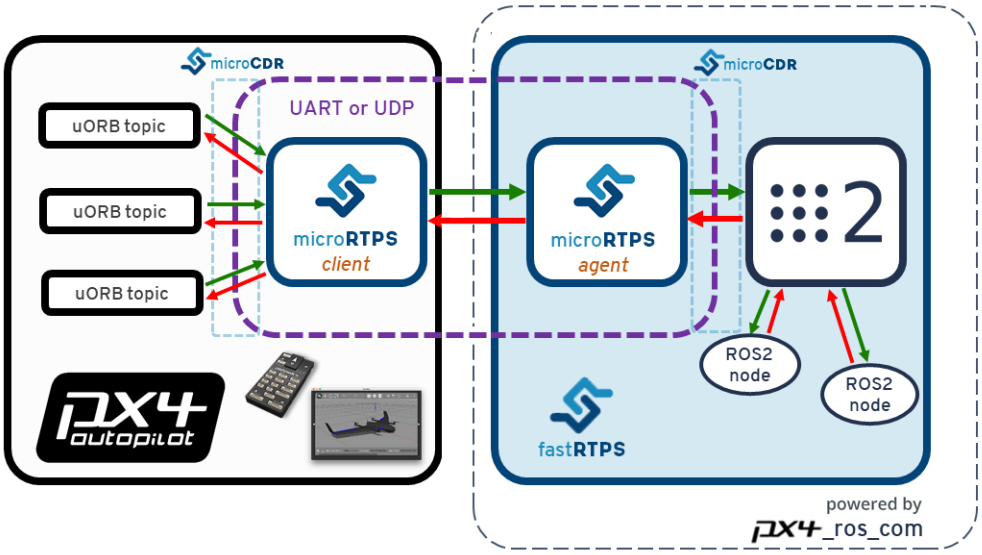

Figure 2.2: MicroRTPS Bridge [3]

The PX4-ROS2 bridge also allows running the flight stack both in a simulated environment on a desktop computer and in a companion computer (Raspberry Pi 4) mounted on the drone.

2.1.2 Control stack

The whole control structure is summarized in the figure. The setpoints are estimated through an EKF. The angle and position control is achieved through a P controller. Velocity and angular rate control also need integrative-derivative control terms. The velocity controller stabilizes the speed of the drone and generates an A_{sp} acceleration. The maximum horizontal speed value allowed to the drone is set in the PX4 firmware in *MPC_XY_VEL_MAX* parameter. After the P and PID controller cascade, the correct thrust setpoint is generated through the mixer to be input to the motors.

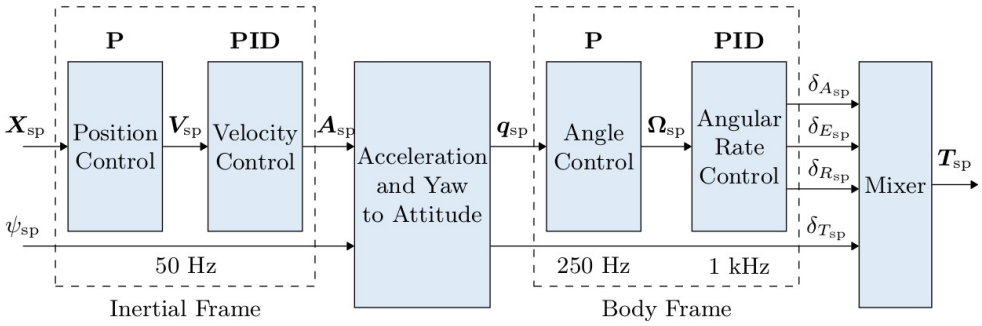

Figure 2.3: Multicopter Control Architecture [4]

2.2 ROS2 Offboard control

This work wants to control the flight and landing of the drone through the code that runs on a companion computer. The *Offboard control mode* is a control mode that helps us in this intent. The main requirements to be respected for this mode to be possible are the following:

- At least one position method available
- Vehicle must be armed before Offboard control mode is engaged
- The setpoints must be received at a frequency >2 Hz before and during Offboard control mode
- Any RC inputs, except changing of the mode, will force an exit from the Offboard control mode [5]

2.3 World and Body frame

PX4 and ROS don't use the same world and body frames. For the body frame, PX4 uses FRD (X Forward, Y Right and Z Down) and ROS uses FLU (X Forward, Y Left, Z Up) convention.
Instead, as far as the fixed reference system is concerned, PX4 uses NED (X North, Y East, Z Down) and ROS uses ENU convention. It is therefore necessary to convert messages from one system to another before being fused in the estimation part or used in the control part.

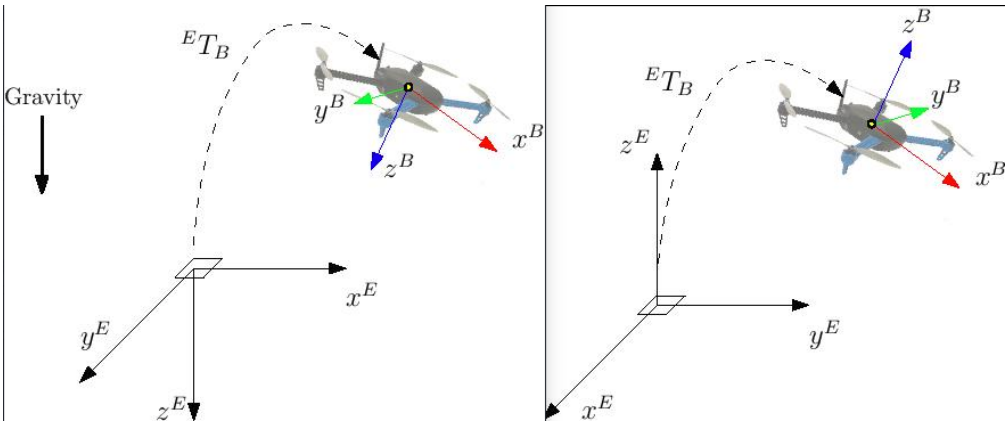

Figure 2.4: Reference frames used by PX4 and ROS [6]

Chapter 3

Positioning techniques

3.1 Classification of Positioning System

There are two types of positioning systems: Global Position System (GPS) and Local Position System (LPS). Due to the limitations of GPS in enclosed spaces or between high buildings, due to a lack of line of sight (LOS), the usage of LPS has become a real necessity to accurately estimate a user's or object's position.

3.1.1 Position estimation techniques

By using estimation methods that are proportional to distance calculation, the measure of receiving positions is derived indirectly. The following are the main positioning techniques :

- *Time of Arrival (TOA):* The distance between the transmitter and receiver can be calculated based on the time it takes for the signal to reach the receiver, the time of flight (TOF). Then this delay time must be multiplied by the speed of light, $d = c \cdot \Delta t$. If there are at least three transmitters in 2D space or three in 3D space, the distances of the transmitters from the receivers can be calculated and thus, the position of the receiver can be uniquely defined. The TOA technique is influenced by several factors, the fundamental one being associated with the synchronization of the clocks of the transmitter and receiver.[7]

- *Time Different of Arrivals (TDOA):* the position of a source transmitter can be calculated through the difference between the time of signal arrival at two or more receivers. The difference in time is multiplied by the speed of light, thus obtaining a difference in the distance between each receiver. This difference removes, then, the error associated with the possible non-synchronization

between transmitter and receiver, since the receivers are affected by the same error. This difference can be plotted as a set of hyperbolic lines in which each pair of receivers represents the foci. The intersection of these lines identifies the position of the source.[8]

- *Angle-of-Arrival:* Angle-based techniques estimate the position of a receiver based on the angle of arrival of the signal. The intersection of the lines (line of bearing or LOB) joining the receiver to each transmitter uniquely identifies the position of the receiver. However, a problem that arises in the identification of the angle of arrival is the need for the use of large antennas, which is not very favourable in terms of size and cost.[9]

- *Received Signal Strength Indicator:* RSSI technique is based on an indicator that depends on the received signal strength. In this method, the estimated position is related to the difference between the transmitted signal and the received signal power. The advantage of this technique is that line of sight is not required. Also, all the equipment that AOA and TDOA techniques require is not required. However, it presents critical problems of reflection, refraction, and multipath fading.

- *Near-Field Ranging (NFR):* The fundamental concept behind this technique is to take advantage of the link between the angle generated by the electric and magnetic fields of the received signal and the distance between the transmitter and receiver. The electric and magnetic field in the proximity of an antenna form an angle of 90 degrees. The EH phase difference decreases with increasing distance from the antenna. As a result, the distance to the transmitter can be determined by a receiver that can measure the electric and magnetic field components of a near-field signal and calculate the EH phase difference. This technology is based on low-frequency signals. Low-frequency signals are better than the high frequency at penetrating obstacles and resisting multipath interference. The main drawback of this technology is the need to use large antennas.[10]

- *Connectivity:* There is no requirement for specialized hardware or time synchronization because the only information used to achieve localization is whether the connection with the anchor nodes is active or not. This technique's main benefit is that it eliminates the need for specialized hardware and node-to-node time synchronization. However, relying just on proximity information results in poor positioning accuracy.

Measurament type	**Positioning technique**	**Description**
AOA	Angle based	Measurament of the angle of arrival of the signal
RSSI	Range based	Measurament of the received signal strength
TOA	Range based	Measurement of the signal propagation delay
TDOA	Range difference based	Measurement of the signal propagation delay difference
NFR	Range-based	Measurement of the EH phase difference
Radio Visibility	Proximity range-free	Connectivity

Table 3.1: Position estimation techniques [11]

3.1.2 Network configuration

The localization method depends on the network architecture and the collection of available measurements.
One of the simplest cases is the problem of determining the position of an agent through N coordinates at known positions. Assuming that the estimated distances are accurate, the position of the agent is given by the intersection of the N circumferences with centres in the anchors and having for radius the respective estimated distances, as in figure 3.1.

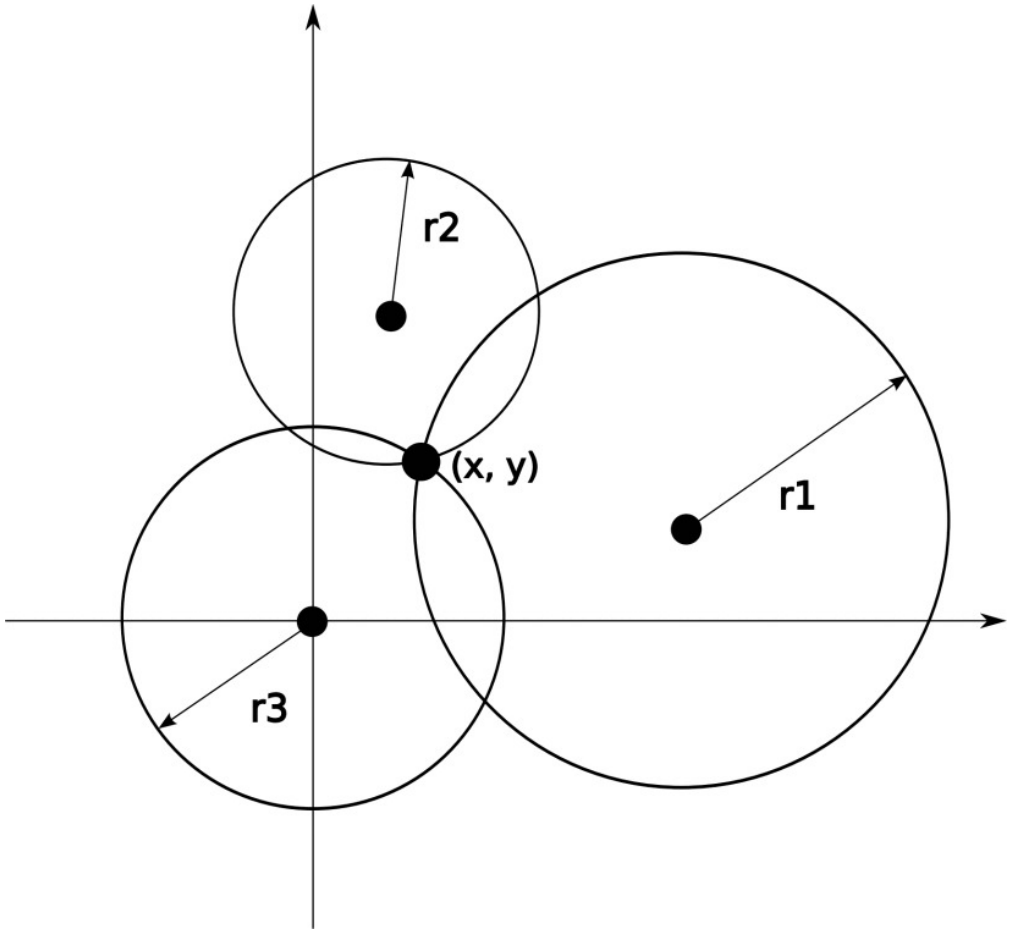

Figure 3.1: Ideal 2D Geometric Positioning [11]

The classification of network types is mainly based on the information we have about the anchors and consequently, the possibility of locating the agent (or tag). If the agent can be identified directly from the known location of a few anchors, a single-hop algorithm can be adopted. If, on the other hand, the number of anchors whose position is known is not sufficient for a direct localisation of the agent, a propagation of information is necessary through the use of a multi-hop algorithm. In the case where the absolute position of no anchor is known, only a relative localisation is possible. This is referred to as anchor-free localisation. A more general classification of how information is processed and the target position calculated leads to the distinction between terminal-centred and network-centred systems. The terminal-centred system directly allows the target to calculate its position once it has received the information. In the network-centred system, the position is calculated by the anchors and then sent to the target.[11] A summary of this classification is presented in table 3.2.

Network Configuration	Description
Anchor-based	some anchors know their positions
Single-hop	the agent can be identified directly from the known location of a few anchors
Multihop	the agent can be identified by intermediate nodes
Anchor-free	only a relative localisation is possible because the absolute position of no anchors is known
Range-free	only connectivity data are used
Terminal-centered	the agent performs its own location on the basis of information coming from anchor nodes
Network-centered	the anchor nodes compute the position of the agent on the basis of the signal coming from the agent

Table 3.2: Network configuration [11]

3.2 Global Positioning System

GNSS is a term denoting a constellation of satellites orbiting the earth and offering positioning, navigation and timing services. 24 satellites are needed to achieve complete coverage of the globe.

3.2.1 Ranging measurements and errors

A GPS device uses satellite measurements to calculate the geocentric coordinates (x, y, z) of a receiver, that is the coordinates relative to the centre of the earth $(0, 0, 0)$.

The receiver receives information from the satellite at time Ti. The satellite sends the following information regarding its position in geocentric coordinates (x_i, y_i, z_i) and time t_i of transmission. Based on the two sent and received times, the satellite-receiver distance ρ_i can be calculated as :

$$\rho_i = c \cdot (T_i - t_i)$$

Where c is the speed of light. At this point, the known data are the data that we get from satellite i:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \\ \rho_i \end{bmatrix}$$

Calculating the geocentric coordinates of the receiver is not an easy problem to solve. The main problem arises from the fact that the satellite clock is not perfectly synchronised with the receiver clock. An error of just one nanosecond results in a deviation from the true position of $0.29 metres$. For this reason, the previously calculated distance ρ_i is called the pseudorange. This error is the same for all pseudoranges since the satellites are synchronised with each other. A common clock bias error b can be defined for all pseudoranges. It is defined as: $b = c\Delta t$, where t is the error in the time of the receiver. The unknown data of the receiver are the following:

$$\begin{bmatrix} x \\ y \\ z \\ b \end{bmatrix}$$

These four unknown data can be computed with the following equation applied to each i-th satellite:

$$\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + b = \rho_i$$

Since the equation has four unknowns, four satellites are required. Furthermore, since the equation is non-linear methods from calculus have to be used, such as Newton's method.
Errors associated with GPS depend on various factors, which must be taken into account to know the magnitude of error affecting our GPS data and in some cases compensate them to increase GPS position accuracy. The main errors are:

- **Atmospheric effects:** Atmospheric conditions cause a delay in the transmitted signal. The main delay is associated with the effects of the ionosphere due

to dispersion phenomena. These phenomena are frequency-dependent and in military and civil situations, which require extreme precision, are corrected by considering two fixed frequency bands, calculating the delay associated with those two frequencies and then, using a mathematical model, estimating the error associated with other frequencies.
However, in cases where high precision is not required, the errors associated with the ionosphere are corrected according to the region or location. In fact, for receivers located in the same place, these corrections are almost similar. The correction error is usually communicated via Satellite Based Augmentation Systems (SBAS).
The errors associated with the troposphere are minor, but less predictable. Humidity and atmospheric pressure have a decisive effect on signal reflection. The only problem is that these effects are more dependent on the specific location where GPS receivers are positioned. Furthermore, they are not frequency-dependent. Consequently, the prediction of these delays is less predictable than those associated with the ionosphere. [12]

- **Multipath distorsion:** Errors associated with multipath effects, due to the reflection of the signal in various types of obstacles such as high buildings, also play a major role.
  Signals affected by long delays are easily identified directly by the receiver. Short signal delays, on the other hand, are more difficult to identify and require the use of specific antennas. [13]

- **Satellite ephemeris and clock errors:** Satellite ephemeris errors depend on the position and trajectory information sent by satellites. This is closely influenced by solar radiation pressure. This error is usually fixed by the ground station passing correction parameters to the receivers.
  Clock errors are due to the drift of the satellites' clock. This, as we will see, can be solved with the use of a differential GPS. [14]

- **Numerical error :** estimated error σ_{num} associated to the numerical computation

Figure 3.2 shows how the errors affect the true receiver position. The intersection of the 4 sphere surfaces represents the estimated point in the ideal case computed with the geometric sphere method.

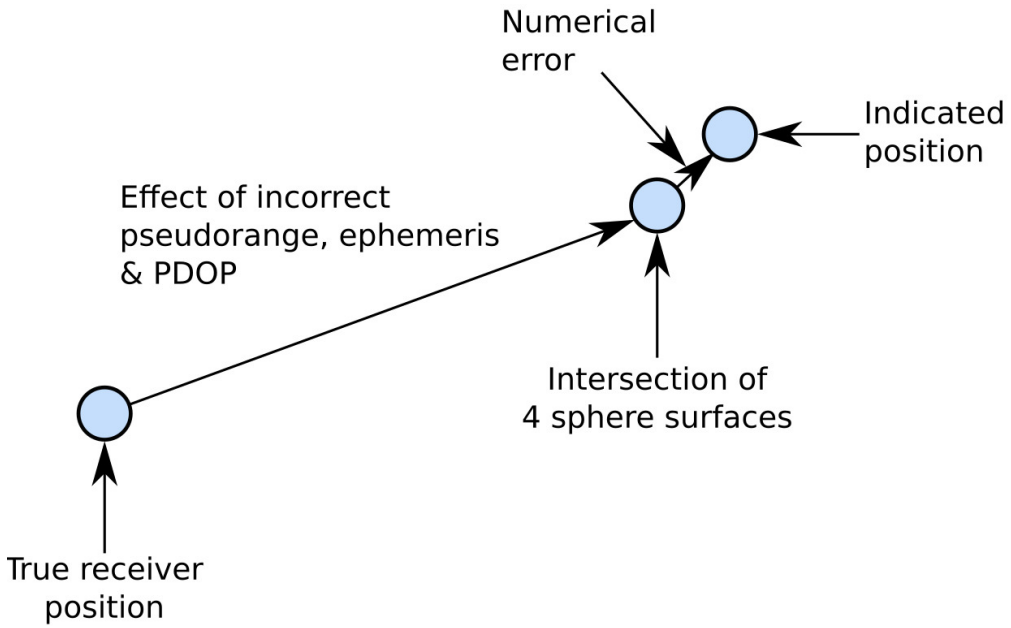

Figure 3.2: GPS errors [15]

In the table 3.3 are reported the user equivalent range errors.

Source	Effect (m)
Ionospheric effects	±5
Tropospheric effects	±0.5
Ephemeris errors	±2.5
Satellite clock errors	±2
Multipath distortion	±1
$3\sigma_r$	±6.7

Table 3.3: User equivalent range errors [15]

The standard deviation of the user equivalent range errors σ_r can be computed as the square root of the square of each error:

$$3\sigma_r = \sqrt{3^2 + 5^2 + 2.5^2 + 2^2 + 1^2 + 0.5^2} m = 6.7m$$

At this point, the standard deviation of receiver position estimate can be computed considering the appropriate PDOP (Position Dilution Of Precision) term. It is a term that indicates how errors in measurements can influence the final estimated

state:

$$GDOP = \frac{\Delta(OutputLocation)}{\Delta(MeasuredData)}$$

Therefore the lower the PDOP the better in terms of accuracy. The standard deviation of the receiver position estimate is computed in this way[15]:

$$\sigma_{rc} = \sqrt{PDOP^2 \cdot \sigma_r^2 + \sigma_{num}^2} = \sqrt{PDOP^2 \cdot 2.2^2 + 1^2} m$$

3.2.2 Standard GPS

The standard GPS computes the receiver's position by solving the preceding equation using one of the analytical methods listed below:

- *Least Square:* it is based on the minimization of the error given by the difference between the pseudoranges ρ_i and the clock bias b in order to find some optimal x, y, z and b,

$$(\hat{x}, \hat{y}, \hat{z}, \hat{b}) = \min_{(x,y,z,b)} \sum_i (\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + bc - \rho_i)^2$$

- *Iterative:* a linearized form of the above equation could be solved with an iterative algorithm (for example Gauss-Newton algorithm)

- *Closed form:* this method could give one or two solutions where only one is probable.

Standard GPS is not able to correct all errors that affect pseudoranges. Only some errors, such as those associated with receiver clock bias can be mitigated by analytical-geometric methods, such as the Hyperboloids method that we mentioned in the previous pages. This generates a low accuracy of about 6 m [16].

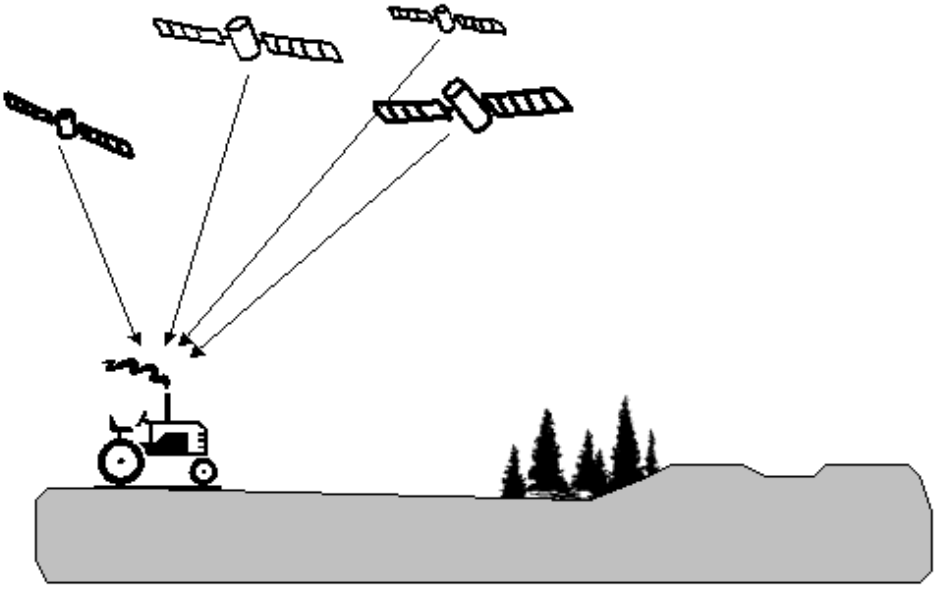

Figure 3.3: Standard GPS [17]

3.2.3 Differential GPS

Differential GPS is a system able to correct the errors of the GPS signals, using a ground reference station at a fixed known position. The position of the ground station is known with high accuracy. Consequently, the difference between the known position and the position measured by the satellites provides the range error. This is the quantity that must be subtracted from each pseudorange.
The range error information is then sent from the ground statiotoS in three distinct ways according to distance.
The correction can be sent directly from the ground station to the GPS receiver via the Ground-Based Augmentation System (GBAS) or Ground-Based Regional Augmentation System (GRAS).
In other cases the correction is first sent to some geostationary satellites which in turn send it to the GPS receiver via the Satellite-Based Augmentation System (SBAS). The table 3.4 shows the coverage area supported by the systems mentioned above.

Correction System	Coverage Area
GBAS	<23 km
GRAS	region area
SBAS	long distance

Table 3.4: Correction GPS system

This correction provides an accuracy below 1 meter. However, SBAS depends on the relative ground station-receiver distance and corrections degrade as the distance increases.

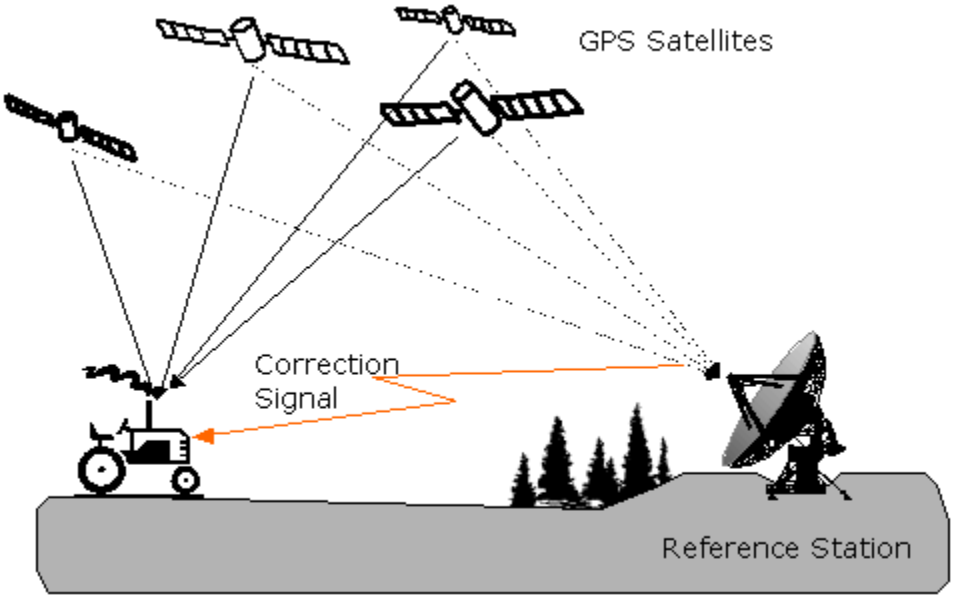

Figure 3.4: Differential GPS [17]

3.2.4 RTK GPS

RTK GPS is a particular type of differential GPS that uses a new technology and communication protocol capable of achieving centimetre-level accuracy.
DGPS and RTK GPS are similar because they both make use of a base station. However, in RTK GPS, the distance calculation is based on the product of the number of carrier cycles between the satellite and rover (receiver) and the carrier wavelength. To this quantity, is added the phase difference calculated by the base station, which allows the errors affecting the GPS to be eliminated. The phase calculated by the base station is compared to that calculated by the rover and the

errors are corrected.
PX4 allows the use of RTK GPS by running QGroundControl in our PC, which acts as a ground station. In addition, the vehicle must be connected via WiFi or Telemetry radio to the ground station. The RTK GPS enables high accuracy, 1 centimetre horizontally and 2 centimetres vertically. [18]

3.2.5 Conversion of GPS data from WGS84 to Cartesian format

GPS coordinates are published on a ROS2 topic, based on the WGS84 geodetic coordinate system.Since these data are to be used in the measurament model, a conversion to the Cartesian reference system is necessary.
To convert the GPS data from the WGS84 system to the Cartesian coordinate system, the origin (reference) of the Cartesian system must be set coherently for both drone and rover. It was decided to take the point where the drone is turned on as the reference. In this way, the Cartesian coordinates of the GPS of the drone and the rover will be published in the same reference system.

3.3 Inertial Measurements Unit

The use of information from the Inertial Measurements Unit (IMU) is of necessary importance for basic flight control. The IMU is responsible for measuring every change in the orientation and acceleration of the vehicle. Consequently, the fusion of GPS data with data from the IMU is vital for any autopilot.
The IMU consists of several inertial sensors, but mainly of two sub sensors: accelerometer and gyroscope.
The accelerometer is responsible for measuring linear acceleration along the three axes X, Y and Z. There are usually three accelerometers, one for each axis.
Data coming from these two sensors are used as feedback for the closed loop autopilot controller. In addition, they can provide information on the attitude, speed and position of the vehicle through integration. The IMU is also able to self-power all its sensors.
Along with the accelerometer and gyroscope, there can be other sensors such as magnetometers, temperature sensors or altimeters. The image below shows a schematic of a basic IMU. [19]

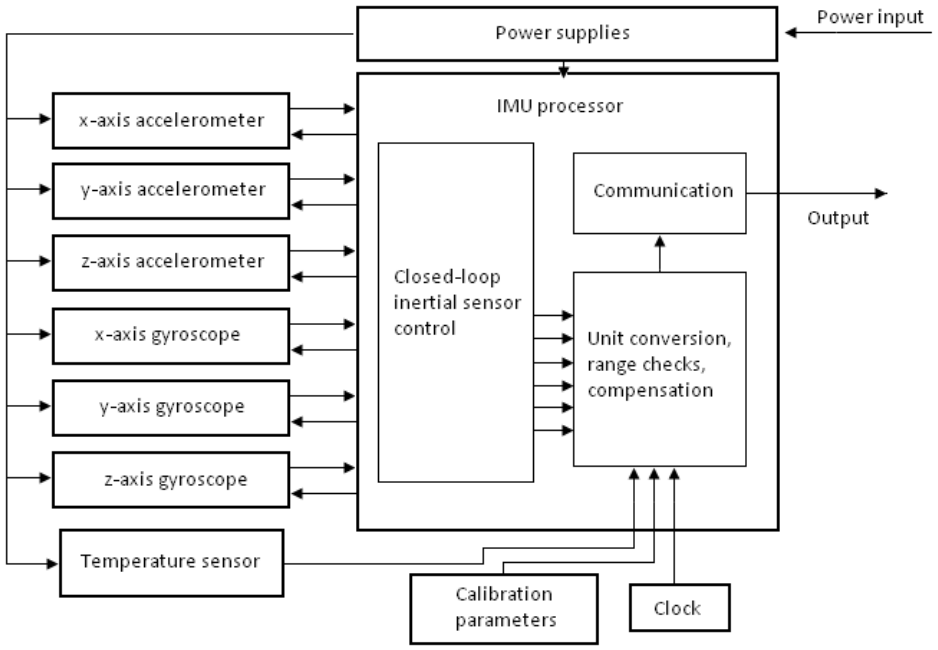

Figure 3.5: IMU schema [19]

3.4 UWB

The Federal Communication Commission (FCC) defines UWB as any device with a fractional bandwidth greater than 0.2 or that covers more than 500MHz of the spectrum, where fractional bandwidth is defined as:

$$FractionalBandwidth = 2\frac{f_H - f_L}{f_H + f_L}$$

Here, fH and fL are the higher and lower frequencies, where measurements have a power spectral density of 10 dB less than its maximum.
As established by the FCC in February 2002, UWB use is permitted in the 3.1-10.6 GHz frequency band and limited power density. The lower power density decreases UWB interference.
Another factor that decreases UWB interference is the fact that it uses Pulse Position Modulation (PPM), with a noise-like signal, which in a closed-loop is difficult to be picked up by other radios technology.[20]

3.4.1 Advantages

The advantages of UWB are mainly related to high levels of accuracy, low costs and low interferences. The main advantages are:

- Low accuracy, ranging accuracy of 10 cm is achieved [21]
- Robust to multipath or attenuation signal thanks to the wide bandwidth
- Low interference due to the lower power densitywith respect to other radio technologies and to the use of PPM with a noise-like signal
- The wide bandwidth allows for a high and low-frequency signal. Consequently, for long distances, low frequencies can be used since low frequencies allow better penetration of obstacles and thus, better propagation than high frequencies.
- High timing capabilities, thanks to the very short time duration of pulse signals
- Low cost and power consumption

3.4.2 Localization with UWB

The estimation of the position of the target via the UWB requires the use of at least three receivers in the case of 2D positioning.
The three receivers are called anchors and are in charge of receiving the signal sent by the tag and calculating the tag-anchors ranges through localization techniques we have discussed in the previous sections, such as TOA or TDOA.
In our work, the UWB is only used to calculate the relative 2D position between drone and rover. The estimated height, on the other hand, is not calculated through the UWB because it is too unreliable. This is due to the fact that the four anchors are positioned on the same level, at the vertices of the rover. The positioning on the same plane results in an unreliable estimate of the z-coordinate.
The choice of using 4 anchors instead of 3 is to provide some redundancy to the system in case one of the sensors stops working or is simply not in the line of sight. Having the distances of each anchor to the tag available, it is possible to calculate the UAV-UGV distance through linear methods such as Least-Square (used in our case) or non-linear methods such as the Gauss-Newton method.

3.5 Camera and Marker description

The Apriltag is a specific marker with a black border and a white inner path. The black border facilitates the detection of the position and orientation of the Apriltag.

The Apriltag, in shape, looks like a QR code. The fundamental difference with the QR node lies in the amount of information carried. Usually, a QR code can carry up to 3 KB of information, unlike an Apriltag, which can carry a maximum of 12 bits of data. The decision to use the Apriltag lies precisely in this simplicity. The Apriltag's simplicity of information allows it to be detected in a robust and precise manner.
Apriltags can be used in many different areas and for many different applications including camera calibration, localization and orientation of an object, etc.

3.5.1 Detection

The detection of the apriltag was done by means of OpenCV by installing the necessary packages. Since there are several Apriltags, we have had to choose one of them. In our work, we used the standard Apriltag "Tag36h11". However, we have a choice of six Apriltag families:

- Tag36h11
- TagStandard41h12
- TagStandard52h13
- TagCircle21h7
- TagCircle49h12
- TagCustom48h12

The Detection algorithm is based on calculating the position of the Apriltag from the identification of the 4 corners. At that point, the position of the centre can be calculated.[22]

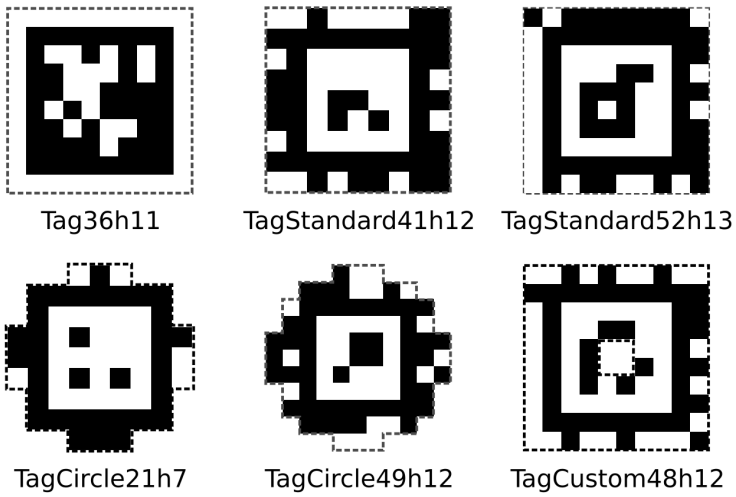

Figure 3.6: AprilTags [23]

3.6 Accuracy and Coverage comparison between different positioning system technologies

In this last section, a comparison of the accuracy and coverage of various positioning technologies is reported.

Technologies	**Coverage (m)**	**Accuracy (m)**
Cameras	1-10	0.0001
UWB	1-50	0.1
Differential GPS	global	0.1
Ultrasound	0.2-10	0.03
RFID	1-50	0.1
WLAN/WIFI	20-50	5

Table 3.5: Comparison between different positioning technologies [24]

It is straightforward to observe how UWB technology offers the best compromise between coverage and accuracy. Although the camera offers one of the best accuracy,

it is not advisable to rely on the camera alone in precision landings. This is because the non visibility of a single piece of the tag results in the no-detection of the tag and therefore the lack of information for the drone to land. For this reason, in our work, the use of the camera is only combined with that of the UWB to correct any errors in estimating the orientation of the rover.

Chapter 4

Filtering and sensor fusion

4.1 Kalman Filter: Theoretical overview

This chapter explains how to obtain a better estimate of the relative position and speed between UAV and UGV using the Kalman filter.
Kalman filter proposed in this work can be considered a 3-state estimation machine, that makes use of GPS measurements when the drone-rover distance is large, UWB when the rover is nearby and the fusion of information from the camera and the UWB when the drone is landing.Further input of the Kalman filter is the orientation coming from the compass of the rover necessary to rotate, in the NED reference system, the relative position outgoing from the Least square and referred to the mobile system of the rover. However, the error affecting the compass output orientation makes the relative rotated position inaccurate and at the same time does not allow the drone to land at high speeds. The orientation of the apriltag allows us to have a more accurate estimate of the relative position and allows landings at high speed.

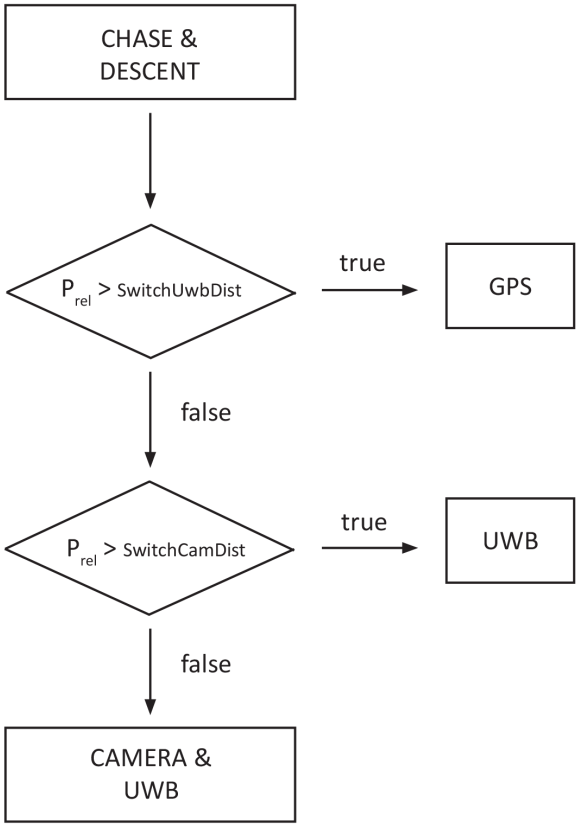

Figure 4.1: 3-state estimation machine

4.2 Filter design

Kalman filter is a linear estimation algorithm. This filter uses a dynamic model of the system, various measurements from sensors and statistical noise to obtain better estimates of the system. The algorithm is divided into two main phases. In the first phase, *predict phase*, an estimate of the state at the current time-step is generated based on the state estimated at the previous time-step. However, this results to be based only on the dynamic model of the system and not on the measurements received by the sensors at that time. Then, in the second phase,

update phase, the estimated state is modified considering a weighted average of predicted state and observations.The Kalman filter operates in a recursive manner, that is, only the current measurements, the previous state and the uncertainty matrix are needed, without the need to know the entire history of the system.

PREDICT PHASE: The predicted state estimate $\hat{x}_{k|k-1}$ and the predicted covariance $\hat{P}_{k|k-1}$ are computed [25]:

$$\hat{x}_{k|k-1} = F_k x_{k-1|k-1} + B_k u_k$$
$$\hat{P}_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

where

- F_k is the state transition model
- B_k is the control-input model which is applied to the control vector
- u_k is the control vector
- Q_k is the covariance associated to the process noise w_k, normally distributed with zero mean:

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

UPDATE PHASE: The a-posteriori state estimate $x_{k|k}$ and estimate covariance $P_{k|k}$ are computed:

$$y_k = z_k - H_k \hat{x}_{k|k-1}$$
$$S_k = H_k \hat{P}_{k|k-1} H_k^T + R_k$$
$$K_k = \hat{P}_{k|k-1} H_k^T S_k^{-1}$$
$$x_{k|k} = \hat{x}_{k|k-1} + k_k y_k$$
$$P_{k|k} = (I - k_k H_k) \hat{P}_{k|k-1}$$
$$y_{k|k} = z_k - H_k x_{k|k}$$

where

- z_k is the measurement(or observation)
- H_k is the observation model
- S_k is the innovation covariance
- K_k is the optimal Kalman gain
- R_k is the covariance associated to the observation noise v_k, Gaussian distributed white noise:

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

P_0, Q_k and R_k must be chosen. Usually, this is done by trial and error and by considerations about the dynamic system

4.2.1 Prediction model

The state $\hat{x}$ of our system is described by 15 values:

- 6 states concerning the position, velocity, and acceleration of the drone along x, y
- 2 states regarding the position and velocity of the drone along z
- 4 states for the position and velocity of the rover along x, y
- 1 state regarding the height of the rover
- 1 state concerning the orientation of the rover with respect to the NED reference frame
- 1 state regarding the orientation error of the rover, given by the difference between the noisy orientation of the compass and the one given by the apriltag

$$\hat{x} = \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ z_{UAV,k} \\ \dot{z}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ z_{UGV,k} \\ \theta_{UGV,k} \\ \theta_{error,k} \end{bmatrix}$$

The dynamic model considered for the drone and the rover is based on considerations and assumptions about their motion. Based on these considerations, the differential equations and coherently the 'sub-matrices' of the state transition model have been written. Then:

- **costant acceleration model** for the drone in the motion along x, y

$$x_{UAV,K} = x_{UAV,k-1} + \dot{x}_{UAV,k-1}dt + \frac{1}{2}\ddot{x}_{UAV,k-1}dt^2$$

$$\dot{x}_{UAV,k} = \dot{x}_{UAV,k-1} + \ddot{x}_{UAV,k-1}dt$$
$$\ddot{x}_{UAV,k} = \ddot{x}_{UAV,k-1}$$

$$y_{UAV,K} = y_{UAV,k-1} + \dot{y}_{UAV,k-1}dt + \frac{1}{2}\ddot{y}_{UAV,k-1}dt^2$$
$$\dot{y}_{UAV,k} = \dot{y}_{UAV,k-1} + \ddot{y}_{UAV,k-1}dt$$
$$\ddot{y}_{UAV,k} = \ddot{y}_{UAV,k-1}$$

In terms of state transition model matrix, these differential equations translate with a pair of identical matrices of this form:

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

It is easy to see how this matrix, multiplied by the state vector, expresses the uniformly accelerated rectilinear motion in matrix form.

- **costant velocity model** for the drone in the motion along z

$$x_{UGV,K} = x_{UGV,k-1} + \dot{x}_{UGV,k-1}dt$$
$$\dot{x}_{UGV,k} = \dot{x}_{UGV,k-1}$$

$$y_{UGV,K} = y_{UGV,k-1} + \dot{y}_{UGV,k-1}dt$$
$$\dot{y}_{UGV,k} = \dot{y}_{UGV,k-1}$$

In terms of state transition model matrix, these differential equations translate with a pair of identical matrices of this form:

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

It is easy to see how this matrix, multiplied by the state vector, expresses the motion at constant speed in matrix form.

- **costant velocity model** for the rover in the motion along x, y

$$x_{UGV,K} = x_{UGV,k-1} + \dot{x}_{UGV,k-1}dt$$
$$\dot{x}_{UGV,k} = \dot{x}_{UGV,k-1}$$

$$y_{UGV,K} = y_{UGV,k-1} + \dot{y}_{UGV,k-1}dt$$
$$\dot{y}_{UGV,k} = \dot{y}_{UGV,k-1}$$

  These differential equations are translated, in terms of state transition model matrix, with a pair of matrices equal to F_2:

$$F_3 = F_2 = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$$

  as F_2 this matrix, multiplied by the state vector, expresses the constant velocity motion in matrix form

- **null velocity** for the rover along z

$$z_{UGV,K} = z_{UGV,k-1}$$

  In this case, the state transition model matrix is trivially 1.

- **null variation in the orientation** for the rover

$$\theta_{UGV,K} = \theta_{UGV,k-1}$$

  In this case, the state transition model matrix is trivially 1.

- **null variation in the orientation error** for the rover

$$\theta_{error,K} = \theta_{error,k-1}$$

  In this case, the state transition model matrix is trivially 1.

In our system,the control input model B_k is equal to zero, so the state vector at time k is

$$x_k = F_k x_{k-1} + w_k$$

where F_k is

$$F_k = \begin{bmatrix} F_1 & 0_{3\times3} & 0_{3\times2} & 0_{3\times2} & (0_{3\times2} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times3} & F_1 & 0_{3\times2} & 0_{3\times2} & 0_{3\times2} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\ 0_{2\times3} & 0_{2\times3} & F_2 & 0_{2\times2} & 0_{2\times2} & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0_{2\times3} & 0_{2\times3} & 0_{2\times2} & F_2 & 0_{2\times2} & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0_{2\times3} & 0_{2\times3} & 0_{2\times2} & 0_{2\times2} & F_2 & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 1 & 0 & 0 \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0 & 1 & 0 \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0 & 0 & 1 \end{bmatrix}$$

The covariance matrix P_0 is initialized as an identity matrix.
The matrix Q is chosen considering a Discrete constant white noise. The matrix Q can have dimensions 2, 3, and 4 and is equal to [26]:

$$Q = GG^T variance$$

where G is the process noise for time k.
For a constant velocity model is equal to:

$$\begin{bmatrix} 0.5dt^2 \\ dt \end{bmatrix}$$

For a constant acceleration model, G is equal to:

$$\begin{bmatrix} 0.5dt^2 \\ dt \\ 1 \end{bmatrix}$$

Then, assuming the motion of UAV along x and y as a constant acceleration model, the covariance associated to the process noise for the states $x_{UAV}, \dot{x_{UAV}}\, \ddot{x_{UAV}}$ and for the states $y_{UAV}, \dot{y_{UAV}}\, \ddot{y_{UAV}}$ is :

$$Q_1 = \sigma_{UAV} \begin{bmatrix} 0.5dt^2 \\ dt \\ 1 \end{bmatrix} \begin{bmatrix} 0.5dt^2 & dt & 1 \end{bmatrix}$$

In the same way, assuming the motion of UAV along z and of UGV along x,y as a constant velocity model:

$$Q_2 = \sigma_{UAV(UGV)} \begin{bmatrix} 0.5dt^2 \\ dt \end{bmatrix} \begin{bmatrix} 0.5dt^2 & dt \end{bmatrix}$$

Clearly, the covariance associated to the process noise, for the orientation of the rover and for errors in the orientation (last state), is simply equal to the variance associated with each of them. Then, the overall covariance matrix Q is:

$$Q_k = \begin{bmatrix} Q_1 & 0_{3\times3} & 0_{3\times2} & 0_{3\times2} & (0_{3\times2} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\ 0_{3\times3} & Q_1 & 0_{3\times2} & 0_{3\times2} & 0_{3\times2} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\ 0_{2\times3} & 0_{2\times3} & Q_2 & 0_{2\times2} & 0_{2\times2} & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0_{2\times3} & 0_{2\times3} & 0_{2\times2} & Q_2 & 0_{2\times2} & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0_{2\times3} & 0_{2\times3} & 0_{2\times2} & 0_{2\times2} & Q_2 & 0_{2\times1} & 0_{2\times1} & 0_{2\times1} \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & \sigma_{z_{UGV}} & 0 & 0 \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0 & \sigma_\theta & 0 \\ 0_{1\times3} & 0_{1\times3} & 0_{1\times2} & 0_{1\times2} & 0_{1\times2} & 0 & 0 & \sigma_{\theta_{error}} \end{bmatrix}$$

4.2.2 Measurement model

The z_k measurements of the various sensors allow us to obtain a better estimate of our state variables. We must then associate each measurement with the corresponding state variable or combination of state variables, via the observation matrix H_k. Furthermore, depending on the accuracy of each sensor, an observation error will be associated with each measurement. In the model, this observation error is translated as the covariance matrix R_k. The higher the value associated with R, the less we trust the respective measurement. We can write five different measurement models:

- **GPS rover observation model:** The rover's GPS data must be converted into Cartesian XYZ coordinates before being used in the Measurement Model, how we have seen in chapter 3.2.5

$$z_{GPS,rover} = \begin{bmatrix} x_{UGV,k} \\ y_{UGV,k} \\ z_{UGV,k} \end{bmatrix} + v_{GPS,k}$$

$$z_{GPS,rover} = \begin{bmatrix} 1 & 0_{1\times2} & 0 & 0_{1\times2} & 0 & 0_{1\times8} \\ 0 & 0_{1\times2} & 1 & 0_{1\times2} & 0 & 0_{1\times8} \\ 0 & 0_{1\times2} & 0 & 0_{1\times2} & 1 & 0_{1\times8} \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ z_{UAV,k} \\ \dot{z}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ z_{UGV,k} \\ \theta_{UGV,k} \\ \theta_{error,k} \end{bmatrix} + v_{GPS,k}$$

where:

$$H_{GPS,k} = \begin{bmatrix} 1 & 0_{1\times2} & 0 & 0_{1\times2} & 0 & 0_{1\times8} \\ 0 & 0_{1\times2} & 1 & 0_{1\times2} & 0 & 0_{1\times8} \\ 0 & 0_{1\times2} & 0 & 0_{1\times2} & 1 & 0_{1\times8} \end{bmatrix}$$

- **UWB observation model:** The Least Square provides the relative drone-rover position $\hat{p}_{UWB}^{UGV}$ in the UGV's mobile reference system. This is rotated in the NED reference system before being used in the measurement model, as we will see in the section 4.3

$$z_{UWB} = \begin{bmatrix} x_{UAV} - x_{UGV,k} \\ y_{UAV,k} - y_{UGV,k} \end{bmatrix} + v_{UWB,k}$$

$$z_{UWB} = \begin{bmatrix} 1 & 0_{1\times2} & 0 & 0_{1\times4} & -1 & 0 & 0 & 0_{1\times4} \\ 0 & 0_{1\times2} & 1 & 0_{1\times4} & 0 & 0 & -1 & 0_{1\times4} \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ z_{UAV,k} \\ \dot{z}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ z_{UGV,k} \\ \theta_{UGV,k} \\ \theta_{error,k} \end{bmatrix} + v_{UWB,k}$$

where:

$$H_{UWB} = \begin{bmatrix} 1 & 0_{1\times2} & 0 & 0_{1\times4} & -1 & 0 & 0 & 0_{1\times4} \\ 0 & 0_{1\times2} & 1 & 0_{1\times4} & 0 & 0 & -1 & 0_{1\times4} \end{bmatrix}$$

- **Camera observation model:** The camera, mounted on the UAV underneath, returns the relative UAV-UGV position and orientation of the rover.

$$z_{cam} = \begin{bmatrix} x_{UAV} - x_{UGV,k} \\ y_{UAV,k} - y_{UGV,k} \\ z_{UAV,k} - z_{UGV,k} \\ \theta_{error} \end{bmatrix} + v_{cam,k}$$

$$z_{cam} = \begin{bmatrix} 1 & 0_{1\times2} & 0 & 0_{1\times2} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0_{1\times2} & 1 & 0_{1\times2} & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0_{1\times2} & 0 & 0_{1\times2} & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0_{1\times2} & 0 & 0_{1\times2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ z_{UAV,k} \\ \dot{z}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ z_{UGV,k} \\ \theta_{UGV,k} \\ \theta_{error,k} \end{bmatrix} + v_{cam,k}$$

where:

$$H_{cam} = \begin{bmatrix} 1 & 0_{1\times2} & 0 & 0_{1\times2} & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0_{1\times2} & 1 & 0_{1\times2} & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0_{1\times2} & 0 & 0_{1\times2} & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0_{1\times2} & 0 & 0_{1\times2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Compass observation model:** The compass outputs the orientation of the rover.

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

$$z_{compass} = \begin{bmatrix} 0_{1\times 13} & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ z_{UAV,k} \\ \dot{z}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ z_{UGV,k} \\ \theta_{UGV,k} \\ \theta_{error,k} \end{bmatrix} + v_{compass,k}$$

where:

$$H_{compass,k} = \begin{bmatrix} 0_{1\times 13} & 1 & 0 \end{bmatrix}$$

- **PX4 observation model:** The sensors integrated in PX4 allow us to know the position, speed, acceleration and orientation of the drone.

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

$$z_{PX4} = \begin{bmatrix} I_{1\times 9} & 0_{1\times 6} \end{bmatrix} \begin{bmatrix} x_{UAV,k} \\ \dot{x}_{UAV,k} \\ \ddot{x}_{UAV,k} \\ y_{UAV,k} \\ \dot{y}_{UAV,k} \\ \ddot{y}_{UAV,k} \\ z_{UAV,k} \\ \dot{z}_{UAV,k} \\ x_{UGV,k} \\ \dot{x}_{UGV,k} \\ y_{UGV,k} \\ \dot{y}_{UGV,k} \\ z_{UGV,k} \\ \theta_{UGV,k} \\ \theta_{error,k} \end{bmatrix} + v_{PX4,k}$$

where:

$$H_{PX4,k} = \begin{bmatrix} I_{1\times 9} & 0_{1\times 6} \end{bmatrix}$$

4.3 Rotation of UWB ranges in NED frame

The UAV-UGV relative distance is computed through a multilateration algorithm from the anchor and tag distances. The Least Square, used as multilateration algorithm, allows the computation of the relative position $\hat{p}_{UWB}^{UGV}$ between drone and rover in the UGV's mobile reference system. This relative position must be rotated in the NED reference system before being used in the filter. However, using the compass angle of the rover to rotate the relative position causes some problems, because the compass output can be affected by misalignment error. Hence, we use the orientation of the apriltag as an additional source of information regarding the orientation of the rover.

4.3.1 Compasses misalignment effect

This work aims to create a system that allows the drone to land when the UGV is moving at high speeds. One of the main limitations of the previous work is to be attributed to an estimate of the orientation of the rover affected by the error, which caused continuous oscillation and continuous corrections in the UAV trajectory. An orientation affected by error leads the drone to move to a different point than

desired. The error in the relative position increases as the error on the angle (fig. 4.2) and the relative distance UAV-UGV (fig. 4.3) increase.

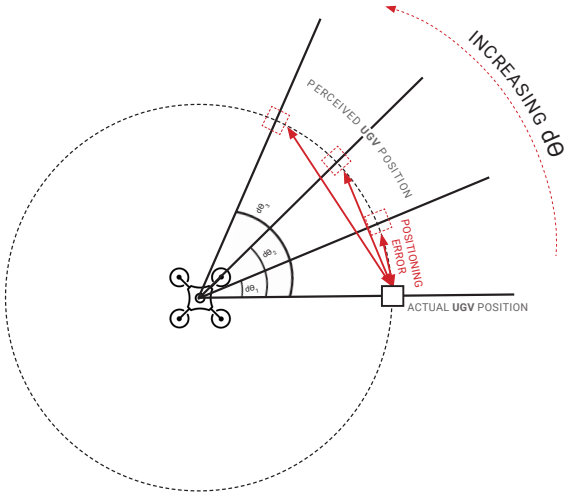

Figure 4.2: Positioning error increasing the misalignment

The estimated relative position when $d\theta$ is different from zero $p_{rel,d\theta\neq 0}$ corresponds to the rotation with respect to the z axis of the real relative position $p_{rel,d\theta=0}$, then:

$$p_{rel,d\theta\neq 0} = \begin{bmatrix} \cos d\theta & -\sin d\theta & 0 \\ \sin d\theta & \cos d\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} p_{rel,d\theta=0}$$

The error associated to the relative position given by dθ is [27]:

$$e_{pos,d\theta} = \|p_{rel,d\theta\neq 0} - p_{rel,d\theta=0}\|_2 = 2\|p_{rel,d\theta=0}\| \left| sin\left(\frac{d\theta}{2}\right)\right|$$

Therefore,the farther the drone is from the rover the greater the associated error:

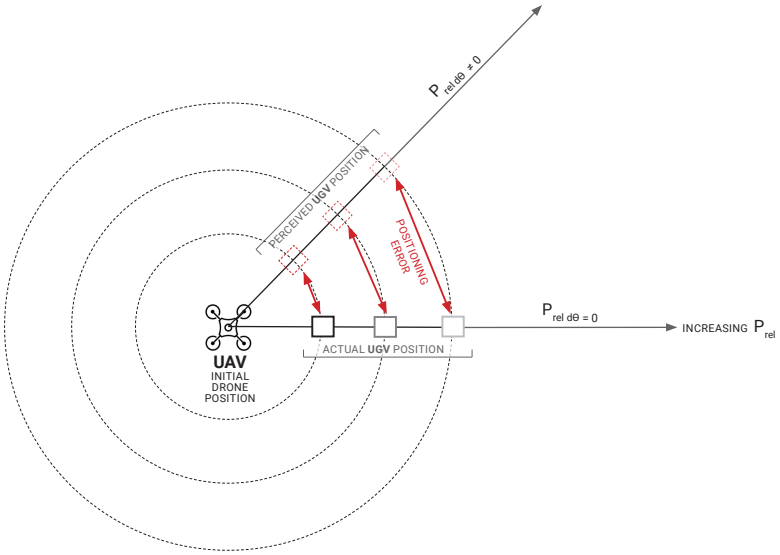

Figure 4.3: Positioning error increasing the distance

It is trivial to deduce that the error in the relative position also increases as the speed of the rover increases. In figure 4.4 the concept is clearly illustrated. The drone will move along the trajectory affected by the misalignment error and after a time-step, the drone following a faster rover will travel more space than another drone following a slower rover. The time interval depends on how often the prediction phase of the kalman filter takes place. Consequently, the error associated with the drone following the fastest platform will be greater.
In the previous work, the main reason why the drone was unable to land on the rover, when the speed was higher than 1 m/s, is to be traced back to the graph illustrated above. This created strong wobbles in the drone and abrupt changes of direction. Furthermore, in the previous work, when the angle was greater than 20 degrees, even at a speed of 1 m/s, strong oscillations were recorded, reaching positioning errors of almost 4 meters

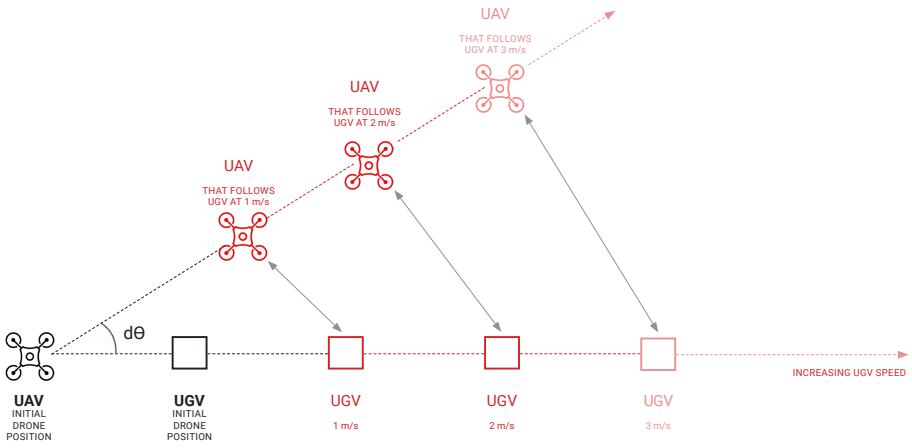

Figure 4.4: Positioning error increasing the rover speed

4.3.2 Rotation with Apriltag orientation

In this work, the use of the apriltag allows for further information on the position of the rover. This estimate merged with the information from the UWB allows the drone to make a very precise landing. The apriltag also allows you to obtain a very precise estimate of the rover's orientation and therefore, allows to combine and replace the noisy information coming from the rover's compass. This leads to the introduction of a new state in the estimation system given by the difference in the orientation provided by the apriltag and the one estimated through the compass. An offset is then generated which, added to the angle at the exit from the compass, corrects the error. In the estimation system all the calculations are referred to the NED frame. Consequently, the orientation must also be referred to this reference system. Since the topic */ImageRaw*, exclusively gives information on the orientation of the apriltag with respect to the reference system of the drone, this information must be rotated in the NED reference system. Then:

$$R_{UGV}^{NED} = R_{UAV}^{NED} R_{UGV}^{UAV}$$

The orientation of the drone with respect to the NED reference system comes from the topic */VehicleAttitude*. These two successive rotations allow to obtain

the matrix that provides the orientation of the rover. Consequently, the noisy orientation information coming from the compass can be discarded.

Chapter 5

System control algorithm

5.1 System state machine

The control system is structured as a state machine and consists of four phases: takeoff, chase, descent and landing. In the first phase, the drone takes off to a certain height, called $HovHeight$. During this phase, the drone is controlled in position by passing the desired altitude $z_{UAV} = HovHeight$ to the autopilot. The other phases are also controlled via offboard control, i.e. by passing setpoints to the autopilot. As soon as the drone has reached the desired height, it switches to the Chase phase. In this phase, the drone is controlled in speed via a PID controller. If the relative position and speed fall below a certain value, the Descent phase begins. The drone, then, begins its descent towards the rover. As soon as the distance to the platform along z decreases below the $LandHeight$ threshold, the drone shuts down its motors and descends onto the platform. However, at higher rover speeds, the drone continues its descent until it is on the platform, to avoid overturning due to its free fall onto the rover.

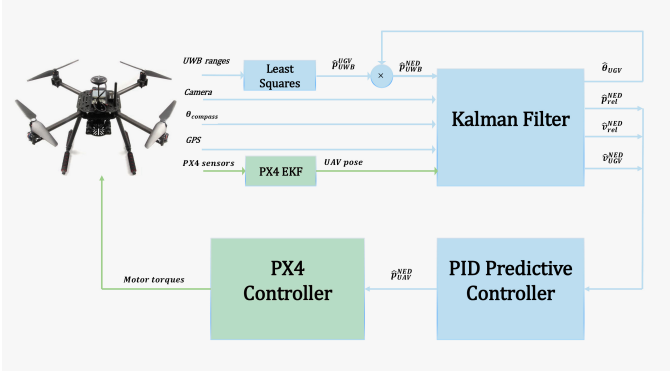

Figure 5.1: Overall system architecture

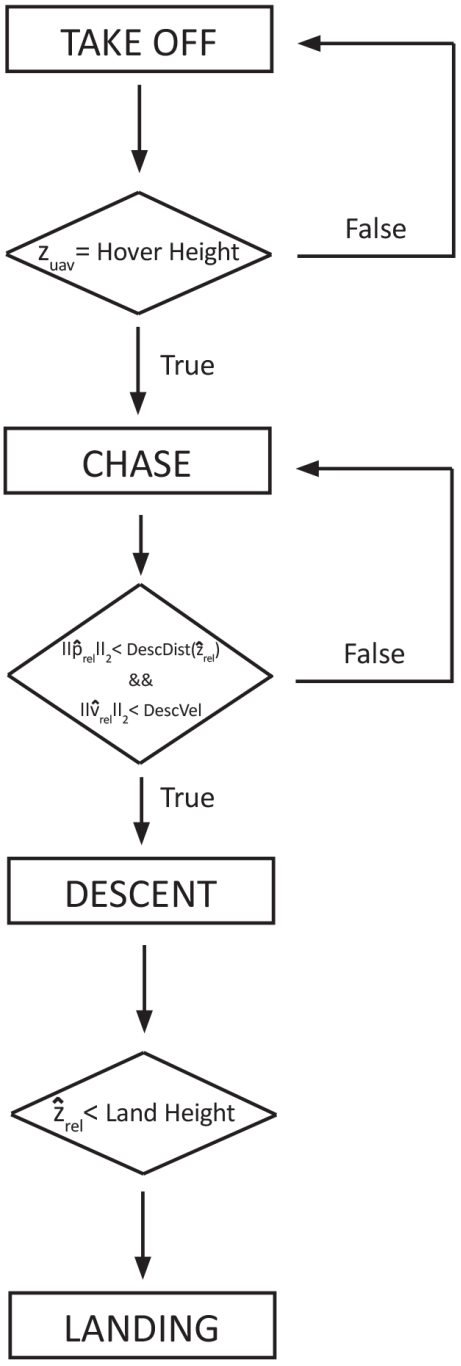

Figure 5.2: Control state machine

5.1.1 Overall control algorithm

During the four phases, the drone is controlled by passing setpoints to the autopilot via offboard control. The exception is the last phase, the landing phase, which is controlled directly by the autopilot for safety reasons. Instead, in the chase phase, the drone is controlled in speed along x and y. On the other hand, along z, position control takes place by keeping z = HovHeight. Speed values are generated via a PID controller or a simple P controller, depending on the relative distance. When the drone is far from the rover a proportional control is used.

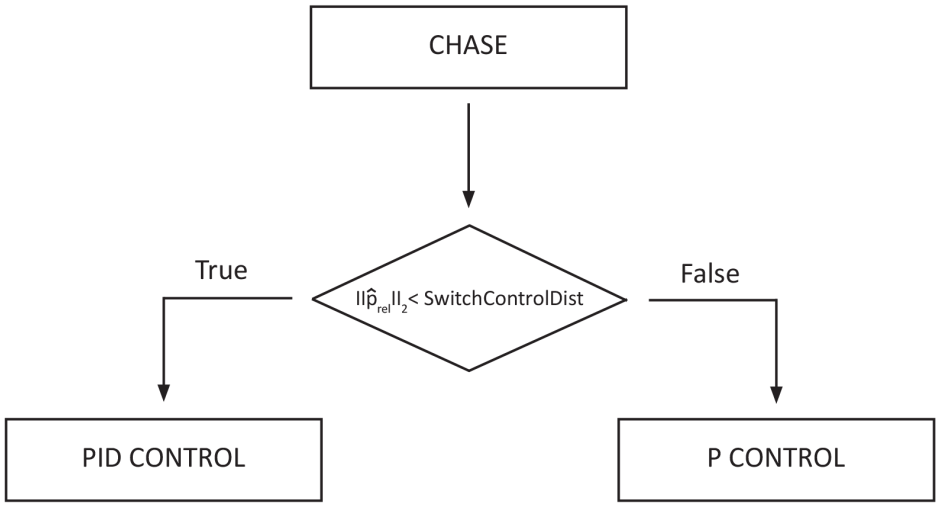

Figure 5.3: Control system

The setpoints passed to the autopilot at this stage are as follows:

$$\hat{\dot{x}}_{UAV,k} = P \cdot \hat{x}_{rel,k}$$

$$\hat{\dot{y}}_{UAV,k} = P \cdot \hat{y}_{rel,k}$$

In the vicinity of the rover, derivative integrative components are of crucial importance in order to ensure fast convergence and small overshoot. The setpoints passed to the autopilot at this stage are as follows:

$$\hat{x}_{UAV,k} = P \cdot \hat{x}_{rel,k} + I \cdot \sum_{0}^{k} \hat{x}_{rel,k} \cdot dt + D \cdot \hat{\dot{x}}_{rel,k}$$

$$\hat{y}_{UAV,k} = P \cdot \hat{y}_{rel,k} + I \cdot \sum_{0}^{k} \hat{y}_{rel,k} \cdot dt + D \cdot \hat{\dot{y}}_{rel,k}$$

5.2 Predictive control algorithm

At high speeds, a predictive control algorithm is necessary. This is because the landing phase takes place in vertical mode, which is managed exclusively by the autopilot for safety reasons. However, vertical descent means that the drone shuts down its motors at a certain height from the rover. The free fall of the drone means that the rover is often missed when it goes at high speed. From the turning off of the motors to the complete landing of the drone, the rover moves by an amount proportional to its speed. The rover's displacement can be predicted since we have an estimate of the rover's speed. The estimated speed of the rover multiplied by the time it takes the drone to complete its descent, the *prediction time*, gives us the prediction of the future displacement of the rover. By adding this value to the relative position, we cause the drone to land at the predicted position, without missing the platform as the speed increases. Then:

$$\hat{x}_{pred,k} = \hat{x}_{rel,k} + T_{pred} \cdot \hat{\dot{x}}_{UGV,k}$$

$$\hat{y}_{pred,k} = \hat{y}_{rel,k} + T_{pred} \cdot \hat{\dot{y}}_{UGV,k}$$

where:

$$T_{pred} = \frac{z_{Land,UAV}}{\dot{z}_{Land,UAV}}$$

Now, proportional control becomes:

$$\hat{\dot{x}}_{UAV,k} = P \cdot \hat{x}_{pred,k}$$

$$\hat{\dot{y}}_{UAV,k} = P \cdot \hat{y}_{pred,k}$$

Similarly, the PID control becomes:

$$\hat{x}_{UAV,k} = P \cdot \hat{x}_{pred,k} + I \cdot \sum_{0}^{k} \hat{x}_{pred,k} \cdot dt + D \cdot \hat{\dot{x}}_{pred,k}$$

$$\hat{y}_{UAV,k} = P \cdot \hat{y}_{pred,k} + I \cdot \sum_{0}^{k} \hat{y}_{pred,k} \cdot dt + D \cdot \hat{\dot{y}}_{pred,k}$$

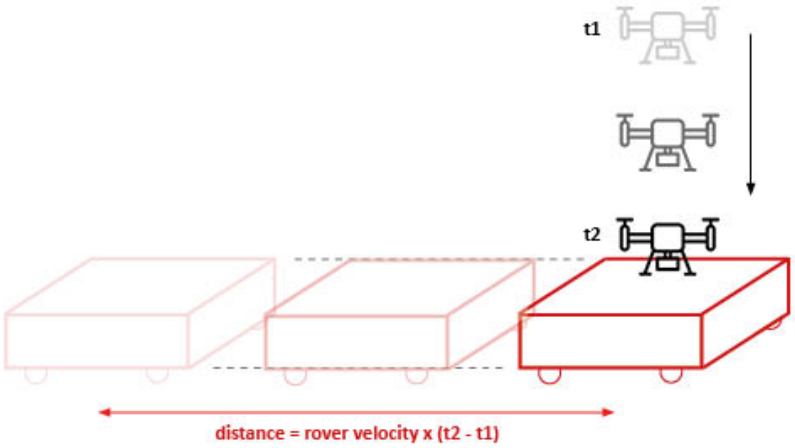

Figure 5.4: Predictive control

5.3 Landing at high speed

In chapter 4.3, I have shown how one of the main problems for a high-speed landing is the displacement of the rover when the drone shuts down its motors. To this problem, two others must be added. One of these is the incorrect estimation of the rover's orientation, which we discussed extensively in Chapter 3.2. Without the correction of the erroneous compass output thanks to the apriltag, it would be impossible to land at high speeds.
The third problem relates to the stability of the drone once it has landed. At high speeds the drone suffers from high vibrations on the platform. In addition, a purely vertical descent from a height of $30cm$ from the rover would cause the rover to overturn, unless it is equipped with magnetic legs that stick it to the platform. To solve this problem, the vertical descent must take place as late as possible.

5.3.1 Free fall drone problem

A convergent descent to the platform is required so that the drone has a chance to stabilise on the platform. The presence of a horizontal velocity component in the direction of movement of the rover prevents the drone from overturning. This is required for a speed above 10 km/h. Instead of free-falling, the drone converges on the platform, trying to stabilise itself on it. Once the drone is almost stabilised on the platform the LAND command can be activated. In the next section, we will examine the autopilot's modus operandi in handling the Land command and the problems associated with it.

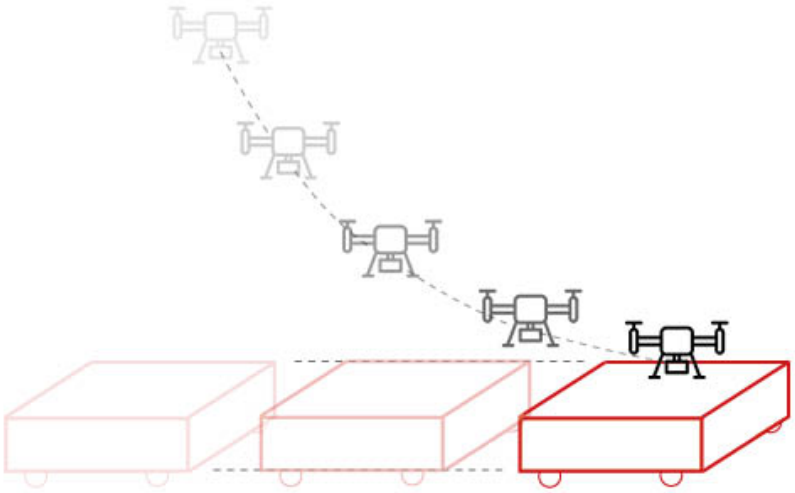

Figure 5.5: Soft landing

5.3.2 Land detector

When the Land flight mode is engaged, the drone will land at the position where it is at that time. If the landed conditions are satisfied, the vehicle will disarm after a number of seconds that can be set in the PX4 parameter COM_DISARM_LAND [28]. The land-detector is the main actor responsible for assessing the conditions that determine whether the vehicle has actually landed. This evaluation is based

on the examination of three states with more stringent conditions passing from one state to the next. Before examining the three states, knowledge of five parameters of the PX4 autopilot (in QGrounControl) is necessary:

- *MPC_THR_HOVER*: it represents the hover thrust, i.e the vertical thrust needed to hover
- *MPC_THR_MIN*: it represents the minimum possible thrust. It is required for a controlled descent.
- *LNDMC_Z_VEL_MAX*: it is the maximum vertical speed allowed during the landing phase
- *LNDMC_XY_VEL_MAX*: it is the maximum horizontal velocity allowed during the landing phase
- *LNDMC_ROT_MAX*: it is the maximum angular speed for each motor during the landing phase

The table shows the possible values that the above parameters can assume and the default values that PX4 recommends.

	Min	**Max**	**Default**
MPC_THR_HOVER	0.1	0.8	0.5
MPC_THR_MIN	0.05	1.0	0.12
LNDMC_Z_VEL_MAX			0.25 m/s
LNDMC_XY_VEL_MAX			1.5 m/s
LNDMC_ROT_MAX			20 deg/s

Table 5.1: Land-Detector Parameters [29]

Let us now examine the three states that must be verified for the landed condition to be verified. Each of these following states must be true for a number of seconds:

- **Ground Contact**: For this state to be passed, the following three conditions must be true for at least 0.35 seconds:
  1. vertical movement lower than LNDMC_Z_VEL_MAX
  2. horizontal movement lower than LNDMC_XY_VEL_MAX
  3. has thrust lower than $MPC\_THR\_MIN + (MPC\_THR\_HOVER - MPC\_THR\_MIN) \cdot 0.3$

- **Maybe Landed**: For this state to be passed, the following three conditions must be true for at least 0.25 seconds:
  1. Ground contact conditions are true
  2. angular velocity is lower than LNDMC_ROT_MAX
  3. has thrust lower than $MPC\_THR\_MIN + (MPC\_THR\_HOVER - MPC\_THR\_MIN) \cdot 0.1$
- **Landed**: For this state to be passed, the following condition must be true for at least 0.3 seconds:
  1. Maybe Landed conditions are true

5.3.3 Problems to disarm

At high speeds, the drone has problems to disarm. These problems are to be found in some of the conditions mentioned above that are not realised. Clearly, at high speeds, vibrations increase dramatically. This causes that the condition of the vertical movement below a certain threshold not to be fulfilled. As a result, the land detector will not disarm the vehicle, even if it is already stable on the platform. This problem could be solved by not using the LAND command as the landing mode. However, for safety reasons, the LAND command is strictly recommended because the autopilot is able to handle the landing in extreme safety for the space around it and for the integrity of the drone itself. The use of magnetic legs, capable of attaching themselves to the platform, could certainly help to achieve this end.

Chapter 6

Simulation

6.1 Gazebo models and plugin

PX4 allows our algorithm to be simulated in Gazebo. This is possible thanks to the SITL package, which allows running the flight stack in the simulated enviroment using ROS2. Moreover, the power of this package also lies in the fact that the same code can be used directly in the tests without any change.

6.1.1 UAV and UGV model

The Gazebo drone model chosen for the simulation is the 3DR IRIS, because it is the most similar to the real drone. This drone was customised for our case through the integration of the camera, the UWB tag and the range sensor.

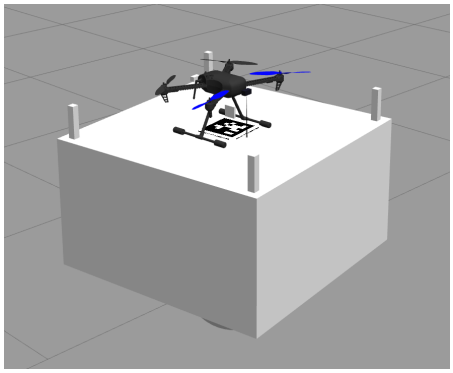

Figure 6.1: UAV and UGV model

The chosen rover is a 3-wheel vehicle with differential drive. Its dimensions are $1m \times 1m \times 0.5m$. Four anchors are fixed to its vertices and an Apriltag is attached

to its surface. In addition, a compass and a GPS device have been inserted.

6.1.2 UWB and GPS plugin

In our system there are four anchors and a tag. The four anchors are fixed to the vertices of the rover. The tag is mounted on the UAV leg. The UWB plugin does not deal with tags and anchors as two different sensors. Both publish data on a common topic. If an anchor acts as a tag, it subscribes to that topic and calculates the distances from each anchor.
Both the drone and the rover are equipped with GPS plugins. Both return their pose in WGS84 geodetic coordinate system. This information has to be rotated in the NED reference frame before they are used. It was decided to choose the point where the rover is turned on as the origin of the Cartesian reference system.

6.2 Results

This section illustrates the results and shows the robustness of this algorithm to possible erroneous information from the sensors. For this purpose, the system was stressed by increasing the noise associated with each sensor.

The figures 6.2 and 6.3 plot the complete flight from the takeoff phase to the final landing, in the case where the rover moves in one direction only or randomly. The red and blue lines show the relative drone-rover potion during the three phases. The change in relative position during the *takeoff* phase is due to the movement of the rover. The black line shows the altitude of the drone in the three regions. In particular, in the descent phase a stepped pattern is plotted. This is due to the fact that the drone checks in each time-step whether certain conditions are realized. If so, the drone can continue the descent.

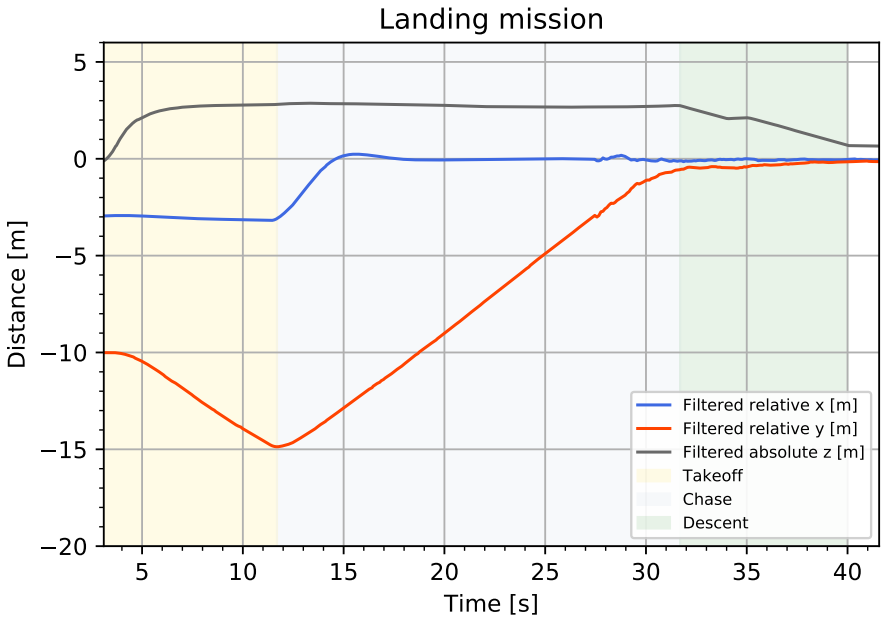

Figure 6.2: Simulated autonomous landing of a UAV on a linear moving UGV

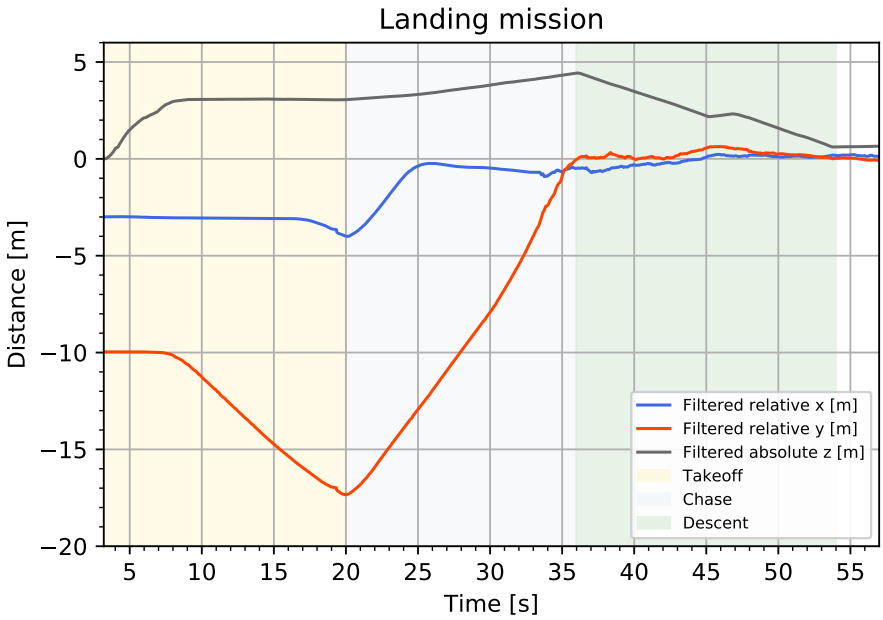

Figure 6.3: Simulated autonomous landing of a UAV on a random moving UGV

The other graphs show the robustness of the algorithm to a possible misalignment of the drone and rover compasses. To simulate this error, an *offset error* was added to the angle coming from the compass. In the first graph the estimated rover orientation, the information about the rover orientation coming from the compass and apriltag, and the true rover orientation are compared. In the second graph, the estimated relative position between drone and rover is plotted, focusing on the most crucial part of the flight where camera and UWB are used. This graph compares the position estimated by the Kalman filter and the position estimated by the Least Square in the case where the system would use the compass as the sole information about the rover orientation.The use of the Apriltag orientation within the Kalman Filter corrects the compass output. In the last graph the positioning error is evaluated, showing comparable values for the 3 cases.

Figure 6.4 shows the angle correction in the ideal case where there is no misalignment between drone and rover compasses. In this case the output orientation of the compass and apriltag are comparable. Figure 6.5 shows how the position estimate would coincide with the LS information in the case where the system would use the compass as the sole information about the rover orientation.

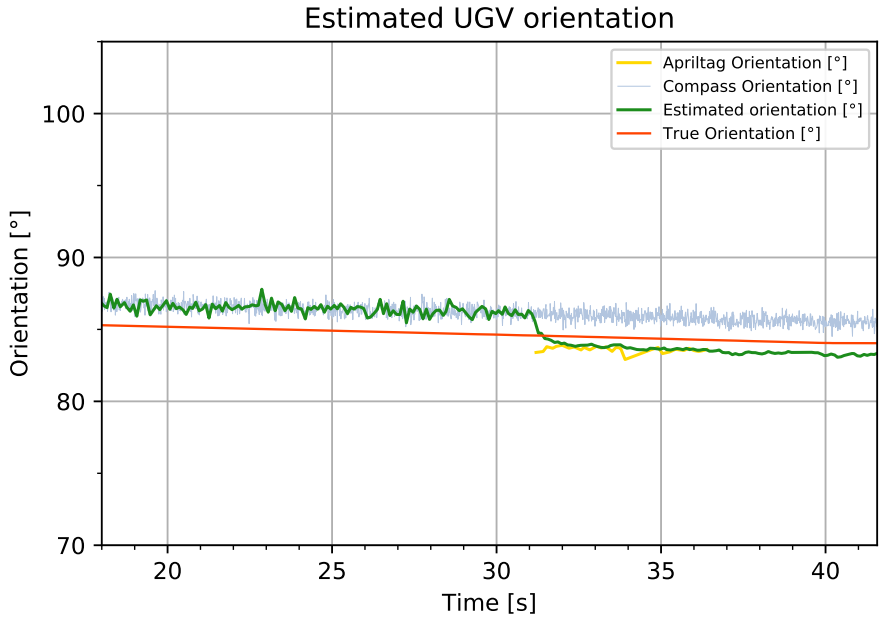

Figure 6.4: Estimated UGV orientation when $d\theta = 0°$

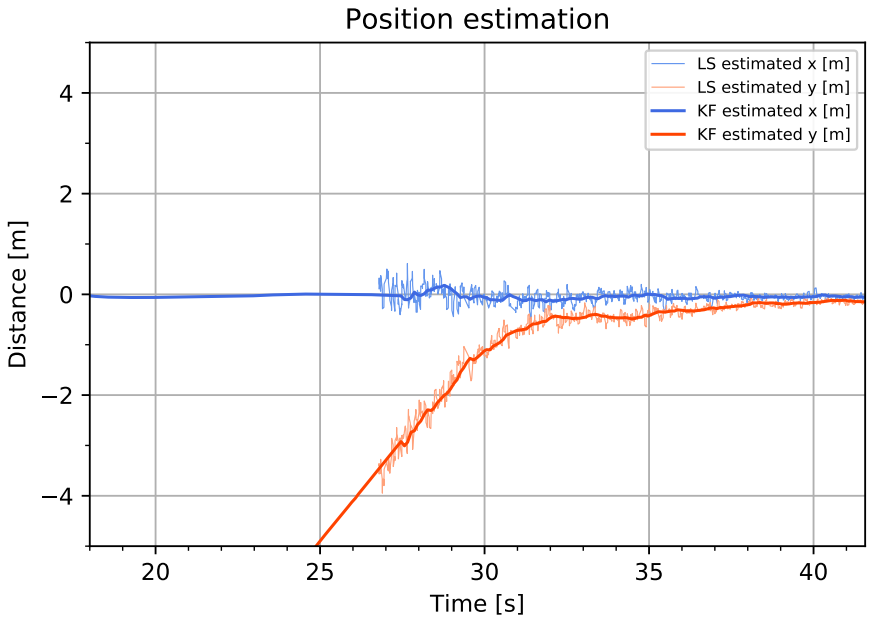

Figure 6.5: Correct LS estimation when $d\theta = 0°$

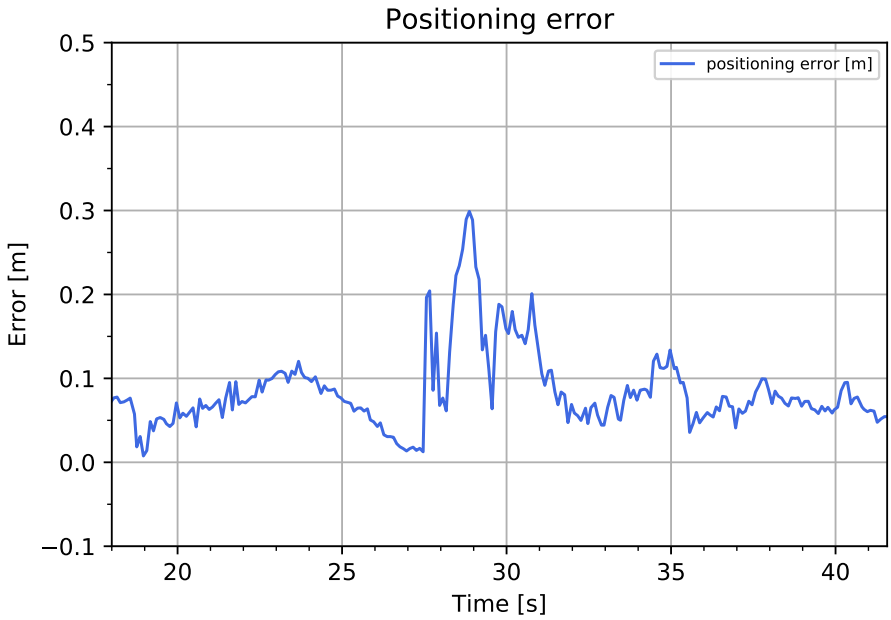

Figure 6.6: Positioning error when $d\theta = 0°$

The situation turns out to be different if an offset error is added to the compass mounted on the rover. In figure 6.8 and 6.11, we can see how the Least Square estimate diverges from the estimated position if only the information of the compass would be used as the rotation angle. The use of the apriltag orientation corrects the erroneous information from the compass. This leads to a corrected position estimate with a total positioning error of less than $0.3m$ and comparable in all 3 cases.

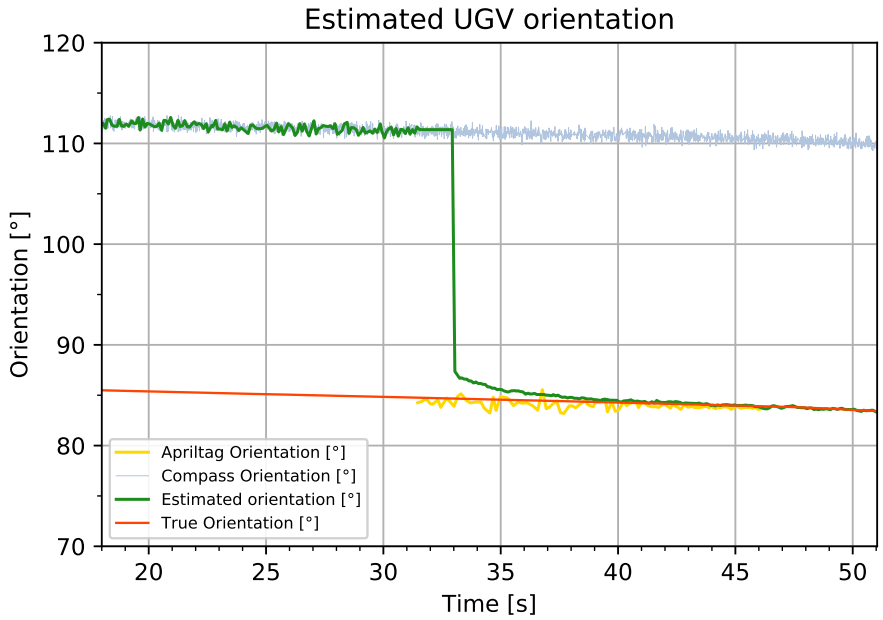

Figure 6.7: Estimated UGV orientation when $d\theta = 25°$

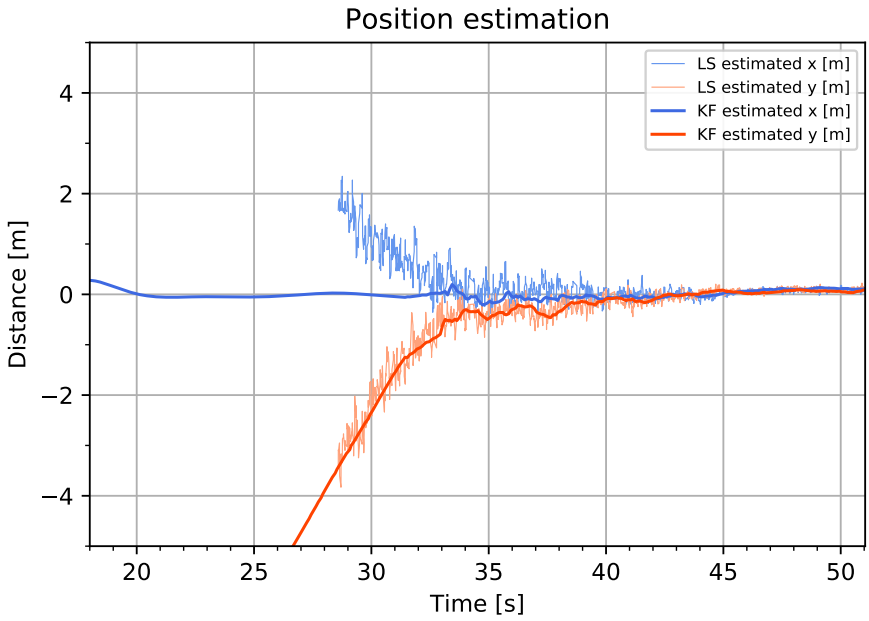

Figure 6.8: Incorrect LS estimation when $d\theta = 25°$

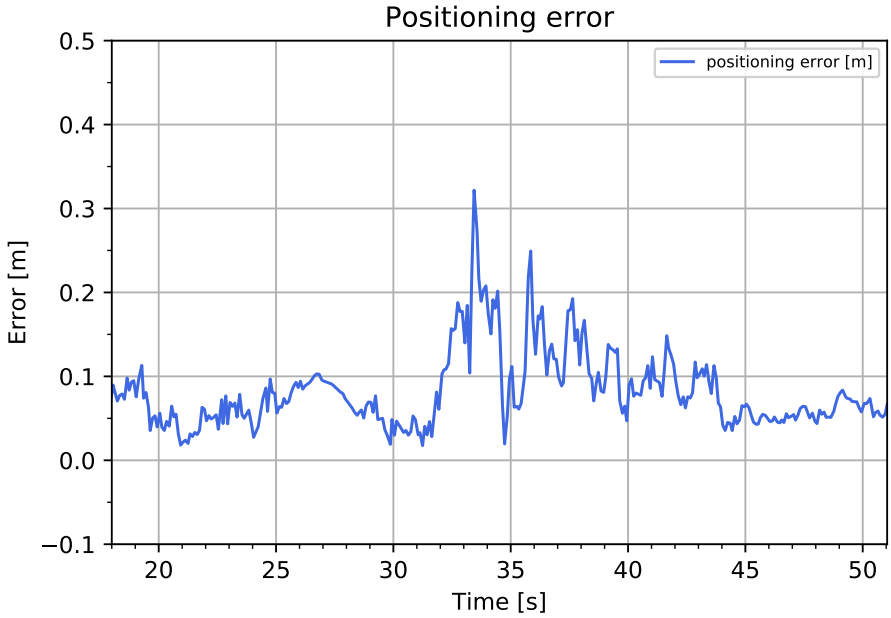

Figure 6.9: Positioning error when $d\theta = 25°$

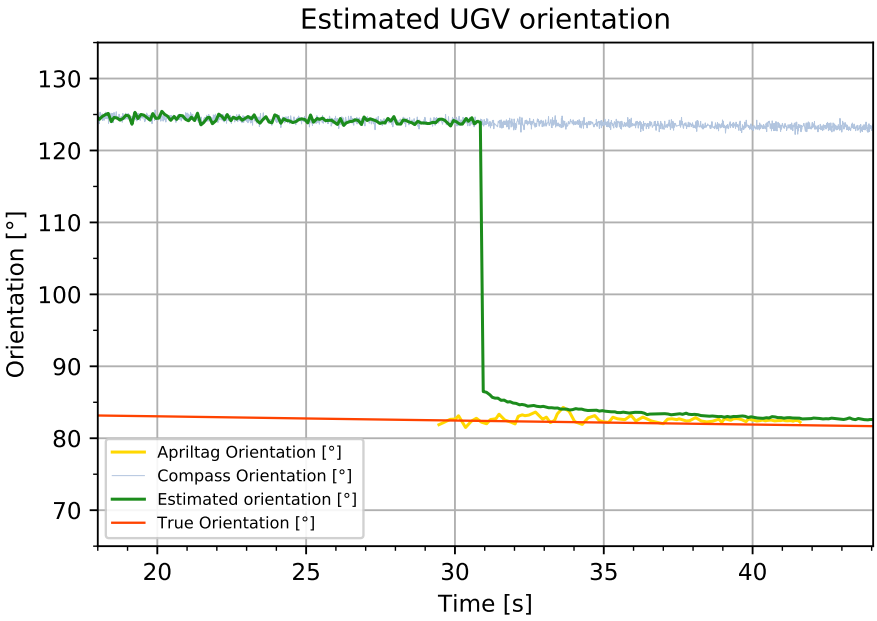

Figure 6.10: Estimated UGV orientation when $d\theta = 40°$

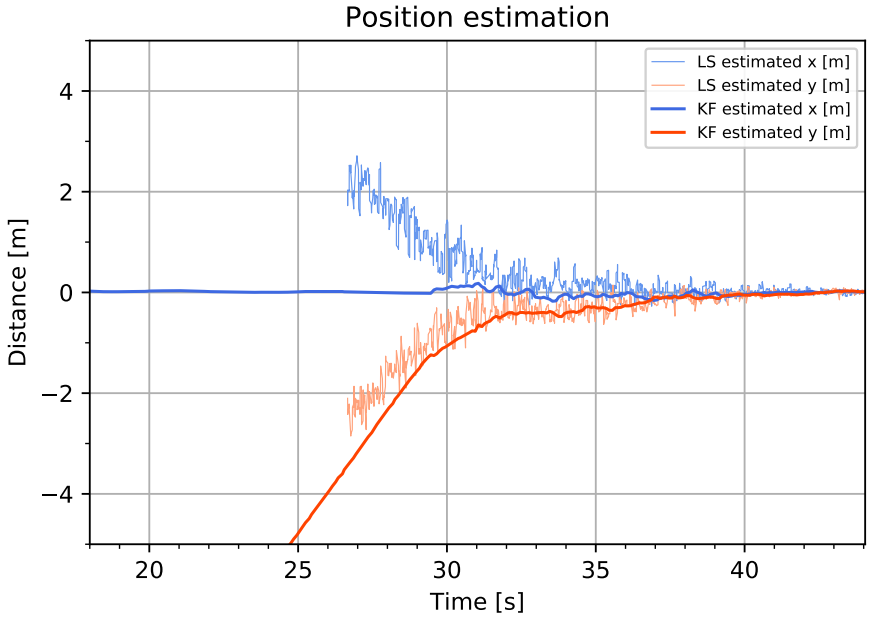

Figure 6.11: Incorret LS estimation when $d\theta = 40°$

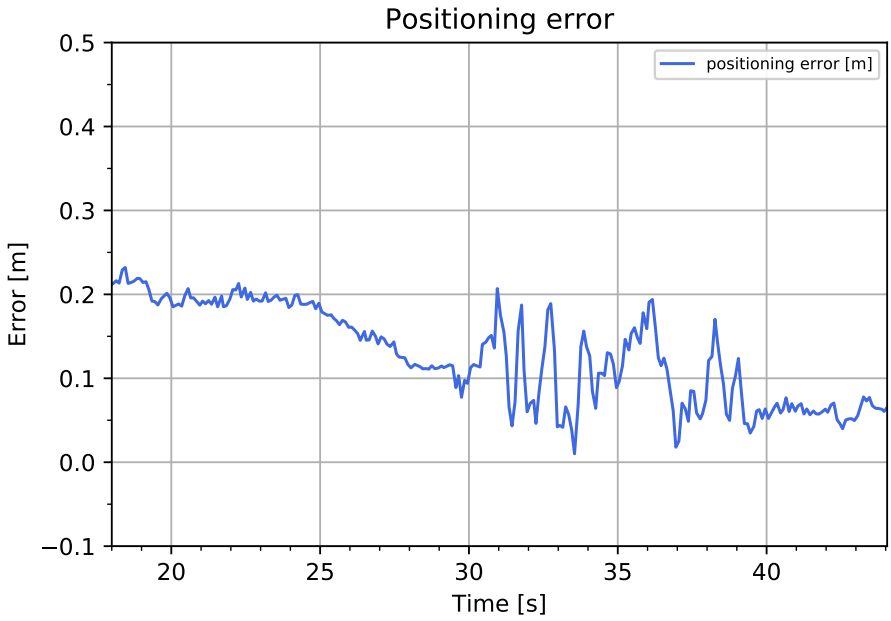

Figure 6.12: Positioning error when $d\theta = 40°$

Chapter 7

Experimental analysis

7.1 Drone setup

The drone used is a Holybro X500. It has a chassis made entirely of carbon fibre, with a series of mounting holes on its surface. The main components of this drone are:

- Pixhawk 4 autopilot
- Power Management PM07
- Pixhawk4 GPS
- Motors - 2216 KV880
- BLHeli S ESC 20A
- Propeller 1045
- 433MHz Telemetry Radio / 915MHz Telemetry Radio
- Power and Radio Cables
- Battery Straps

A Raspberry Pi 4 is mounted on the platform to run the estimation and control algorithms, sending setpoints to the PX4 autopilt, via UART. In addition, it is possible to send commands to the UAV by connecting the raspberry to the wifi.

Figure 7.1: Holybro X500 UAV[30]

7.2 Rover setup

Figure 7.2: Husky UGV [31]

The rover used is the Husky UGV by Cleoorpath robotics. It provides an API support for ROS. It has a dimensions of $990 \times 670 \times 390mm$ and a weight of $50kg$. The maximum weight supported is $75kg$. The speed limit is set at $1.0m/s$. It is also equipped with GPS device, compass and four anchors attached to its vertices. Orientation data are published on a ROS2 topic via wifi.

7.3 Instrumentations

7.3.1 Camera

The camera used for the tests is a Raspberry Pi cam, which is compatible with all Rasberry Pi boards. This camera provides a low noise image, with resolution of 8 megapixels. It has dimensions $24 \times 25 \times 9mm$ and has a weight of $3g$. The CSI connector allows for high data rate. The max image transfer rate is $30fps$ for a $1080p$ image and $60fps$ for a $720p$ one.[32]

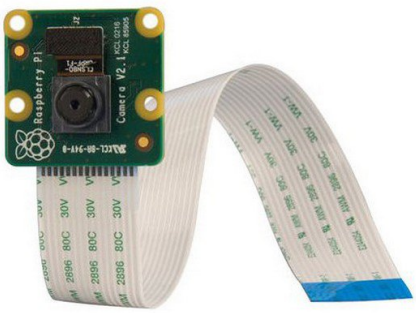

Figure 7.3: Raspberry Pi Camera Module [33]

7.3.2 UWB sensors

The four anchors, attached to the rover, and the tag, fixed to the drone leg, are provided by Decawave EVB1000 board. It is connected to the microcontroller via USB. It shows a high precise position below than 5 cm and are able to have a coverage of about 150m in LoS condidions. It has a weight of 40g and dimensions $70 \times 120 \times 13mm$.

Figure 7.4: Decawave EVB1000 board [34]

7.3.3 GPS sensors

The rover is equipped with Piksi RTK GPS that allows centimeter accurate positioning. The GPS mounted on the drone is a H-RTK F9P.It is based on

a Ublox F9P module. The indicator LEDs let you know when the GPS receiver starts to acquire information. The TTFF (time to first fix) may vary depending on the availability of satellites in the area where you turn on the drone. The cold start is less than 26 seconds. In addition, H-RTK F9P uses an IST8310 compass, which is able to make the flight more stable.

Figure 7.5: H-RTK F9P GNSS [35]

7.4 Results

The experimental tests show the correct working of the autonomous algorithm in three scenarios. First, the landing of the drone on the stationary platform was tested. The drone was placed at a long distance from the rover in order to test the proper operation of the state machine. In figure 7.6 the relative drone-rover distance during the entire mission is plotted. The drone take off at an altitude of $3m$. During the first part of the chase phase, the accuracy of the RTK GPSs allows the drone to approach the rover precisely without oscillation, reporting an accuracy of less than $10cm$. The fusion of UWB and Camera allows to achieve a landing accuracy of about $5cm$.

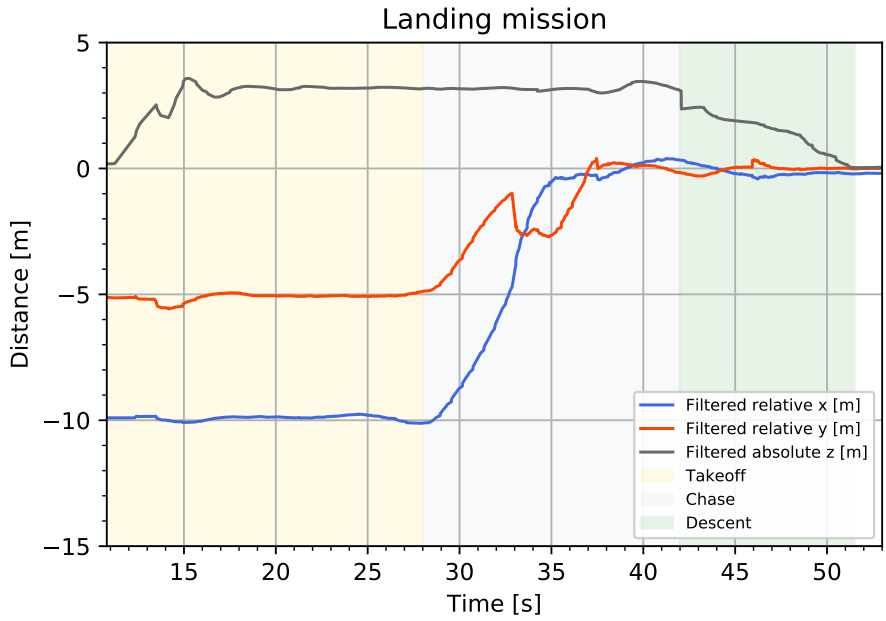

Figure 7.6: Autonomous landing of a UAV on a stationary UGV

In the flight presented in the following figures, the rover moves at different speeds. However, the maximum speed at which the system was tested is $1m/s$, as this represents the speed limit of the Husky UGV. In figure 7.7 the rover is moving at speed with a linear motion. During the takeoff phase, the change in relative position is due to the movement of the rover. The linear trend of the curve suggests that rover moves in a single direction. The change in the slope of the two curves is due to the beginning of the chase phase. In contrast, in the figures 7.8 and 7.9 the oscillatory trend of the relative position suggests that the rover moves in a random way. To land on the platform the use of prediction parameters is of vital importance. A *prediction time* factor of 0.2 s was considered. Considering a maximum rover speed of 1 m/s, this corresponds to a landing of the drone at maximum $20cm$ forward, avoiding missing the platform. The adoption of a predictive control system allows accurate and fast tracking and landing on the moving target, making it possible to reach a landing accuracy of about $5cm$.

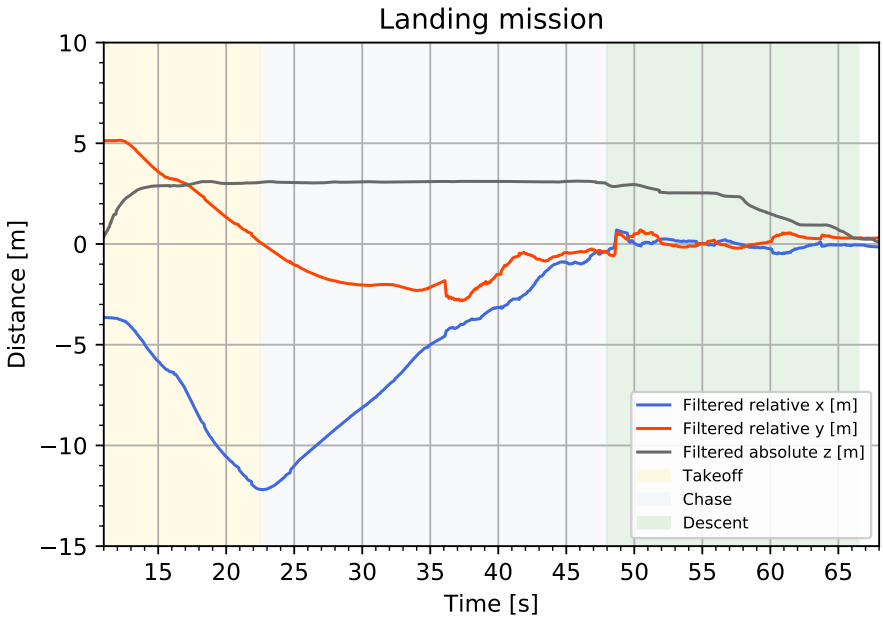

Figure 7.7: Autonomous landing of a UAV on a linear moving UGV

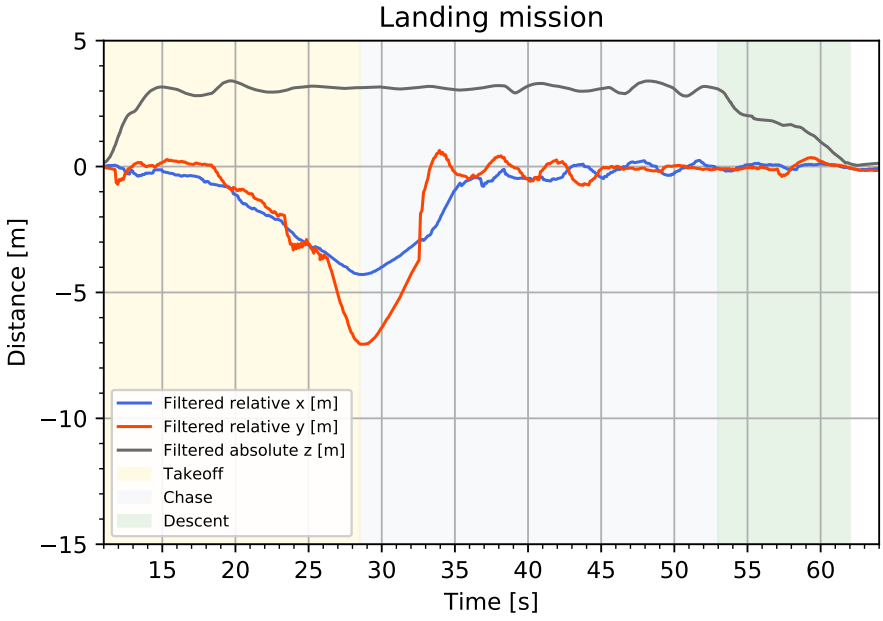

Figure 7.8: Autonomous landing of a UAV on a random moving UGV

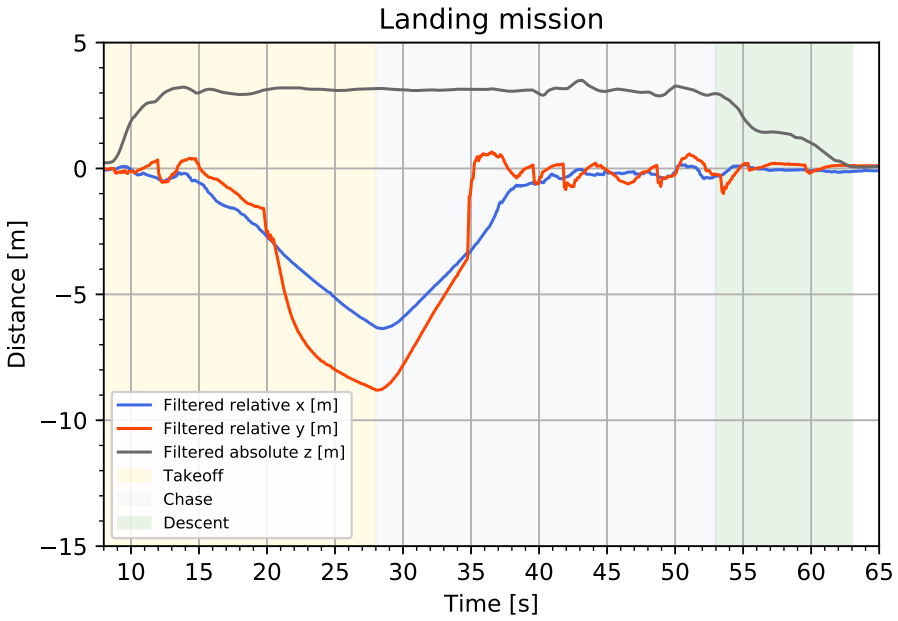

Figure 7.9: Autonomous landing of a UAV on a random moving UGV

Chapter 8

Conclusions

This work implements an autonomous algorithm that allows for the landing of a UAV onto a vehicle that is moving at high speed, through the use of different sensors, chosen depending on the relative drone-rover position. The large number of sensors used ensures a high level of redundancy in positioning, while offering greater stability to the drone during the flight.The Kalman filter manages the information coming from the various sensors and generates an estimate of the relative position and relative speed. These are then passed to a PID speed controller that allows accurate and fast tracking and landing on the moving target. By mounting a camera on the drone that computes the orientation of the apriltag with extreme precision, the noisy information of the compass can be corrected. The adoption of a predictive control system, the introduction of new sensors and the correction of the misalignment between the compasses of the drone and rover made it possible to reach landing speeds above 30 km/h. The simulated and experimental results validated the correct working of the algorithm, reporting a landing accuracy of less than 10 cm. During the tests, it was possible to test the drone for a maximum rover speed of 1 m/s, which corresponds to the speed limit of the Husky rover. At high speeds, the stability of the drone on the platform could clearly be compromised by the high vibrations. Therefore, blocking systems must be developed for the integrity of the drone and for safety reasons. A possible solution could be the use of a magnetic catcher [36] on the platform, capable of making magnetic contact with the drone.

Bibliography

[1] *Flight Modes*. URL: https://docs.px4.io/v1.12/en/getting_started/px4_basic_concepts.html (cit. on p. 3).

[2] *PX4 Architectural Overview*. URL: https://docs.px4.io/main/en/concept/architecture.html#middleware (cit. on p. 4).

[3] *PX4-ROS 2 Bridge*. URL: https://docs.px4.io/v1.12/en/ros/ros2_comm.html (cit. on p. 4).

[4] *Controller Diagrams*. URL: https://docs.px4.io/main/en/flight_stack/controller_diagrams.html (cit. on p. 5).

[5] *Offboard Mode*. URL: https://docs.px4.io/v1.12/en/flight_modes/offboard.html (cit. on p. 5).

[6] *Reference Frames and ROS*. URL: https://docs.px4.io/v1.12/en/ros/external_position_estimation.html (cit. on p. 6).

[7] Hameedah Hasan, Mohamed Hussein, Shaharil mad saad, and Mohd Azuwan Mat Dzahir. «An Overview of Local Positioning System: Technologies, Techniques and Applications». In: *International Journal of Engineering and Technology(UAE)* 7 (Aug. 2018), pp. 1–5. DOI: 10.14419/ijet.v7i3.25.17459 (cit. on p. 7).

[8] *Time Difference of Arrival (TDOA)*. URL: https://dl.cdn-anritsu.com/en-us/test-measurement/files/Application-Notes/Application-Note/11410-01009D.pdf (cit. on p. 8).

[9] *Angle-of-Arrival*. URL: https://www.sciencedirect.com/topics/engineering/angle-of-arrival (cit. on p. 8).

[10] Peng Wang, Zhiyang Liu, Xiaotong Zhang, Liyuan Xu, Jie He, and Yadong Wan. «Wideband Signal Based Near-Field Electromagnetic Ranging for Indoor Localizatio». In: *Proceedings of the 2018 International Conference on Advanced Control, Automation and Artificial Intelligence (ACAAI 2018)*. Atlantis Press, 2018/03, pp. 243–247. ISBN: 978-94-6252-483-5. DOI: https://doi.org/10.2991/acaai-18.2018.57. URL: https://doi.org/10.2991/acaai-18.2018.57 (cit. on p. 8).

[11] Davide Dardari, Marco Luise, and Emanuela Falletti. *Satellite and Terrestrial Radio Positioning Techniques: A Signal Processing Perspective*. eng. 1st ed. Saint Louis: Elsevier Science Technology, 2011. ISBN: 0123820847 (cit. on pp. 9–11).

[12] *Physics of the Ionosphere and Troposphere*. URL: https://web.archive.org/web/20140522193825/http://www.navipedia.net/index.php/Earth_Sciences#Troposphere_Monitoring (cit. on p. 13).

[13] *Multipath*. URL: https://web.archive.org/web/20120430025036/http://www.navipedia.net/index.php/Multipath (cit. on p. 13).

[14] *New Empirically Derived Solar Radiation Pressure Model for Global Positioning System Satellites*. URL: https://ipnpr.jpl.nasa.gov/progress_report/42-159/159I.pdf (cit. on p. 13).

[15] *Error analysis for the Global Positioning System*. URL: https://en.wikipedia.org/wiki/Error_analysis_for_the_Global_Positioning_System (cit. on pp. 14, 15).

[16] *Differential GPS*. URL: https://www.oc.nps.edu/oc2902w/gps/dgpsnote.html (cit. on p. 15).

[17] *GPS – how the Global Positioning System works*. URL: https://www.umweltanalysen.com/en/gps/ (cit. on pp. 16, 17).

[18] *Real-Time Kinematic (RTK)*. URL: https://novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/real-time-kinematic-rtk (cit. on p. 18).

[19] *Inertial Navigation Systems and Its Practical Applications*. URL: https://www.intechopen.com/chapters/39779 (cit. on pp. 18, 19).

[20] *Federal Communications Commission*. URL: https://transition.fcc.gov/Bureaus/Engineering_Technology/Orders/2002/fcc02048.pdf (cit. on p. 19).

[21] Marko Malajner; Peter Planinšič; Dušan Gleich. «UWB ranging accuracy». In: (). URL: https://ieeexplore-ieee-org.ezproxy.biblio.polito.it/document/7314177 (cit. on p. 20).

[22] *AprilTag with Python*. URL: https://pyimagesearch.com/2020/11/02/apriltag-with-python/ (cit. on p. 21).

[23] *AprilTag*. URL: https://april.eecs.umich.edu/software/apriltag (cit. on p. 22).

[24] Hameedah Hasan, Mohamed Hussein, Shaharil mad saad, and Mohd Azuwan Mat Dzahir. «An Overview of Local Positioning System: Technologies, Techniques and Applications». In: *International Journal of Engineering and Technology(UAE)* 7 (Aug. 2018), pp. 1–5. DOI: 10.14419/ijet.v7i3.25.17459 (cit. on p. 22).

[25] Wikipedia contributors. *Kalman filter — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=1119129240. 2022 (cit. on p. 26).

[26] *FilterPy 1.4.4 documentation*. URL: https://filterpy.readthedocs.io/en/latest/common/common.html (cit. on p. 30).

[27] G. Scarati. «UAV precise ATOL techniques using UWB technology». In: (). URL: https://webthesis.biblio.polito.it/secure/21180/1/tesi.pdf (cit. on p. 37).

[28] *Land Mode*. URL: https://docs.px4.io/main/en/flight_modes/land.html (cit. on p. 46).

[29] *Parameter Reference*. URL: http://docs.px4.io/main/en/advanced_config/parameter_reference.html (cit. on p. 47).

[30] *X500 Kit*. URL: https://shop.holybro.com/x500-kit_p1180.html (cit. on p. 59).

[31] *Husky*. URL: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/ (cit. on p. 60).

[32] *Raspberry Pi*. URL: hhttps://docs.rs-online.com/3b9b/0900766b814db308.pdf (cit. on p. 60).

[33] *Raspberry Pi Camera V2*. URL: https://www.tradeinn.com/techinn/it/raspberry-pi-camera-v2/137489672/p (cit. on p. 61).

[34] *EVK1000*. URL: https://www.qorvo.com/products/p/EVK1000#overview (cit. on p. 61).

[35] *H-RTK F9P GNSS Series*. URL: https://shop.holybro.com/h-rtk-f9p-gnss-series_p1226.html (cit. on p. 62).

[36] Chongfeng Liu, Zixing Jiang, Ruoyu Xu, Xiaoqiang Ji, Lianxin Zhang, and Huihuan Qian. «Design and Optimization of a Magnetic Catcher for UAV Landing on Disturbed Aquatic Surface Platforms». In: Apr. 2022. DOI: 10.1109/ICRA46639.2022.9812270 (cit. on p. 66).