Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2020

Autonomous Landing of an Unmanned Aerial Vehicle on an Unmanned Ground Vehicle in a GNSS-denied scenario

Albin Andersson Jagesten and Alexander Källström

Master of Science Thesis in Electrical Engineering

Autonomous Landing of an Unmanned Aerial Vehicle on an Unmanned Ground Vehicle in a GNSS-denied scenario:

Albin Andersson Jagesten and Alexander Källström

LiTH-ISY-EX–20/5327–SE

Supervisor: **Anton Kullberg**
ISY, Linköpings universitet
Jouni Rantakokko
Swedish Defence Research Agency
Jonas Nygårds
Swedish Defence Research Agency
Joakim Rydell
Swedish Defence Research Agency

Examiner: **Gustaf Hendeby**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Copyright © 2020 Albin Andersson Jagesten and Alexander Källström

Abstract

An autonomous system consisting of an unmanned aerial vehicle (UAV) in cooperation with an unmanned ground vehicle (UGV) is of interest in applications for military reconnaissance, surveillance and target acquisition (RSTA). The basic idea of such a system is to take advantage of the vehicles strengths and counteract their weaknesses. The cooperation aspect suggests that the UAV is capable of autonomously landing on the UGV. A fundamental part of the landing is to localise the UAV with respect to the UGV. Traditional navigation systems utilise global navigation satellite system (GNSS) receivers for localisation. GNSS receivers have many advantages, but they are sensitive to interference and spoofing. Therefore, this thesis investigates the feasibility of autonomous landing in a GNSS-denied scenario.

The proposed landing system is divided into a control and an estimation system. The control system uses a proportional navigation (PN) control law to approach the UGV. When sufficiently close, a proportional-integral-derivative (PID) controller is used to match the movements of the UGV and perform a controlled descent and landing. The estimation system comprises an extended Kalman filter that utilises measurements from a camera, an IMU and ultra-wide band (UWB) impulse radios. The landing system is composed of various results from previous research.

First, the sensors used by the landing system are evaluated experimentally to get an understanding of their characteristics. The results are then used to determine the optimal sensor placements, in the design of the EKF, as well as, to shape the simulation environment and make it realistic. The simulation environment is used to evaluate the proposed landing system. The combined system is able to land the UAV safely on the moving UGV, confirming a fully-functional landing system. Additionally, the estimation system is evaluated experimentally, with results comparable to those obtained in simulation. The overall results are promising for the possibility of using the landing system with the presented hardware platform to perform a successful landing.

Acknowledgments

We would like to thank our supervisors Jouni Rantakokko, Jonas Nygårds and Joakim Rydell at FOI, who have all shown interest in this work and helped us throughout this master's thesis project. Special thanks goes to Jouni Rantakokko for his thorough work in proofreading and giving criticisms on this thesis, and Joakim Rydell who assisted in the experimental test.

At Linköping Univerisity, we would like to thank our supervisor Anton Kullberg for his help with this thesis. We would also like to take the opportunity to thank our opponents Carl Hynén Ulfsjö and Theodor Westny for their constructive comments. Finally, we would like to thank our examinator Gustaf Hendeby for his great interest throughout this master's thesis project as well as his aid during the experimental test.

As a closing remark, we would like to praise both FOI and Linköping University for doing their very best in making sure that this master's thesis could proceed, despite the covid-19 pandemic.

Linköping, June 2020
Albin Andersson Jagesten and Alexander Källström

Contents

Notation

Notations

Notation	Meaning
$[x, y, z]$	Position in a Cartesian coordinate system
$[\phi, \theta, \psi]$	Euler angles roll, pitch and yaw
{a}	Coordinate frame a
${}^{a}\mathcal{R}_{b}$	Rotation matrix from $\{b\}$ to $\{a\}$
$\mathbf{x}$	State vector
c_x	$\cos x$
s_x	$\sin x$
R	Covariance matrix
$\|\cdot\|_n$	n-norm of $\cdot$
$\dot{a}$	Time derivative of a
$\otimes$	Kronecker product
A^T	Transpose of matrix A
$:=$	Definition statement

Abbreviations

Abbreviation	Meaning
AWGN	Additive White Gaussian Noise
CRLB	Cramér-Rao Lower Bound
CSI	Camera Serial Interface
FOI	Swedish Defence Research Agency
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
EKF	Extended Kalman Filter
FOV	Field of View
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LOS	Line of Sight
MPC	Model Predictive Control
NIS	Normalised Innovation Squared
NLOS	Non Line of Sight
PD	Proportional, Differential (controller)
PDF	Probability Density Function
PID	Proportional, Integral, Differential (controller)
PN	Proportional Navigation
RMSE	Root Mean Square Error
ROS	Robot Operating System
RTK	Real Time Kinematic
RSTA	Reconnaissance, Surveillance and Target Acquisition
SITL	Software In The Loop
TOA	Time of Arrival
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UWB	Ultra-Wide Band

1 Introduction

This master's thesis project was conducted at the Swedish Defence Research Agency (FOI) at the Division of C4ISR[1] in Linköping. The main goal of the project was to investigate the possibility of landing a quadrotor Unmanned Aerial Vehicle (UAV) on top of a mobile Unmanned Ground Vehicle (UGV) in a GNSS-denied scenario.

The purpose of this chapter is to present the background as to why the project was conducted, related work done in the research community, the approach of the thesis, the problems the thesis aims to answer and the limitations and contributions of the project. Finally, an outline of the thesis is provided.

1.1 Background

During the last couple of decades, the interest for, as well as the research and development (R&D) on, unmanned and autonomous systems has grown rapidly. In this thesis, autonomous systems are defined as systems which can operate without direct human control or intervention. The various R&D activities have resulted in unmanned systems that can perform increasingly more complex tasks and support military operations in all domains (air, ground and sea). Currently, the R&D community is exploring various approaches for combining different types of unmanned platforms, which can complement each other's abilities. One such conceivable scenario is an unmanned ground vehicle (UGV) acting as a host platform for an unmanned aerial vehicle (UAV) to carry out various missions, as explored in [1]. The underlying idea is to exploit the advantages while mitigating the disadvantages for both types of vehicles. The ground vehicle can continue to operate during harsh weather conditions and has the endurance to operate for an extensive period. In contrast, small UAV:s have limited endurance. The benefit

[1] Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance

of the UAV is instead that it can move at higher speeds, can cover large areas and provide an overview of the surrounding environment. The UAV may also act as an elevated communication relay. Additionally, the UGV acts as a transport and recharge station for the UAV.

The task of developing a combined UAV and UGV reconnaissance, surveillance and target acquisition (RSTA) system has several non-trivial research challenges, which each could encapsulate a single thesis project. Therefore, the challenge which this master's thesis is exploring concerns the autonomous landing of the UAV on top of a moving UGV.

In order to perform an autonomous landing, a basic requirement is to estimate the relative position and orientation of the UAV with respect to the UGV. The localisation is performed with sensors on the UAV as well as on the UGV. Because of the military application and the importance of the landing phase, there are requirements on the sensors regarding robustness as well as low probability of detection. Additionally, a future goal is for the UAV to be able to land at night time as well, which should be considered in the choice of sensors. One common approach to localise a UAV is with the use of global navigation satellite system (GNSS) receivers. Though GNSS-receivers are small, low cost and readily available, they do come with some flaws. In urban environments GNSS signals can be subjected to multi-path propagation, attenuation and diffraction which can impair the accuracy of the GNSS-receivers [2]. GNSS-receivers are also sensitive to jamming and spoofing. For example Kerns et al. [3] were able to send a drone into an unwanted dive through spoofing of a global positioning system (GPS) signal. Because of these flaws, this thesis will investigate the possibility of autonomously landing a UAV on a moving UGV in a GNSS-denied scenario.

A localisation technique that has gained attention lately, is the use of ultra-wide band (UWB) impulse radios. UWB radios send data at bandwidths surpassing 500 MHz. The large bandwidth has many advantages, such as being insensitive to disturbances as well as not being affected by multipath propagation [4]. Additionally, it results in a low power spectral density, which reduces the risk of being detected. A thorough description of how UWB radios work and how they can be used for localisation is provided in [4]. The standard approach of localisation using UWB radios is based on having multiple stationary UWB radios (anchors) placed in an environment in which a mobile UWB radio (tag) is to be located. The distances between the tag and the anchors are measured and used to calculate an estimate of the tag's position relative to the anchors. When combining the distance measurements, a 3D position can be triangulated with a minimum of four anchors [5].

1.2 Related Work

The task of landing a UAV on a mobile UGV is not entirely new and has been studied before in various scientific publications. A couple of research contributions propose solutions that works well in real world conditions with the UGV moving at a relatively high velocity. Persson [6] was able to land a fixed-wing UAV

on a car moving at 70 km/h. During the landings a PID control structure was used. The position estimation of the vehicles was achieved using two real time kinematic (RTK) GPS receivers together with a camera system measuring the relative position between the vehicles. The work was extended in [7] where a model predictive control (MPC) structure was used instead.

In [8] a similar result was achieved with a quadrotor UAV. Borowczyk et al. managed to land the quadrotor on a car travelling at speeds up to 50 km/h. In this case a proportional navigation (PN) controller was used to approach the car. A switching strategy was used, where a proportional-derivative (PD) controller was used during the landing phase. An inertial measurement unit (IMU), a camera and GPS receivers were used together with a Kalman filter to estimate the motion of the two vehicles.

In 2017, the International Robotics Challenge (MBZIRC) competition was organised with the goal of expanding the state-of-the-art in a variety of robotics domains [9]. One challenge in the competition was to autonomously land a UAV on a moving UGV (15 km/h). The system Baca et al. [9] participated with achieved the shortest landing time. In their solution a MPC structure was used. To get a state estimate of the UAV's movements, inertial sensors, a rangefinder, a camera and an RTK GPS receiver was used. However, no sensors could be placed on the UGV. Baca et al. therefore used a prediction strategy for the UGV's movements. Although these research groups achieved the goal of autonomous landing, all the proposed approaches rely on GNSS-receivers.

Autonomous landing of a UAV in a GNSS-denied scenario is an active research field, primary considering confined indoor environments, and the concept have been investigated by various research groups. The most common strategy is positioning based on visual data, usually a camera identifying a fiducial marker. In [10] a vision-based detection algorithm is used to estimate the 3D-position of a UAV relative a tag. Wenzel et al. [11] utilise an IR-camera mounted on a UAV to track a pattern of IR-lights on a moving platform. The AprilTag fiducial algorithm is used with a camera for positioning during an autonomous landing of a UAV in [12] and [13]. The common denominator of these works is that they all utilise a proportional-integral-derivative (PID) controller in order to accomplish the autonomous landing. The focus on visual data in these works means that they can only land from a starting position in a close proximity of the UGV, since the UGV must be within the camera's field of view (FOV) during the majority of the landing.

An approach to GNSS-denied positioning that works at longer distances is by using UWB anchors positioning a tag in a sensor network. For example, [14, 15] examined the specific task of estimating the position of a UAV with UWB radios. These articles show successful results; however, the positioning is conducted in a closed environment where the tag (on the UAV) is positioned in the volume spanned by the anchors. Such an approach is not feasible in an arbitrary environment. Lazzari et al. [5] numerically investigate the approach of having anchors mounted on a UGV and a tag mounted on a UAV. The results from [5] are promising, with 3D-positioning of the UAV achieved at distances up to 80 m.

1.3 Approach

In this work, the estimation techniques used in GNSS-denied scenarios are combined with control approaches that has proven to work well in challenging real-world scenarios. The landing system is divided into an *estimation system* and a *control system*. The estimation system has the goal of estimating the relative position between the vehicles. The control system utilises these estimates to steer the UAV towards the UGV, and eventually land.

The approach in the estimation system is to substitute GNSS-receivers with a UWB sensor network. The UWB sensor network consists of four UWB anchors mounted on the UGV measuring the distance to a UWB tag, which is mounted on the UAV. To aid in the relative positioning at shorter distances, a camera system consisting of a UAV-mounted camera identifying a fiducial marker on the UGV is added. Additionally, an IMU is mounted on each vehicle. The IMU provides 3D accelerations, rotation rates as well as magnetic field and air pressure (scalar) measurements. An extended Kalman filter (EKF) utilises these sensors to estimate a relative position between the vehicles.

The proposed control system is mainly based on [8]. Two control laws are used. A PN law is used to approach the UGV. At closer distances, the control system switches to a PID controller with the goal of matching the movement of the UGV, while descending towards it.

1.4 Problem Formulation

The high level goal of the system is to land the UAV in a safe and controlled manner on the UGV, by utilising the sensors mounted on both vehicles. Based on the approach presented, the thesis aims to answer the following questions:

- How should the estimation system be designed in order to provide adequate state estimates for a landing in a GNSS-denied scenario?
- How should the control system be designed to achieve a controlled landing on a mobile UGV?

Also, since the long-term goal is a system that can land independent of weather conditions or time of day, the following question is also interesting:

- Is the camera vital in the estimation system?

1.5 Limitations

This thesis will only consider the landing phase for the UAV, which is when the UAV is at most 50 m away from the UGV. The UGV is therefore assumed to be in line of sight of the UAV during the entire landing task. Additionally, it is assumed that the landing takes place in an obstacle-free environment. The UGV is also assumed to be moving only in the horizontal plane. Finally, the landing

system can only control the UAV when performing the landing manoeuvre. These limitations do not restrict the core problem of autonomously landing a UAV in a GNSS-denied scenario.

1.6 Contributions

Due to the complexity of the task, openly available software is utilised whenever available. The main contribution of the thesis is the selection and evaluation of estimation and control algorithms, as well as the design and software integration of the system. More specifically, the contributions in this thesis are the following:

- Evaluation of UWB radios performance.
- Analysis of UWB anchor configuration using CRLB.
- Design of a complete landing system, consisting of an estimation and a control system.
- Evaluation of the landing system in a realistic simulation environment.
- Experimental evaluation of the estimation system's performance during the final descend.

With regard to the personal contributions, Albin Andersson Jagesten was responsible for integrating the simulation environment, implementing the landing system and the development of the Control System. Alexander Källström had the responsibility of the development of the Estimation System, the evaluation of the UWB radios, analysis of the UWB anchor configuration as well as the experimental evaluation of the estimation system. However, both authors were involved in the development of all the various parts of this master's thesis project.

1.7 Outline of Thesis

The outline of this thesis is:

- Chapter 2 gives a description of the UAV and UGV, as well as the sensors mounted on each vehicle. The simulation environment is presented as well.
- Chapter 3 presents the theory behind the topics relevant to this work.
- Chapter 4 describes the estimation system, as well as a description and evaluation of each sensor.
- Chapter 5 presents the idea behind, and implementation of the control system as well as the results generated in simulation.
- Chapter 6 describes the results generated in simulation using both the estimation and the control system. Experimental results of the estimation system are also presented.
- Chapter 7 discusses the main results and presents ideas for future work.

2

System Description

This chapter aims to describe the different parts that comprise the landing system including hardware platforms, sensors and software. A description of the simulation environment is provided as well. The hardware platform, as well as, the sensors mounted on the UAV and UGV are described in Section 2.1 and 2.2, respectively. A short description of the software tools used in the project is provided in Section 2.3. Finally, the simulation environment is presented in Section 2.4.

2.1 Unmanned Aerial Vehicle

The UAV consists of a LIN700 platform equipped with a 3DR Pixhawk Mini[2] autopilot. The Pixhawk Mini is running the open-source flight controller firmware ArduCopter. However, the main processing unit on the UAV is a Jetson Nano Developer Kit[3] single-board computer. The Jetson Nano handles the data from the sensors on the UAV, UGV and from the Pixhawk Mini. Based on the received sensor data, the Jetson sends appropriate control commands to the Pixhawk with the goal of landing the UAV on the UGV. An overview of the UAV system architecture is presented in Figure 2.1. The individual parts of the UAV are described in the following subsections.

2.1.1 Platform

The UAV used in this project is a quadrotor platform that can be seen in Figure 2.2. The UAV has the following sensors mounted on it:

[2]For a detailed description, see: https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk_mini.html

[3]Jetson Nano Developer Kit documentation: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

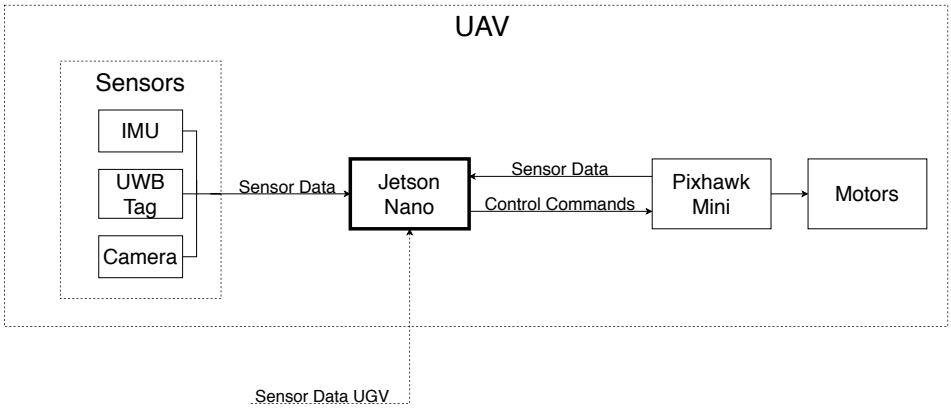


Figure 2.1: *Overview of the components that make up the landing system.*

- A Decawave DWM1001-DEV UWB module.
- A Raspberry Pi Camera Module V2.
- An XSens MTi-100 IMU.

The sensors are mounted on a platform, see Figure 2.3, which in turn is mounted underneath the UAV. The sensors are connected to the Jetson Nano processor unit, which is also mounted on the platform. The drivers used to communicate with the sensors are described in Appendix A.

2.1.2 Sensors

The UAV is equipped with a Decawave DWM1001 UWB module acting as a UWB tag in the UWB sensor network. The DWM1001 contains a UWB-transceiver as well as a processing unit. The UWB tag measures the individual distances to the anchors in the UWB sensor network.

The Raspberry Pi Camera Module V2 is a camera facing downwards from the UAV. The camera has a horizontal field of view (FOV) of 62.2° and is configured to send images with a dimension of 640×480 pixels at a rate of 10 Hz.

The Xsens MTi-100 IMU is utilised for its accelerometer and barometer measurements. The three-axis accelerometer measures the acceleration vector of the UAV and the barometer measures the pressure of the air around the UAV.

As presented in Figure 2.1, the Jetson Nano receives sensor data from the Pixhawk Mini, which is in the form of an attitude estimate of the UAV.

Figure 2.2: *LIN700 UAV platform.*

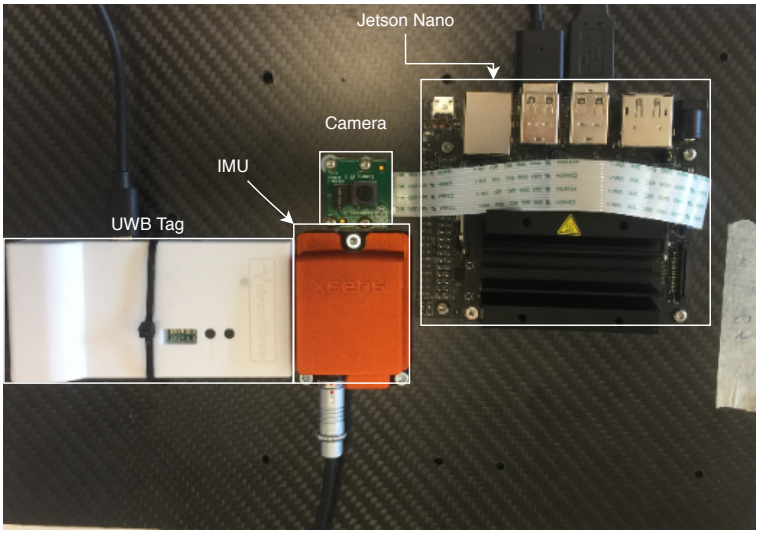


Figure 2.3: *Sensor platform attached to the UAV.*

2.2 Unmanned Ground Vehicle

The UGV platform and the sensors mounted on it are described.

2.2.1 Platform

The UGV is a modified electrical wheelchair. It has been used as a test platform in previous research at FOI [16, 17]. On top of the wheelchair a landing platform is mounted with dimensions 1.5 x 1.5 m^2. A fiducial marker with dimensions 1.4 x 1.4 m^2 is placed on the centre of the landing platform. Four 0.5 m long beams pointing upwards are attached to each of the platform's corners. The beams are used for sensor placement. The following sensors have been mounted on the UGV:

- Four Decawave DWM1001 UWB modules.
- An XSens MTi-600 DEV IMU.

The resulting UGV platform can be seen in Figure 2.4.

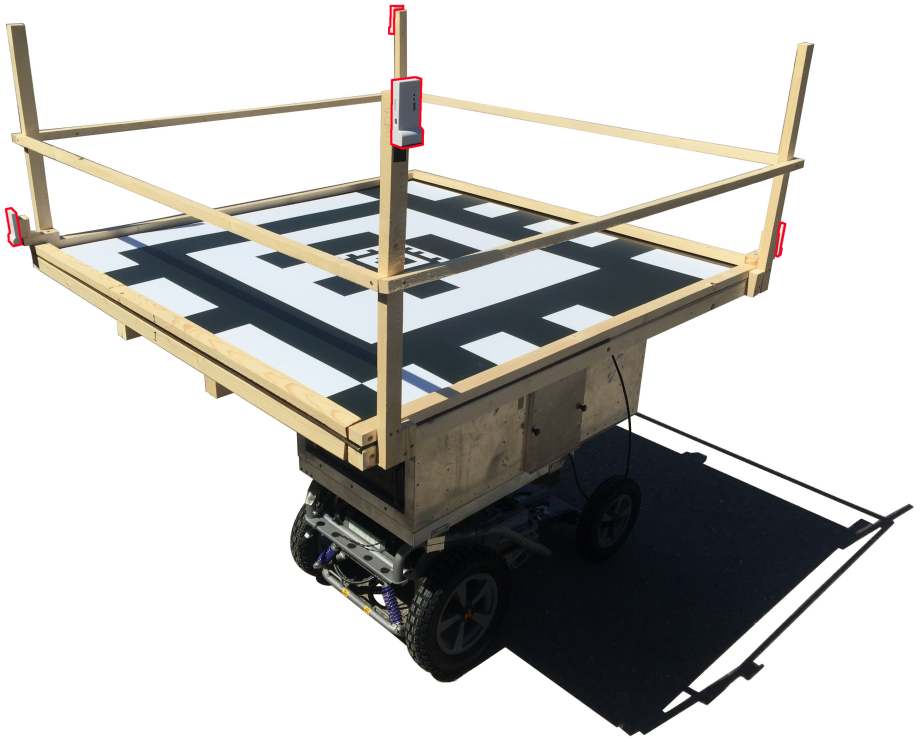


Figure 2.4: *UGV platform with four UWB anchors and an AprilTag fiducial marker. The UWB anchors are highlighted with red.*

2.2.2 Sensors

The four Decawave DWM1001 UWB modules are used as anchors in the UWB sensor network. The UWB modules are attached to the landing platform corners, two in level with the platform and two on top of the beams, see Figure 2.4.

The Xsens IMU provides acceleration and air pressure measurements, as well as estimates of the UGV orientation.

2.3 Software Tools

This section describes the main software tools utilised in the system.

2.3.1 Robot Operating System

The Robot Operating System[4] (ROS) is an open-source framework for robot software. The framework provides a communication interface together with a variety of open-source software packages for robot applications. Additionally, open-source tools for hardware abstraction, data visualisation, data recording and data playback that support the ROS communication interface are available.

2.3.2 ArduCopter Flight Controller

ArduCopter[5] is an open-source UAV flight controller made by ArduPilot. The flight controller supports a variety of multi-rotor UAV's and in this thesis a quadcopter is used. An open-source software in the loop (SITL) simulator[6] of the flight controller is utilised. The SITL simulator enables the use of the flight controller when flying a simulated quadcopter in Gazebo.

2.4 Simulation Environment

To allow efficient and safe development of the landing system, a realistic simulation environment is needed. In this work the open source simulation environment Gazebo is used. Gazebo is specifically tailored for robotics and it is fully compatible with ROS and the ArduCopter SITL implementation.

A UAV Gazebo model is included in the ArduCopter SITL implementation. The model is slightly modified such that it has a fixed downfacing camera. The UAV model can be seen in Figure 2.5. The model is equipped with the same sensors as the real UAV. The UGV model was created from scratch in Gazebo. It can be seen in Figure 2.6. The UGV is equipped with an IMU and four UWB-anchors. Plugins, which are blocks of code that executes during runtime, are used to interact with the models. The plugins created in this work are described in

[4] For more information about the robot operating system, see: https://www.ros.org/about-ros/

[5] ArduCopter documentation: https://ardupilot.org/copter/

[6] ArduPilot Software In The Loop simulator documentation: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

Appendix C. The implementation of the simulated sensors used in the simulation environment are presented in Appendix B.

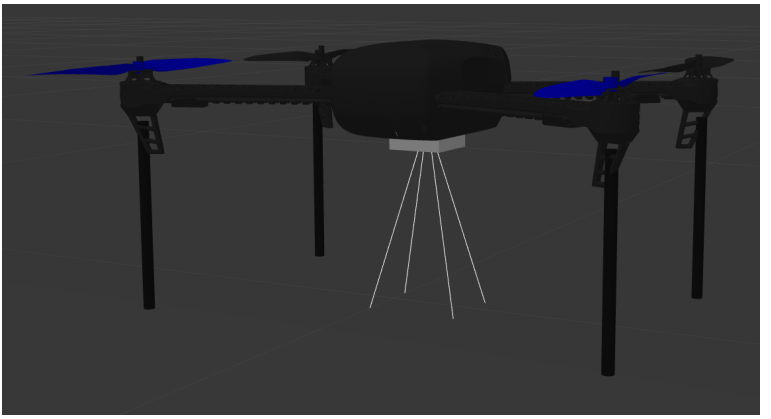


Figure 2.5: *Gazebo UAV simulation model.*

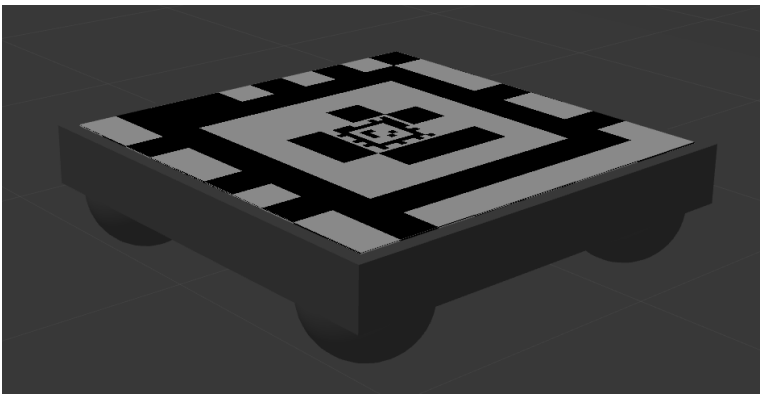


Figure 2.6: *Gazebo UGV simulation model.*

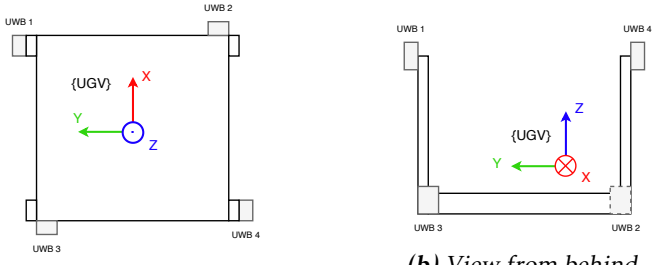
3

Theory

This chapter presents the various theoretical concepts that are applied in this thesis.

3.1 Coordinate Systems

This section describes the coordinate frames used in this thesis. Three main coordinate frames are introduced in Figure 3.2. The frame {W} denotes the world frame where both the UAV and the UGV reside. The frame {UAV} and the frame {UGV} represent the body fixed frame for the UAV and the UGV respectively. The placement of the UWB sensors on the UGV can be seen in Figure 3.1. The UAV has a camera and an IMU attached. The orientation of the two sensors in the {UAV} frame can be seen in Figure 3.3.



(a) *View from above.*

(b) *View from behind.*

Figure 3.1: *The placement of the UWB sensors in the {UGV} frame.*

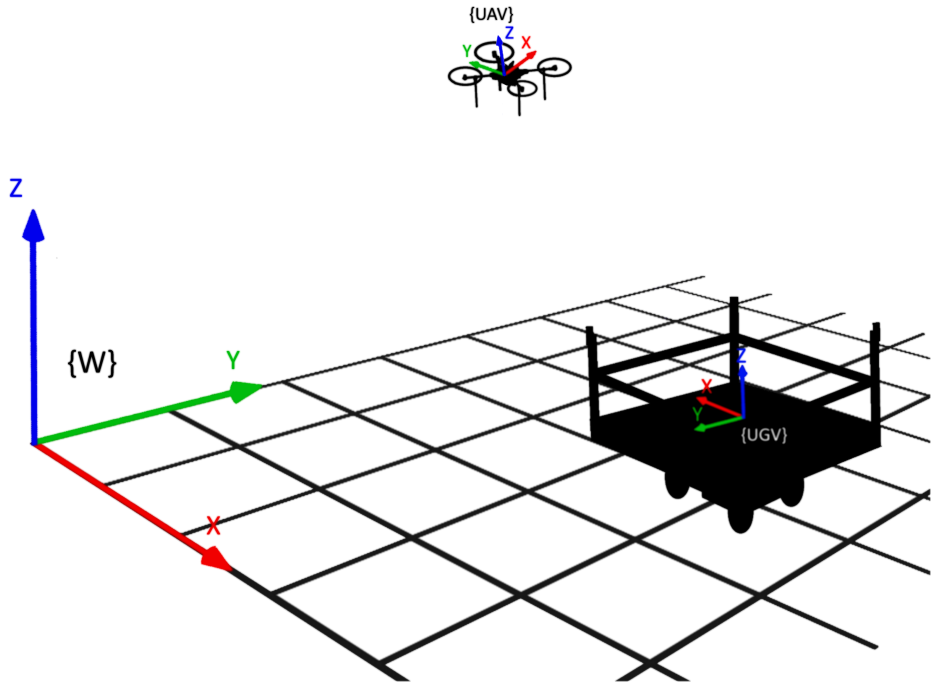


Figure 3.2: *The three main coordinate frames. {W} denotes the world frame , {UAV} denote the body fixed coordinate frame of the UAV and {UGV} denote the body fixed coordinate frame of the UGV.*

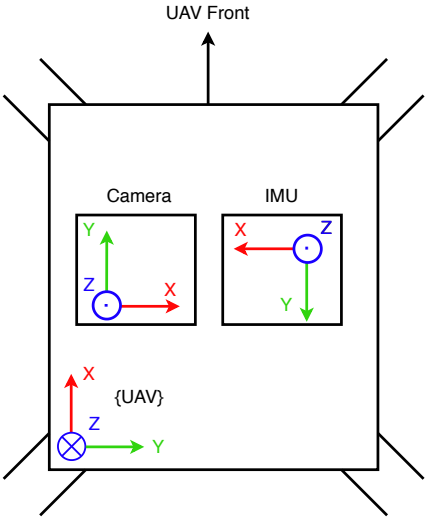


Figure 3.3: *The camera and IMU placement and orientation in the {UAV} frame. The view is from underneath.*

3.2 Rotation Representations

This section introduces two ways of representing the rotation of one coordinate frame with respect to another. The first representation is with Euler angles and the second with a unit quaternion vector.

The Euler angles representation is based on three subsequent coordinate rotations. These coordinate rotations can describe any arbitrary rotation [18]. The rotations are specified by the rotation angles roll, pitch and yaw. In this work the right-hand rule is used as the rotation convention. The roll angle, ϕ, represents a rotation around the x-axis, the pitch angle, θ, represents a rotation around the y-axis and the yaw angle, ψ, represents a rotation around the z-axis. The rotation matrices are expressed as follows

$$\mathcal{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix}, \quad \mathcal{R}_\theta = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}, \quad \mathcal{R}_\psi = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where $c_v = \cos(v)$ and $s_v = \sin(v)$.

These three rotation matrices form the rotation matrix

$$^a\mathcal{R}_b = \mathcal{R}_\psi \mathcal{R}_\theta \mathcal{R}_\phi \,, \tag{3.2}$$

where a and b are two coordinate frames and b is rotated relative to a. A vector $\mathbf{x}_b \in \mathbb{R}^3$ expressed in frame b can be transformed to frame a as follows

$$\mathbf{x}_a = {}^a\mathcal{R}_b \mathbf{x}_b \,. \tag{3.3}$$

A unit quaternion $\mathbf{q}$ is a vector with length one that comprises four real numbers $\mathbf{q} = [x, y, z, w]^T$. One way of understanding the quaternion representation is to divide the quaternion vector into a vector part $[x, y, z]^T$ and a scalar part w. Any finite rotation can be described by a single rotation α around a normalised axis $\mathbf{n}$ [18]. The vector part $[x, y, z]^T$ is equal to $\mathbf{n}\sin(\frac{\alpha}{2})$ and the scalar part w is equal to $\cos(\frac{\alpha}{2})$. The resulting quaternion is therefore

$$\mathbf{q} = \begin{bmatrix} \mathbf{n}\sin(\frac{\alpha}{2}) \\ \cos(\frac{\alpha}{2}) \end{bmatrix}. \tag{3.4}$$

The conversion from Euler angles to a unit quaternion vector is

$$x = \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \tag{3.5a}$$
$$y = \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \tag{3.5b}$$
$$z = \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \tag{3.5c}$$
$$w = \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \tag{3.5d}$$

3.3 Quadcopter Dynamical Model

This section presents the dynamical model of a quadcopter. The low-level control commands for a quadcopter is the thrust of each rotor. The rotor thrusts can be

modelled more thoroughly, but in this thesis they are simplified to the orientation of the UAV as well as a total thrust vector along the z-axis of the body-frame. The dynamical model has the purpose of describing the resulting dynamics from the orientation and thrust vector.

Let (x, y, z) be the position of the UAV in the {W}-frame, (ϕ, θ, ψ) the roll, pitch and yaw angle of the UAV with respect to the {W}-frame. Furthermore, T denotes the total thrust generated by the rotors, i.e. the magnitude of the force vector propelling the UAV along the direction of the z-axis in the {UAV}-frame. A second force vector affecting the UAV is from the force of gravity. Mahony et al. [19] present a model of the resulting dynamics from the two force vectors as the combined accelerations expressed in the {W}-frame. The model is

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = m\begin{bmatrix}0\\ 0\\ -g\end{bmatrix} + {}^{W}\mathcal{R}_{\text{UAV}}\begin{bmatrix}0\\ 0\\ T\end{bmatrix}, \tag{3.6}$$

where m is the mass of the quadcopter, g is the gravitational acceleration and ${}^{W}\mathcal{R}_{\text{UAV}}$ the rotation matrix from {UAV} to {W}, which is given by

$${}^{W}\mathcal{R}_{\text{UAV}} = \begin{bmatrix}c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi\\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi\\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta\end{bmatrix}. \tag{3.7}$$

By inserting (3.7) into (3.6) and dividing by m, the resulting model is

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -g\end{bmatrix} + \begin{bmatrix}c_\phi s_\theta c_\psi + s_\phi s_\psi\\ c_\phi s_\theta s_\psi - s_\phi c_\psi\\ c_\phi c_\theta\end{bmatrix}\frac{T}{m}. \tag{3.8}$$

3.4 Kalman Filter Theory

When utilising measurements from multiple sensors with varying characteristics such as uncertainty and update rate, it is not trivial to combine the sensor data. One way of combining the data is with a Kalman filter [20]. The Kalman filter is a recursive algorithm that estimates several unknown variables, called states, with a multivariate normal distribution, which is described with a mean $\boldsymbol{\mu}$ and a covariance matrix Σ. The filter estimates the states using sensor measurements, a measurement model and a dynamic model. The Kalman filter assumes a dynamic and measurement model where the states enter as linear arguments. However, that is not the case for most real world systems.

The extended Kalman filter (EKF) is an extension of the Kalman filter that handles nonlinear dynamic and measurement models [21]. The key idea is to linearize the nonlinearities. This can be done in several ways, but a common approach is using first order Taylor series expansions.

Consider the nonlinear dynamical model $\mathbf{g}'$ and the nonlinear sensor measurement model $\mathbf{h}'$ with zero-mean additive white Gaussian noise (AWGN), the

combined expressions are defined as

$$\mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t, \boldsymbol{\epsilon}_t)) = \mathbf{g}'(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, Q_t), \tag{3.9}$$

$$\mathbf{h}(\mathbf{x}_t, \delta) = \mathbf{h}'(\mathbf{x}_t) + \boldsymbol{\delta}_t, \quad \boldsymbol{\delta}_t \sim \mathcal{N}(0, R_t). \tag{3.10}$$

Using the first order Taylor expansion, $\boldsymbol{\mu}$ and Σ are estimated with the following recursive scheme

$$\hat{\boldsymbol{\mu}}_t = \mathbf{g}'(\tilde{\boldsymbol{\mu}}_{t-1}, \mathbf{u}_t) \tag{3.11a}$$

$$\hat{\Sigma}_t = G_t \tilde{\Sigma}_{t-1} G_t^T + Q_t \tag{3.11b}$$

$$S_t = H_t \hat{\Sigma}_t H_t^T + R_t \tag{3.11c}$$

$$K_t = \hat{\Sigma}_t H_t^T S_t^{-1} \tag{3.11d}$$

$$\tilde{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\mu}}_t + K_t(\mathbf{z}_t - \mathbf{h}'(\hat{\boldsymbol{\mu}}_t)) \tag{3.11e}$$

$$\tilde{\Sigma}_t = (I - K_t H_t)\hat{\Sigma}_t (I - K_t H_t)^T + KRK^T \tag{3.11f}$$

where S denotes the innovation covariance, K the Kalman gain and

$$G_t = \frac{d\mathbf{g}'(\mathbf{x}, \mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\hat{\boldsymbol{\mu}}_t, \mathbf{u}=\mathbf{u}_t} \tag{3.12a}$$

$$H_t = \frac{d\mathbf{h}'(\mathbf{x})}{d\mathbf{x}}\Big|_{\mathbf{x}=\hat{\boldsymbol{\mu}}_t}. \tag{3.12b}$$

The recursions start from the initial guess of $\hat{\mu}_0$ and $\hat{\Sigma}_0$. The equations (3.11a) and (3.11b) are called the prediction step. These equations propagate the system dynamics through time using the dynamical model. This step predicts the current state of the system ($\hat{\boldsymbol{\mu}}_t$, $\hat{\Sigma}_t$). During the prediction step, the uncertainty in the estimate grows, i.e. the covariance matrix gets larger.

The second step incorporates the measurements in the estimate and is called the measurement step. In this step the prediction is corrected by the measurement and the confidence in the estimate increases. The measurement step consists of (3.11c)-(3.11f) and returns the estimates ($\tilde{\boldsymbol{\mu}}_t$, $\tilde{\Sigma}_t$). The final step in (3.11f) is expressed in the Joseph form. The Joseph form is equivalent to the $\tilde{\Sigma}_t$ calculation of the standard EKF, however, it is stated in a numerically stable expression [22].

3.4.1 Measurement Outliers

Real sensor data often contains measurement outliers that contain large errors. These outliers should not be used in the filter. One way of detecting measurement outliers is with the Mahalanobis distance which is used to form the normalised innovation squared (NIS) test [23]. The NIS is computed as the squared Mahalanobis distance of the measurement error with the innovation matrix S, equation (3.11c), as normaliser:

$$\text{NIS} = (\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t))^T S^{-1} (\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t)). \tag{3.13}$$

The NIS is χ^2-distributed, where the degrees of freedom are equal to the number of measurements in the measurement vector $\mathbf{z}_t$. A confidence level of the χ^2-distribution is used to classify the measurements as either a valid sample or as an outlier.

3.4.2 Measurement Information Gain

When considering the information gain of a measurement the Cramér-Rao lower bound (CRLB) can be used. The CRLB is calculated from a sensor measurement model, and returns a lower bound for the covariance matrix of the estimated states, i.e.

$$\Sigma \geq \text{CRLB}\,, \tag{3.14}$$

assuming an unbiased estimator. The lower bound gives an indication of how accurately the states can be estimated. To calculate the CRLB, consider a linear sensor measurement model $\mathbf{h}'(\boldsymbol{x}) = H\boldsymbol{x}$ with a constant covariance matrix $R_t = R$. As described in [24], the CRLB is

$$\text{CRLB} = (H^T R^{-1} H)^{-1} \tag{3.15}$$

3.5 Position Trilateration

As mentioned earlier, the EKF algorithm requires an initial guess. In this work, the EKF will be used to estimate the position of the UAV relative to the UGV. The initial guess for the relative position can be obtained by triangulating the range data from the UWB radios. In [25] a number of methods are investigated to estimate the position of a radiating source using range measurements from m beacons, which resembles the UWB radios in this work. Beck et al. [25] assume that the range measurements from beacon i has the following form

$$r_i = \|\mathbf{x} - \mathbf{a}_i\|_2 + \epsilon_i\,, \tag{3.16}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the position to be estimated, $\mathbf{a}_i \in \mathbb{R}^n$ is the position of the ith beacon and ϵ_i is AWGN. One of the examined methods is an unconstrained least square approach, which has the following form

$$\min_{\mathbf{y}\in\mathbb{R}^{n+1}} \|A\mathbf{y} - \mathbf{b}\|_2\,, \tag{3.17}$$

where

$$A = \begin{bmatrix} -2\mathbf{a}_1^T & 1 \\ \vdots & \vdots \\ -2\mathbf{a}_m^T & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} r_1^2 - \|\mathbf{a}_1\|_2^2 \\ \vdots \\ r_m^2 - \|\mathbf{a}_m\|_2^2 \end{bmatrix}. \tag{3.18}$$

The solution is then given by

$$\mathbf{y}^* = (A^T A)^{-1} A^T \mathbf{b}\,. \tag{3.19}$$

The first n terms of $\mathbf{y}^*$ is the estimate of $\mathbf{x}$.

3.6 Camera Model

A pinhole camera model is used to model the camera attached to the UAV. The pinhole model assumes that a 3D-scene is projected on the image plane via a perspective transformation, see Figure 3.4. The model consists of intrinsic and extrinsic parameters [26]. The extrinsic parameters describe the spatial relation between the camera and the objects observed in the camera frame, i.e. a translation and a rotation. The intrinsic parameters describe the relation between world coordinates and pixel coordinates in the image data. The intrinsic parameters are associated with the specific camera and its settings. Therefore, the intrinsic parameters can be estimated once and then reused.

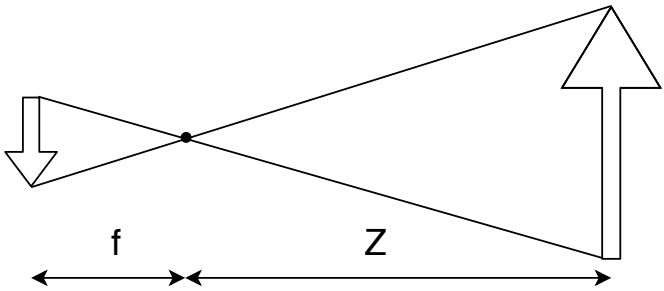


Figure 3.4: *Pinhole model illustration where f is the focal length and Z is the distance to the camera.*

The goal of the camera model is to map global coordinates $[X, Y, Z]^T$ to pixel coordinates $[u, v]^T$. The first step in the model is to transform the global coordinates to the camera frame using the extrinsic parameters as follows

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathcal{R} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t\,, \tag{3.20}$$

where $[x, y, z]^T$ are the coordinates in the camera frame. The matrix $\mathcal{R}$ and the vector t describe a rotation and a translation respectively, which together encapsulates the extrinsic parameters. The next step is to project $[x, y, z]^T$ to a normalised image plane using a perspective transformation as follows

$$x' = x/z, \quad y' = y/z. \tag{3.21}$$

The next step in the model is to account for radial and tangential distortions with the plumb bob model [27]

$$x'' = \left[1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right] x' + 2p_1 x' y' + p_2 \left[r^2 + 2x'^2\right] \tag{3.22a}$$

$$y'' = \left[1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right] y' + p_1 \left[r^2 + 2y'^2\right] + 2p_2 x' y' \tag{3.22b}$$

$$r = \sqrt{x'^2 + y'^2}\,, \tag{3.22c}$$

where k_1,k_2 and k_3 are radial coefficients and p_1 and p_2 are tangential distortion coefficients. The pixel coordinates u and v are finally obtained as

$$u = f_x \cdot x'' + c_x \tag{3.23a}$$
$$v = f_y \cdot y'' + c_y\,, \tag{3.23b}$$

where f_x and f_y describe the focal lengths in pixel coordinates. The constants c_x and c_y describe the centre of the image in pixel coordinates. To summarise, the intrinsic parameters are: k_1, k_2, k_3, p_1, p_2, f_x, f_y, c_x and c_y.

3.7 Proportional-Integral-Derivative Control

Proportional-integral-derivative (PID) control is a feedback control approach that is widely used in industrial control systems [28]. Assume that a system with input $u(t)$ and output $y(t)$ is to be controlled. With a desired set point $r(t)$, the goal of the controller is to generate an input $u(t)$ that makes the output $y(t)$ follow the set point, i.e. $y(t) = r(t)$. A feedback controller controls the input $u(t)$ based on the error signal $e(t)$, which is the difference between the set point and the output, i.e. $e(t) = r(t) - y(t)$.

The PID-controller consists of three parts:

- P – Accounts for the current error between $r(t)$ and $y(t)$ and generates an output that is proportional to the error.
- I – Accounts for past values of the error signal and generates an output that is proportional to the integral of the error, which therefore counteracts a static error.
- D – The generated signal is proportional to the derivative of the error and therefore accounts for estimated future values of the error signal, i.e. reduces overshoots.

The various parts have the corresponding proportional gains (K_p, K_i, K_d). The combined control signal is:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \tag{3.24}$$

Figure 3.5 presents a PID control feedback system.

A PID controller without the I-part ($K_i = 0$) is called a PD-controller.

3.8 Proportional Navigation Control

Proportional navigation (PN) is a guidance law [29]. Guidance laws are control laws with the goal of navigating a pursuer towards a target and reach it as fast as possible. The PN law is based upon the idea that the pursuer eventually will

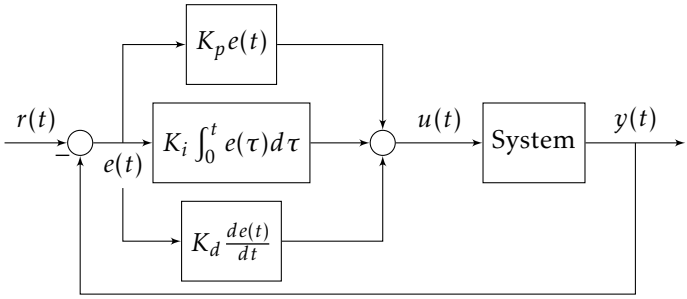


Figure 3.5: *A PID control feedback system.*

collide with the target if the line of sight (LOS) vector between them remains constant and that the distance is decreasing. This is achieved by generating an acceleration target $\mathbf{a}_\perp$ for the pursuer, which is perpendicular to the LOS-vector, such that when the target and the pursuer are moving, the LOS-vector remains the same.

Let the position and velocity of the pursuer be $\mathbf{p}_\mathrm{P}$ and $\mathbf{v}_\mathrm{P}$. Additionally, let the position and velocity of the target be $\mathbf{p}_\mathrm{T}$ and $\mathbf{v}_\mathrm{T}$. Then, let the relative position and velocity between the vehicles be $\mathbf{p}_\mathrm{rel} = \mathbf{p}_\mathrm{T} - \mathbf{p}_\mathrm{P}$ and $\mathbf{v}_\mathrm{rel} = \mathbf{v}_\mathrm{T} - \mathbf{v}_\mathrm{P}$. The acceleration $\mathbf{a}_\perp$ is then computed as follows

$$\mathbf{a}_\perp = -\lambda |\mathbf{v}_\mathrm{rel}| \frac{\mathbf{p}_\mathrm{rel}}{|\mathbf{p}_\mathrm{rel}|} \times \boldsymbol{\Omega}\,, \quad \boldsymbol{\Omega} = \frac{\mathbf{p}_\mathrm{rel} \times \mathbf{v}_\mathrm{rel}}{\mathbf{p}_\mathrm{rel} \cdot \mathbf{p}_\mathrm{rel}}\,, \tag{3.25}$$

where λ is a positive gain parameter controlling the rate of rotation. See Figure 3.6 for an illustration.

3.9 Barometric Formula

Both the UAV and the UGV are equipped with a barometer with the purpose of estimating the relative altitude between the vehicles. According to [30] the relation between relative altitude and pressure is described by the following equation:

$$z_\mathrm{rel} = z_\mathrm{A} - z_\mathrm{B} = \frac{T}{L}\left[\left(\frac{P_\mathrm{A}}{P_\mathrm{B}}\right)^{-\frac{LR}{g}-1}\right], \tag{3.26}$$

where T is the air temperature, z_rel is the difference in altitude between point A and point B, while P_A and P_B is the pressure at A and B respectively. Furthermore, L is the temperature lapse rate[7], g is the gravitational acceleration and R is the real gas constant for air. The equation is valid up to an absolute altitude of 11 km and can be inverted such that pressure is obtained from an altitude. The values of L, R and g are given in [31] which presents the Standard Atmosphere Mean Sea Level Conditions. From [31] the mean temperature at sea level $\bar{T}$ as well as the mean pressure at sea $\bar{P}_s$. The constants are presented in Table 3.1.

[7] The rate at which the temperature in the atmosphere falls with altitude.

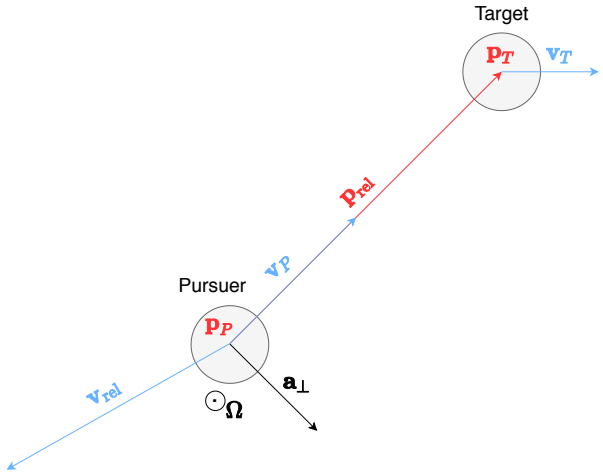


Figure 3.6: *Illustration of how the acceleration* $\mathbf{a}_\perp$ *is generated from the PN control law.* $\mathbf{p}_{rel}$ *and* $\mathbf{v}_{rel}$ *are the relative position and the relative velocity between the pursuer and the target.* $\boldsymbol{\Omega}$ *describes the rotation of the* LOS *vector and is perpendicular with* $\mathbf{p}_{rel}$ *and* $\mathbf{v}_{rel}$*. The acceleration* $\mathbf{a}_\perp$ *is in the direction of the cross product from* $\mathbf{p}_{rel}$ *and* $\boldsymbol{\Omega}$*.*

Table 3.1: *Barometric constants.*

	L	R	g	$\bar{T}$	$\bar{P}_s$
Value	-0.0065	287.04	9.80665	288	101.325
Unit	$\frac{K}{m}$	$\frac{m^2}{K \cdot s^2}$	$\frac{m}{s^2}$	K	kPa

4

Estimation System

The goal of the estimation system is to provide an estimate of the relative position and velocity between the UAV and the UGV. This chapter starts with an overview of the estimation system in Section 4.1, the underlying filter structure used for the estimation is described in Section 4.2 and a detailed description of the sensors that are used by the estimation system is presented in Sections 4.4–4.7.

4.1 Estimation System Description

The overall goal of the estimation system is to utilise sensor data from the sensors that are mounted on the UAV and UGV to estimate the relative position and velocity between the vehicles. Furthermore, the estimation system should be able to provide an estimate when the vehicles are over 50 meters apart. At those distances the estimate can be coarse, but as the UAV flies closer to the UGV, a higher accuracy is required. The most critical part in terms of the accuracy requirement for the estimation system is when the UAV is descending towards the UGV. The sensors used by the estimation system are described in Sections 2.1.2 and 2.2.2. The measurements from these sensors are:

- Attitude measurements of the UAV and UGV from the flight control unit as well as the IMU on the UGV, see Section 4.3.
- Distance measurements between the tag and the four anchors in UWB sensor network, see Section 4.4.
- Relative position measurements from the camera system, see Section 4.5.
- Accelerometer measurements from the IMU on the UAV and UGV, see Section 4.6.

- Barometer measurements from the IMU on both vehicles, see Section 4.7.

The sensors are sampling data at different frequencies, the sampled data has different accuracy and the sensors measure different quantities. In order to account for these differences, and combine the sensor measurements into estimates, an EKF is used. The EKF takes the sensor data and outputs an estimate of the relative position and velocity of the UAV with respect to the UGV in the {W}-frame. An overview of the estimation system is presented in Figure 4.1

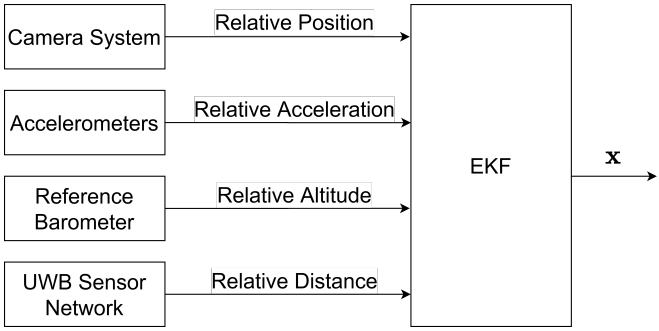


Figure 4.1: *Overview of the estimation system.*

4.2 Extended Kalman Filter

The EKF is a filter that estimates the states in a process based on a model as well as measurements of the process, see Section 3.4. The process examined in this case is the relative movement of the UAV with respect to the UGV. The estimated states are the relative position and velocity of the UAV with respect to the UGV, expressed in the {W} coordinate frame. The state vector $\mathbf{x}$ is therefore defined as

$$\mathbf{x} := \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T . \tag{4.1}$$

The input of the model, u, is the relative acceleration between the UAV and UGV. The relative acceleration is modelled with additive white Gaussian noise (AWGN) as follows

$$\mathbf{u} = \begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z} \end{bmatrix}^T + \epsilon_u, \quad \epsilon_u \sim \mathcal{N}(0,\ P_u) \tag{4.2}$$

The dynamical model used in the filter is based on the model used in [8]. However, instead of estimating the vehicles' individual movements, the relative movements between the vehicles are estimated. Additionally, this work assumes that the acceleration enters as an input instead of a state. The dynamical model is

$$\mathbf{x}_{t+1} = G\mathbf{x}_t + B\mathbf{u}_t + \epsilon, \quad \epsilon \sim \mathcal{N}(0,\ Q) \tag{4.3}$$

$$G = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \otimes I_{3x3}, \quad B = \begin{bmatrix} \frac{T_s^2}{2} \\ T_s \end{bmatrix} \otimes I_{3x3}, \quad Q = q \cdot \begin{bmatrix} \frac{T_s^5}{20} & \frac{T_s^4}{8} \\ \frac{T_s^4}{8} & \frac{T_s^3}{3} \end{bmatrix} \otimes I_{3x3} \tag{4.4}$$

where T_s is the period time of the filter, q a tuning constant and $\otimes$ the Kronecker product. The model assumes that the derivative of the acceleration is a zero-mean Gaussian white noise process with the same variance in all direction, i.e. $[x^{(3)}\ y^{(3)}\ z^{(3)}]^T \sim \mathcal{N}(0,\ q \cdot I_{3x3})$.

From (3.11a) and (3.11b) follows that the prediction step consists of the following equations:

$$\hat{\boldsymbol{\mu}}_t = G\boldsymbol{\mu}_{t-1} + B\mathbf{u}_t \tag{4.5a}$$

$$\hat{\Sigma}_t = G\Sigma_{t-1}G^T + Q + BP_uB^T \tag{4.5b}$$

The prediction step is usually executed with a frequency equal to the frequency of the sensor with the highest sampling rate. Therefore the frequency of the prediction step is set at 50 Hz, since the update rate of the accelerometer sensor is 50 Hz, see Section 4.6. The time period is therefore $T_s = 1/50$ s.

With the sensors measurement model in (3.10), (3.11c) – (3.11e) gives the following measurement step

$$K_t = \hat{\Sigma}_t H_t^T (H_t \hat{\Sigma}_t H_t^T + R)^{-1} \tag{4.6a}$$

$$\tilde{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\mu}}_t + K_t(z_t - h(\hat{\boldsymbol{\mu}}_t)) \tag{4.6b}$$

$$\tilde{\Sigma}_t = (I - K_tH_t)\hat{\Sigma}_t(I - K_tH_t)^T + KRK^T \tag{4.6c}$$

where $H_t = \frac{dh(\mathbf{x})}{d\mathbf{x}}\Big|_{\mathbf{x}=\mu_t}$ and K_t is the Kalman gain. Equation (4.6c) is written in Joseph form, see Section 3.4. The measurement step will be different for each sensor type since they have different models and measure different states. The sensor models used in the filter are described in the sections below.

As stated in Section 3.4.1, a sensor measurement might be an outlier and should in that case be removed. In order to detect and remove invalid sensor data the NIS test is used. The NIS is calculated as described in (3.13) and is approximaly χ^2-distributed. A 95%-confidence level is used to classify the measurements as an outlier or a valid measurement.

The initial guess of the z-position is taken as a mean from a series of altitude estimates provided by the barometers, see Section 4.7. The mean altitude value is denoted z_0. To calculate the initial guess in the x- and y-position (x_0, y_0), two series of UWB distance measurements are taken from each UWB anchor. From the measurements, the position is trilaterated using the theory in Section 3.5, only considering the horizontal position. The initial relative velocity is set to zero. Hence, the initial guess μ_0 is

$$\mu_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ \mathbf{0}_{3\times1} \end{bmatrix}. \tag{4.7}$$

The initial guess of the covariance was determined ad hoc to be

$$\Sigma_0 = 0.1\,\mathcal{I}_{6\times6}\,. \tag{4.8}$$

4.3 Attitude Measurements

The attitude of the vehicles is estimated with the flight control unit on the UAV and the IMU on the UGV. The attitude data is used to create a rotation matrix ${}^{\mathrm{W}}\mathcal{R}_{\mathrm{b}}$, which maps vectors in a body fixed coordinate system to the {W} coordinate frame as follows

$$\mathbf{x}^{\mathrm{W}} = {}^{\mathrm{W}}\mathcal{R}_{\mathrm{b}}\mathbf{x}^{\mathrm{b}}, \tag{4.9}$$

where $\mathbf{x}^{\mathrm{b}} \in \mathbb{R}^3$ is a vector expressed in a body fixed coordinate system and $\mathbf{x}^{\mathrm{W}} \in \mathbb{R}^3$ is the vector expressed in the {W} frame.

4.4 Ultra-Wide Band Sensor Network

The ultra-wide band (UWB) sensor network consists of one UWB tag mounted on the UAV and four UWB anchors mounted on the UGV, see Section 3.1. For convenience a definition is made: An anchor-tag pair designates an anchor and the tag, which is included in all pairs. With this definition there are four anchor-tag pairs in the system. The anchor and the tag in an anchor-tag pair communicates with each other; and the Euclidean distance between the anchor and the tag is measured. Distance measurements between each anchor-tag pair are provided at 10 Hz. Hence, the tag receives 40 measurements per second. These measurements are forwarded to the EKF with the same rate. An overview of the UWB sensor network can be seen in Figure 4.2.

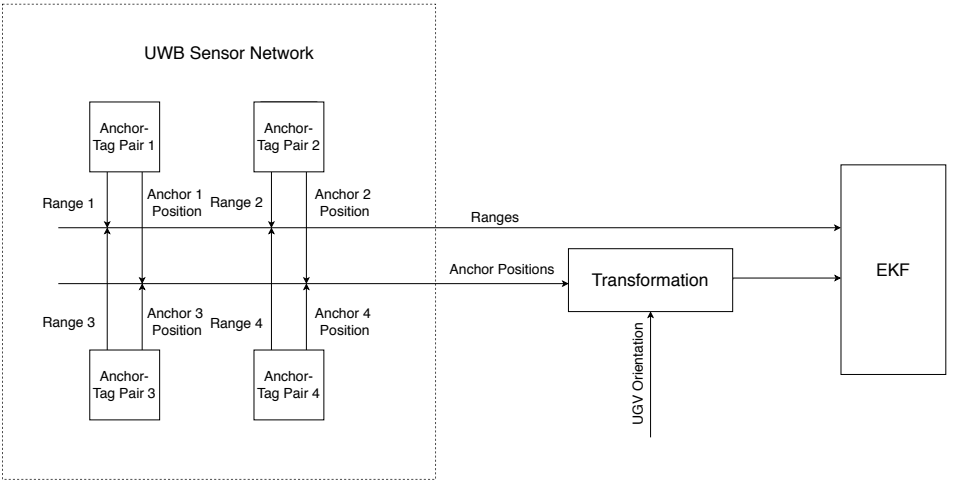


Figure 4.2: *Overview of the UWB sensor network. Each anchor-tag pair returns the range between them as well as the position of the anchor in the {UGV}-frame. The anchor positions are transformed to {W}-frame and sent with the ranges to the EKF.*

4.4.1 Sensor Model

Let the anchor in each anchor-tag pair correspond to a numerical index $i \in (1, 2, 3, 4)$. The position of anchor i is known in terms of an offset expressed in the {UGV}-frame, labelled as δ_i^{UGV}. Using the rotation matrix ${}^{W}\mathcal{R}_{\text{UGV}}$, the offset is transformed to the {W}-frame:

$$\delta_i^{\text{W}} = {}^{W}\mathcal{R}_{\text{UGV}}\delta_i^{\text{UGV}}. \tag{4.10}$$

Then, let the position of anchor i in the {W}-frame be $\mathbf{p}_i^{\text{W}}$ which is expressed as

$$\mathbf{p}_i^{\text{W}} = \mathbf{p}_{\text{UGV}}^{\text{W}} + \delta_i^{\text{W}}, \tag{4.11}$$

where $\mathbf{p}_{\text{UGV}}^{\text{W}}$ is the position of the UGV in the {W}-frame. The distance measurement d_i between one of the anchors and the tag (mounted on the UAV) can then be expressed as follows

$$d_i = \left\|\mathbf{p}_{\text{UAV}}^{\text{W}} - \mathbf{p}_i^{\text{W}}\right\|_2, \tag{4.12}$$

where $\mathbf{p}_{\text{UAV}}^{\text{W}}$ is the position of the UAV in the {W}-frame. Since $\mathbf{x}_{1:3} = \mathbf{p}_{\text{UAV}} - \mathbf{p}_{\text{UGV}}$, the distance d_i is

$$d_i = \left\|\mathbf{x}_{1:3} - \delta_i^{\text{W}}\right\|_2. \tag{4.13}$$

However, the distance measurement z_{UWB}^i from anchor-tag pair i is not without error. In [32], Gou et al. assumed a linear model between the measurement z_{UWB}^i and true distance d_i, i.e.

$$d_i = a_i z_{\text{UWB}}^i + b_i, \tag{4.14}$$

where a_i is a scaling factor and b_i is the bias. Additionally, assuming zero-mean AWGN, the sensor model of anchor-tag pair i is

$$z_{\text{UWB}}^i = h_{\text{UWB}}^i(\mathbf{x}) + \epsilon = \frac{\left\|\mathbf{x}_{1:3} - \delta_i^{\text{W}}\right\|_2 - b_i}{a_i} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, R_{\text{UWB}}^i), \tag{4.15}$$

where ϵ is error and R_{UWB}^i the (1×1) error covariance matrix of anchor-tag pair i. The parameters of each model (a_i, b_i, R_{UWB}^i) was determined empirically, which is described in the following section. The Jacobian of $h_{\text{UWB}}^i(\mathbf{x})$ is

$$H_{\text{UWB}}^i(\mathbf{x}) = \frac{dh_{\text{UWB}}^i(\mathbf{x})}{d\mathbf{x}} = \frac{1}{a}\left[\mathbf{x}_{1:3} - \delta_i^{\text{W}} \quad \mathbf{0}_{1x3}\right] / \left\|\mathbf{x}_{1:3} - \delta_{\text{W}}^i\right\|_2. \tag{4.16}$$

4.4.2 Model Parameter Tests

This section aims to present how the parameters of the sensor model, see (4.15), were determined for each anchor-tag pair. Tests were conducted to evaluate the performance and the characteristics of the range measurements from each anchor-tag pair. In these tests, the measured range was compared to a ground truth based on a measurement with a laser measuring tool. These tests were executed for each anchor-tag pair. For convenience sake, the four anchor-tag pairs are labelled P1, P2, P3 and P4 in this section.

Each test consisted of an anchor-tag pair (one UWB anchor and the UWB tag), and a laser measuring tool, which can measure distances with sub-centimetre precision. However, with the test setup used, unevenness of the ground could cause errors in the centimetre range. The test was conducted in open space outside to reduce effects from multipath propagation. The tag and anchor were each placed on top of cardboard boxes to reduce reflections from the ground, see Figure 4.3.

Figure 4.3: *Cardboard box with UWB tag to the left. Cardboard box with UWB anchor to the right.*

The test setup can be seen in Figure 4.4. The ground truth d is calculated as

$$d = d_l - d_{\text{offset}}\,. \tag{4.17}$$

where d_l is the distance from the laser measuring tool and d_{offset} is the distance between the laser and UWB tag. To achieve an adequate sensor model, the setup was repeated at 20 different distances, 1-30 m. At each distance the true distance d was measured and 100 samples of the range measurement r were collected from the UWB tag. The measurements were then repeated for all anchor-tag pairs. P1 was tested an additional time to examine if the model parameters vary between tests.

As described in (4.15), the relation between the range measurement r and ground truth d is

$$r = \frac{d-b}{a} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, R_{\text{UWB}})\,. \tag{4.18}$$

The parameters (a, b) can be estimated with least squares regression. To estimate

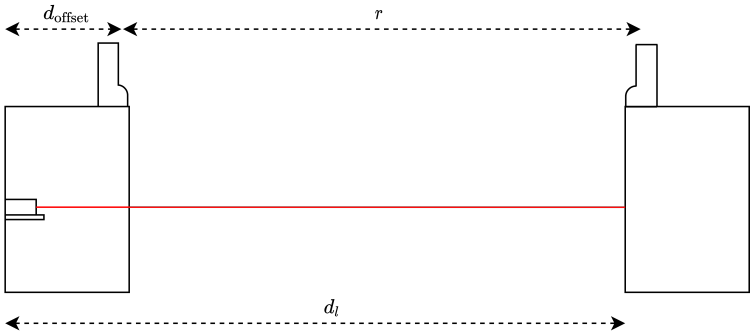


Figure 4.4: *Model parameter test setup. The UWB anchor-tag pair measures the distance r between them. The laser measuring tool measures the reference distance to the cardboard box d_l, which is used as an estimate of the ground truth.*

(a, b) from the measurements from a test, the following matrices are formed

$$A_r = \begin{bmatrix} r_1^1 & 1 \\ \vdots & \vdots \\ r_{100}^1 & 1 \\ \vdots & \vdots \\ r_1^{20} & 1 \\ \vdots & \vdots \\ r_{100}^{20} & 1 \end{bmatrix}, \quad \mathbf{x}_{\text{par}} = \begin{bmatrix} a \\ b \end{bmatrix}, \quad \mathbf{y}_d = \begin{bmatrix} d^1 \\ \vdots \\ d^1 \\ \vdots \\ d^{20} \\ \vdots \\ d^{20} \end{bmatrix} \tag{4.19}$$

where r_j^k is the j:th sample from the range measurements at distance d^k. The parameters are then calculated as

$$\mathbf{x}_{\text{par}} = (A_r^T A_r)^{-1} A_r^T \mathbf{y}_d \,. \tag{4.20}$$

The parameters were estimated for each of the five tests, see Table 4.1. The bias b can vary between tests, with estimated bias values between $1 - 10$ cm. Furthermore, examining the separate tests with anchor-tag pair P1 the results are not consistent over time. We conclude that estimating the model of each individual anchor-tag pair will not be a fruitful endeavour.

Table 4.1: *Estimated parameters a and b from the data from each test.*

anchor-tag pair	a	b
P1, test 1	1.0030	0.0820
P1, test 2	1.0043	0.0130
P2	1.0035	0.0348
P3	1.0033	0.0659
P4	1.0015	0.1003

However, the pattern of the estimated parameters is a positive bias b as well as a scaling parameter a around 1.00. We therefore conclude that a linear model could still improve the measurements from the UWB-sensors. The solution is to estimate a and b with the data from every test. The resulting parameters are:

$$a = 1.0032, \quad b = 0.0580 \tag{4.21}$$

To estimate the covariance matrix R_{UWB}, (4.20) is rewritten as

$$\epsilon = r - \frac{d-b}{a}\,. \tag{4.22}$$

Since R_{UWB} is a 1×1 covariance matrix, it is simply the variance of ϵ. Therefore, R_{UWB} is estimated as

$$R_{\text{UWB}} = \frac{1}{N-1}\sum_{i=1}^{N} r_i - \frac{d_i - b}{a} = 0.0015\,, \tag{4.23}$$

where N is the total number of measurements.

To illustrate the resulting sensor model, the error ϵ from each sample versus the true distance at the samples is scatter-plotted. The line $(a-1)r+b$ representing expected error based on the model is plotted as well. The result is presented in Figure 4.5.

4.4.3 Anchor Configuration

The placement of the UWB anchors on the UGV has an impact on the information gain from the sensors. Therefore, the placement of the anchors was carefully thought out so as to maximise the positional information gained from a measurement. This section describes how the anchor configuration was chosen, yielding a favourable geometry but still adhering to the practical limitations of the UGV landing pad.

As described in Section 3.4.2, the CRLB can be used to indicate the information gain of a measurement, with a lower CRLB indicating more information. Therefore, different anchor configurations are compared based on their CRLB. The CRLB can be calculated from a linear sensor model, see (3.15).

As described in Appendix A.3, the measurements from all anchor-tag pairs are sent simultaneously. They are merely handled separately in the EKF. Therefore, a sensor model where all measurements are concatenated is used. The

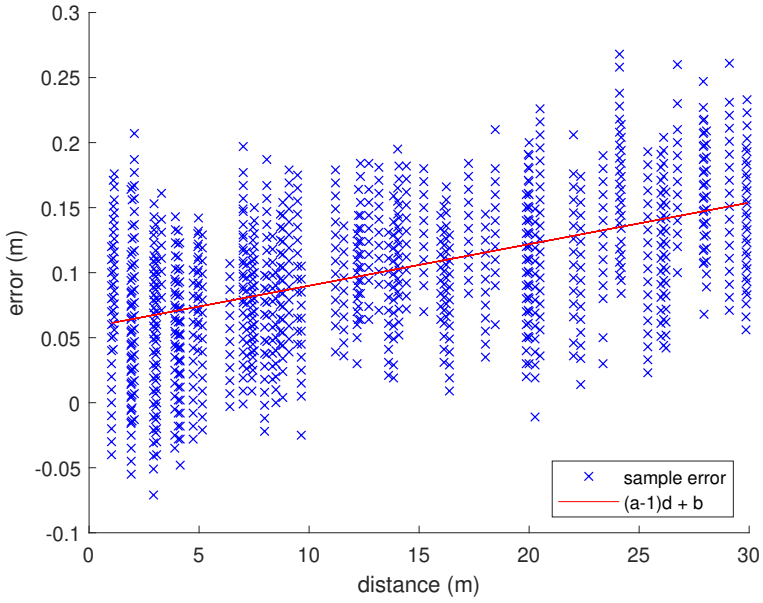


Figure 4.5: *The sample errors for all model parameter tests plotted against distance.*

model is derived from (4.15),

$$\mathbf{z}_{\text{UWB}} = \begin{bmatrix} z^1_{\text{UWB}} \\ z^2_{\text{UWB}} \\ z^3_{\text{UWB}} \\ z^4_{\text{UWB}} \end{bmatrix} = \begin{bmatrix} h^1_{\text{UWB}}(\mathbf{x}) \\ h^2_{\text{UWB}}(\mathbf{x}) \\ h^3_{\text{UWB}}(\mathbf{x}) \\ h^4_{\text{UWB}}(\mathbf{x}) \end{bmatrix} + \mathbf{e}, \quad \mathbf{e} \sim \mathcal{N}(0, R_{4\times4}), \tag{4.24}$$

where $R_{4\times4} = R_{\text{UWB}}\mathcal{I}_{4\times4}$ is the covariance matrix of the measurement error $\mathbf{e}$ and R_{UWB} is the variance from a UWB range measurement determined in the previous subsection. The model in (4.24) is nonlinear and is therefore linearized using the first order term from the Taylor series expansion at a state $\mathbf{x}_j$. The resulting model is

$$\mathbf{y} = \begin{bmatrix} H^1_{\text{UWB}}(\mathbf{x}_j) \\ H^2_{\text{UWB}}(\mathbf{x}_j) \\ H^3_{\text{UWB}}(\mathbf{x}_j) \\ H^4_{\text{UWB}}(\mathbf{x}_j) \end{bmatrix} \mathbf{x} + \mathbf{e}\,. \tag{4.25}$$

When evaluating different anchor configurations, it is not sufficient to compare the CRLB at a single state $\mathbf{x}_j$. However, a number of states can provide a statistically significant benchmark. Since the UWB measurements only give positional information, the velocities are irrelevant and set to zero. Also, since the UWB sensor network will be used to estimate positions above ground ($z \geq 0$), negative z-positions are not studied. In order to achieve a statistically significant

benchmark, points are randomly placed on a semi-sphere with a fixed radius r. In this case $N = 10000$ points are placed out. Figure 4.6 illustrates the semi-sphere where $r = 10$ m. For each corresponding state $\mathbf{x}_j$, $\text{CRLB}(\mathbf{x}_j)$ is calculated. The element $\text{CRLB}(\mathbf{x}_j)_{1:1}$ in the CRLB-matrix is the lower bound of the variance of the estimated x-position. The lower bound $\text{CRLB}(\mathbf{x}_j)_{1:1}$ is saved for each state and a mean $\overline{CRLB_x}$ is calculated. The mean is an indication of the positional information in the x-direction at the current semi-sphere radius r is. The CRLB-means for the estimation of the y- and z-position are calculated as well.

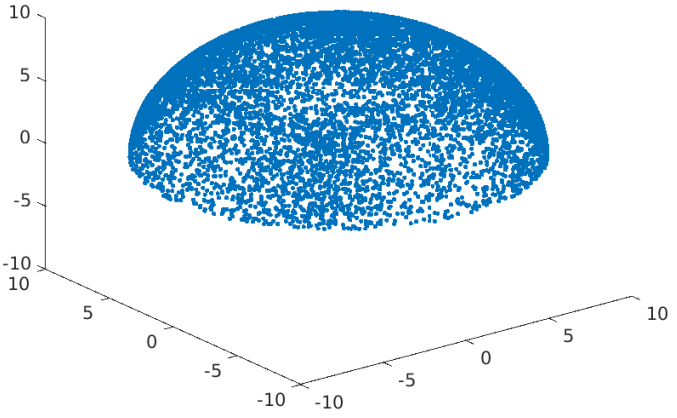
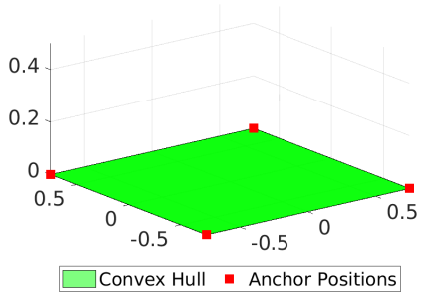


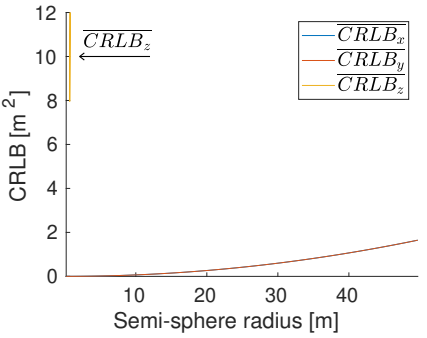
Figure 4.6: *10000 points randomly placed on a semi-sphere with a radius $r = 10$ m.*

Before deciding on an anchor configuration, the limitations of the landing platform of the UGV needs to be considered. The shape of the landing platform is described in Section 2.2. Initial tests in MATLAB indicated that a larger convex hull bounded by the anchor positions would lead to a better estimation. Therefore, the anchors should be placed at the extremities of the landing pad, i.e. on the corners of the square. The only variation is in the height of the anchors, when compared to the base of the landing pad. From these specifications, three different potentially interesting configurations were chosen: all anchors on the base of the pad (Conf 1), three on the pad and one placed at the top of a beam (Conf 2) and lastly two on the pad and two on top of a beam (Conf 3). The corresponding anchor positions of Conf 1, 2 and 3 as well as their convex hulls can be seen in Figures 4.7a, 4.7c and 4.7e.

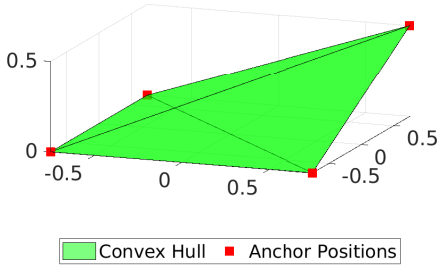
To compare configurations, the CRLB-means $\overline{CRLB_x}$, $\overline{CRLB_y}$ and $\overline{CRLB_z}$ were calculated for semi-sphere radii between 0.1 and 50 m. The results from each configuration are shown in Figures 4.7b, 4.7d and 4.7f.



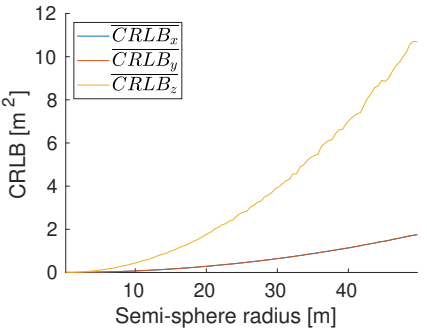
(a) Conf 1.



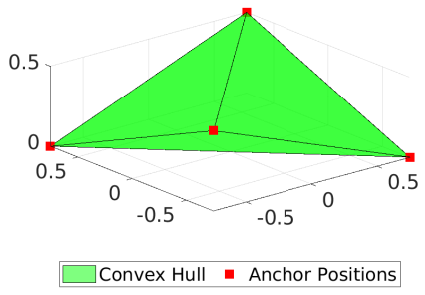
(b) CRLB-means over distance, Conf 1.



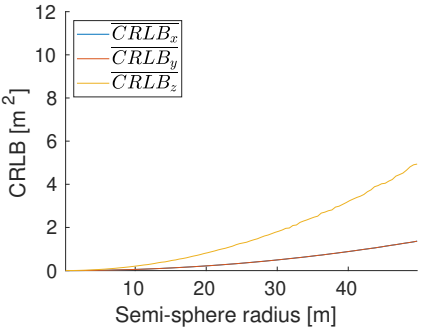
(c) Conf 2.



(d) CRLB-means over distance, Conf 2.



(e) Conf 3.



(f) CRLB-means over distance, Conf 3.

Figure 4.7: *The points and convex hulls corresponding to each anchor configuration. For each configuration, the CRLB-means are also shown.*

From Figures 4.7b, 4.7d and 4.7f it is observed that the $\overline{CRLB_y}$ is basically identical with $\overline{CRLB_x}$, i.e. the CRLB-mean is identical in each horizontal direction. Another observation is that the CRLB-means seem to grow quadratically with distance. When studying Figure 4.7b it is clear that Conf 1 leads to essentially no positional information in the z-direction at any radius. When comparing Figures 4.7b and 4.7d, it can be seen that Conf 2 leads to slightly more positional information of the horizontal positions. However, the positional information of Conf 2 in the z-direction is worse when compared to Conf 3. The conclusion is to choose Conf 3 as anchor configuration.

As stated previously, the CRLB-means seem to grow quadratically with radius r. To study Conf 3 further, we define the azimuth angle and inclination angle in the spherical coordinate system as $\alpha \in [-\pi, \pi]$ and $\beta \in [0, \pi]$, respectively. The radius is fixed to $r = 10$ m, and for each point, α and β are calculated. From the calculations as well as the CRLB of each point, the relation between α and CRLB as well as β and CRLB is shown in Figures 4.8 and 4.9.

When analysing Figure 4.8, we first notice the spread of points for each value of α, caused by the varying β-angle. The opposite behaviour can be seen in Figure 4.9. However, there is still a clear pattern. When the α points in the x-direction, more information about the x-position is gained. This is the same for the y-position. A similar behaviour can also be seen in Figure 4.9. When β points more in the z-direction, the positional information of the z-position is better. Furthermore, we conclude that α has no effect on the position information from the z-position.

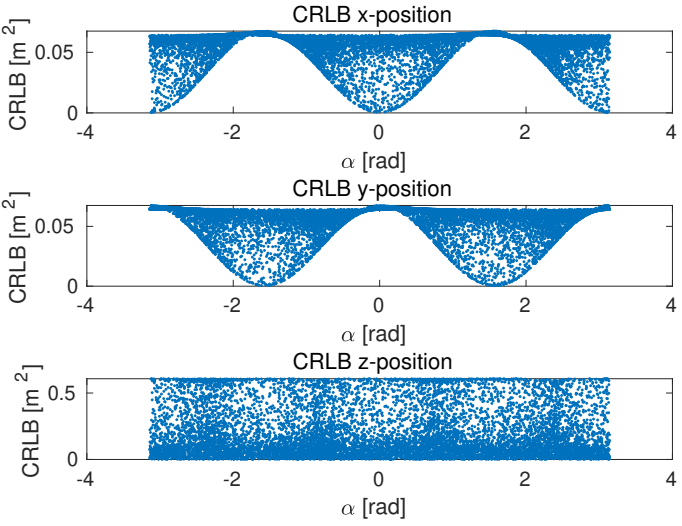


Figure 4.8: *CRLB over azimuth angle α. Calculated from 10000 points randomly placed on a semi-sphere with a radius $r = 10$ m.*

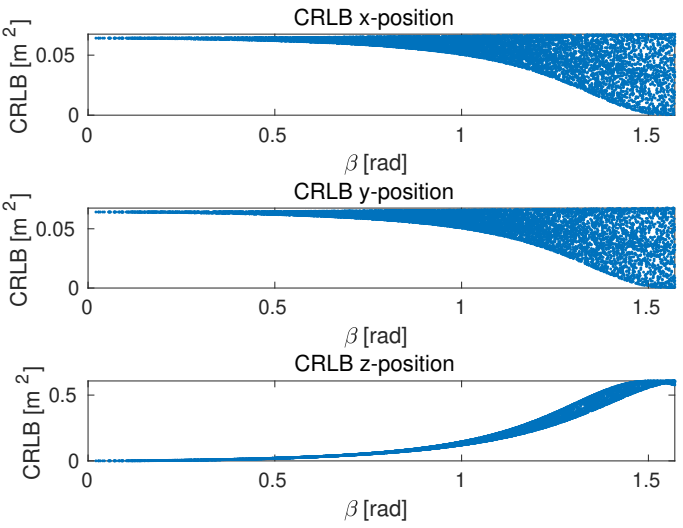


Figure 4.9: *CRLB over inclination angle β. Calculated from 10000 points randomly placed on a semi-sphere with a radius $r = 10$ m.*

4.5 Camera System

This section aims to describe the camera system, which estimates the position of the UGV with respect to the UAV. The estimated position is expressed in the {W} coordinate frame and the image data is captured with a calibrated camera. The camera system can only provide estimates when the UAV is above and in a close proximity to the UGV since the camera is facing downwards. This section contains information about how the position estimation is executed and how the camera is calibrated. It also presents performance results of the camera system, using both a simulated and a real camera.

The camera system is divided into four separate parts:

- A calibrated downwards facing camera mounted on the UAV.
- A fiducial detection algorithm that detects and estimates the position of a fiducial marker in image data.
- A fiducial marker with known dimensions mounted on the UGV.
- A coordinate transform that transforms positions from the camera's reference frame to the {W}-frame.

The fiducial detection algorithm processes images from the camera. If the marker mounted on the UGV is detected in an image its position is estimated. The marker's estimated position is expressed in the camera's reference system and needs to be transformed to global coordinates, which is done with a coordinate transform.

See Figure 4.10 for an overview. These four components are each described in the following four sections.

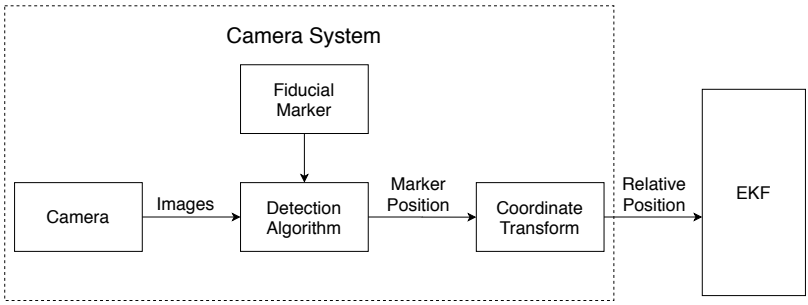


Figure 4.10: *Camera system overview.*

4.5.1 Camera

As described in Section 2.1, a camera is mounted on the UAV. The camera is modelled as a pinhole camera, see Section 3.6. The parameters of the pinhole model were determined through a calibration procedure.

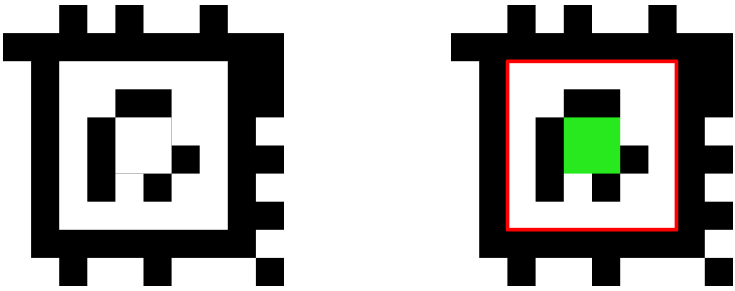
The calibration procedure is explained in [33]. In essence, the procedure involves holding a checkerboard with known dimensions in front of the camera. The checkerboard is moved between different poses while the camera is taking pictures of it. The checkerboard contains 9x7 squares, however, it is the 8x6 interior vertices that are being used by the calibration algorithm. When enough images of the checkerboard have been taken, the calibration algorithm solves an optimisation problem in order to obtain an estimate of the camera parameters.

4.5.2 Fiducial Detection Algorithm

A fiducial detection algorithm is an estimation method that estimates the position of a fiducial marker relative to a camera. The research area regarding fiducial algorithms is widespread and there are many open-source algorithms that are robust and well tested. One such algorithm is called AprilTag, which is used in this project. AprilTag was created by the APRIL Robotics Laboratory at The University of Michigan. The system was initially presented in [34] and further developed in [35] and [36]. AprilTag was chosen because it has a ROS implementation and could therefore be easily integrated with the rest of the system. Initial tests in simulation also proved that the AprilTag algorithm had adequate performance.

4.5.3 Fiducial Marker

A fiducial marker is an artificial object with a unique design, which can be identified in image data and used as a reference. In the specific case of the camera



(a) Original tag. *(b)* Tag showing the different regions.

Figure 4.11: *Description of the regions of a tag from the TagCustom48h12-family. The green area corresponds to the part of the tag not used in the detection. The part of the tag within the red lines is the estimation area.*

system the fiducial marker is used to estimate the UGV's position with respect to the camera mounted on the UAV. This estimation is possible since the size of the fiducial marker is known. The marker is mounted on top of the UGV, see Section 2.2.

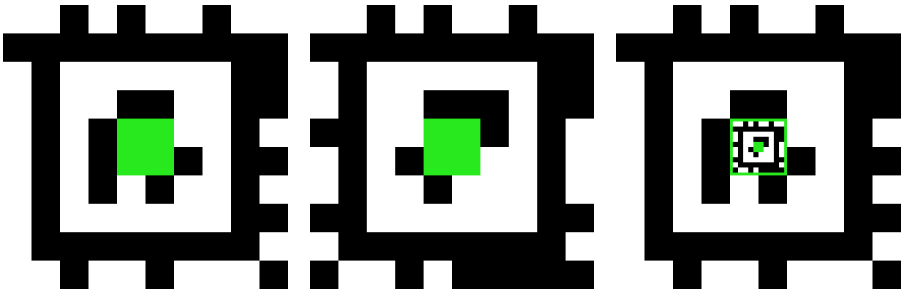
AprilTag can detect a variety of tags, known as tag families[8]. Variations between families can come in size, appearance and other properties. A tag family contains multiple tags, each identified by their individual ID (positive integer). The tag family used in this project is called *TagCustom48h12*, which was designed in [37]. A tag can usually be divided into two parts: One used for the identification and used for the position estimation, formally called the *estimation area*. The tags from the *TagCustom48h12*-family also contain a space in the centre which is not used in neither the identification nor the estimation process, see Figure 4.11. This enables for the creation of a *recursive tag*, which herein is defined as:

A recursive tag is a tag with an inner tag placed in the centre. The inner tag can also hold another tag placed in its centre, increasing the level of recursion.

The advantage of recursive tags is that tags with different sizes can be combined into a new tag with an extended range of detection compared to the individual tags. By range of detection the authors refer to the interval between the minimum and maximum distance at which a tag can be detected with a camera. One of the governing quantities for the range of detection is the size of the tag, since it determines at which distances it can be seen in in the FOV of the camera. When a smaller tag is placed within a larger tag, the combined tag can be seen between the maximum distance of the larger tag and the minimum distance of the smaller tag, given that there is no gap between the tags' range of detection.

The recursive tag on the UGV is a combination of two tags in the *TagCustom48h12* family. The tags have IDs 0 and 1, see Figure 4.12a and 4.12b. The

[8] A description of the different kinds of AprilTag families can be seen at: https://april.eecs.umich.edu/software/apriltag.html



(a) Outer tag with ID 0. *(b)* Inner tag with ID 1. *(c)* Recursive tag.

Figure 4.12: *Description of the recursive tag consisting of two tags combined into one. The green area corresponds to the part of the tag not used in the detection.*

dimensions of the tags as well as their estimation areas are presented in Table 4.2. The tag with ID 1 is placed inside the tag with ID 0 as explained in Figure 4.12.

Table 4.2: *Tag descriptions.*

	Dimension	Estimation area
Tag 0	1.4×1.4 m^2	0.84×0.84 m^2
Tag 1	0.252×0.252 m^2	0.151×0.151 m^2

4.5.4 Coordinate Transform

The coordinate transform layer transforms positions in the camera's frame of reference to the {W}-frame. First, the estimated position from the AprilTag algorithm, $\mathbf{p}^{\text{cam}}$, is transformed to the {UAV}-frame via the rotation matrix ${}^{\text{UAV}}\mathcal{R}_{\text{cam}}$. Secondly, a rotation is required from the {UAV}-frame to the {W}-frame, where the rotation matrix is denoted ${}^{\text{W}}\mathcal{R}_{\text{UAV}}$. The coordinate transform layer therefore performs the following transformation

$$\mathbf{p}^{W} = {}^{\text{W}}\mathcal{R}_{\text{UAV}}\,{}^{\text{UAV}}\mathcal{R}_{\text{cam}}\mathbf{p}^{\text{cam}}, \tag{4.26}$$

where $\mathbf{p}^{W}$ is the estimated position from the AprilTag algorithm expressed in the {W}-frame.

4.5.5 Error Model

The camera system's performance was evaluated, both in simulations and experimentally. First, the range of the camera system was evaluated in Gazebo, see Figure 4.13. The camera was moved in a grid-like pattern in order to examine where it could detect and estimate the positions of the AprilTags. Figure 4.14 shows that the combination of a small and a large tag allows an extended range

of detection. The small tag has a slightly wider area of detection, which is motivated by the fact that parts of the larger tag disappears out of the FOV before the small tag. The range of detection for the small tag was experimentally confirmed using the camera mounted on the UAV. The tag could be detected at distances up to 8 m.

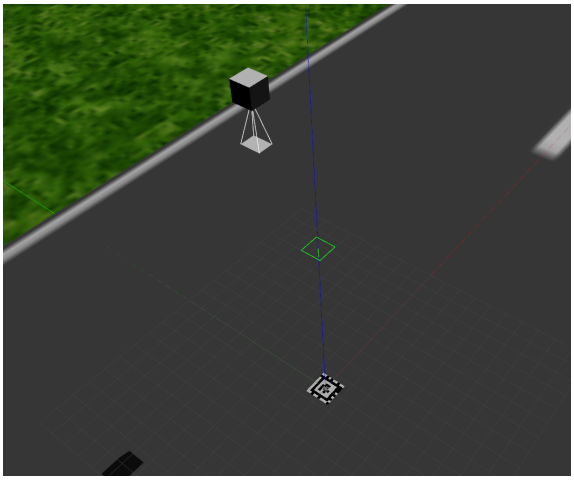


Figure 4.13: *Illustration of the camera simulation environment. The camera is mounted underneath the box and the AprilTag can be seen on the ground. The AprilTag has the same dimensions as the tag on the UGV.*

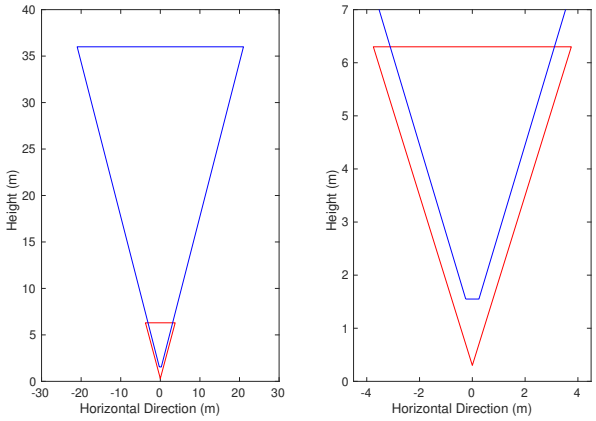
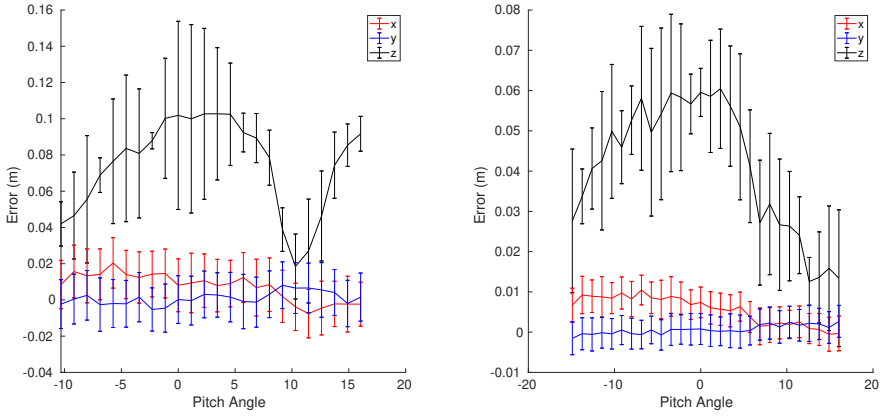


Figure 4.14: *Detection range of the small and the large tag. The blue and red line indicate the areas where the large and small tags can be detected, respectively. The right figure is a zoomed-in version of the left figure.*

The positioning accuracy was also examined with several simulations:

- For a given position and orientation in the simulation environment, 100 position estimates of the tag were sampled using the camera system. The bias and the standard deviation are computed using the samples.
- Between each sample the camera is moved randomly in the horizontal plane. The length of the movement is drawn from a uniform distribution. The limits of the distribution correspond to the tag moving two pixel lengths. This is done to make sure that the tag is not seen in the exact same pixels for each sample.

First, the impact of the pitch angle of the camera is investigated. Figure 4.15 displays the bias and standard deviation when the pitch angle is varied for a typical position. The pitch angle mostly affects the accuracy of the z-estimate, and the highest error occurs when the camera is parallel with the tag. The maximum bias in the height estimate is approximately 10 cm for the large tag and 6 cm when using the small tag. The same applies for the roll angle. In the subsequent tests the camera is parallel with the tag to obtain a conservative estimate of the error.



(a) *Large tag, camera at position (0.5, 0, 5) (m).*

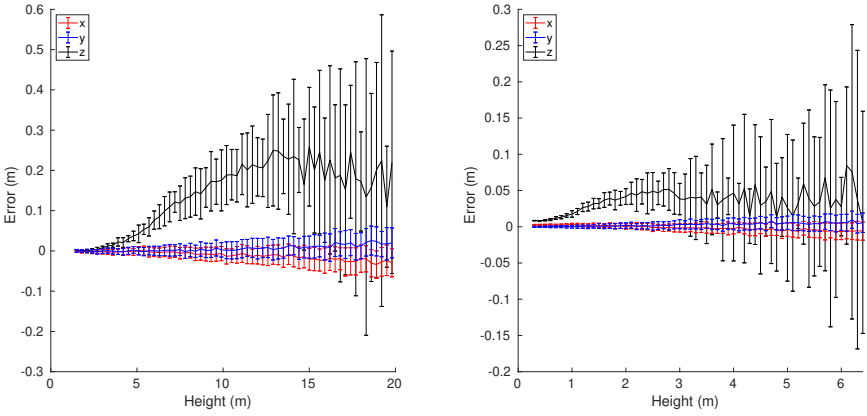
(b) *Small tag, camera at position (0.2, 0, 2) (m).*

Figure 4.15: *Bias and standard deviation when varying the pitch angle of the camera relative to the tag. The angles vary between $-17.3°$ to $17.3°$, which are the maximum allowed attitude, see Section 5.3. For the largest negative pitch angles the tags were not entirely inside the FOV of the camera.*

The next test investigates the estimation quality when the distance along the z-axis varies, and the results can be seen in Figure 4.16 where it is apparent that the height estimate is affected the most. Figure 4.17 shows the bias and the standard deviation for the position estimate when the camera's x-position varies at

a constant height. The standard deviation and bias from the y-estimate is not affected when the camera is moving in the x-direction. The bias and the standard deviation in the x-direction grows as the relative x-distance increases. It has been confirmed that the opposite applies when the y-position varies.

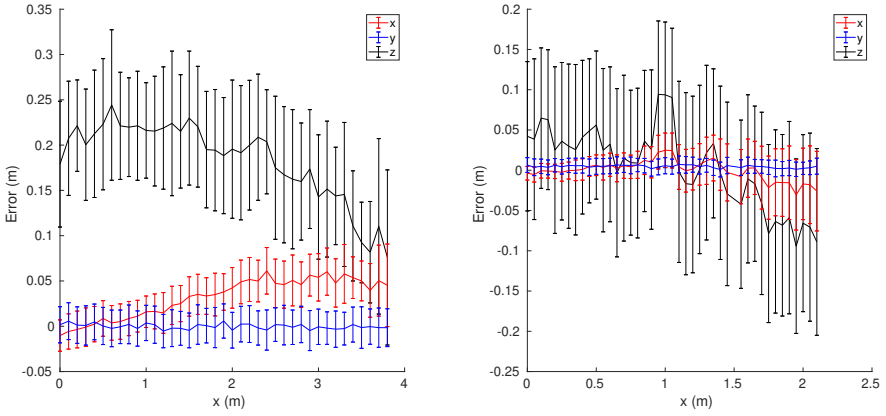
The highest altitude the UAV flies at during the landing manoeuvre is 10 m. At 10 m the bias in the z-direction for the large tag is around 20 cm and quickly decreases as the altitude gets smaller. The bias in the horizontal plane is in the order of a couple of cm throughout all the test. Comparing these biases to the precision of the camera sensor placement, which is also in the order of a couple of cm, it is concluded that they are of similar size. Therefore, the bias is negligible.



(a) *Large tag.* ***(b)*** *Small tag.*

Figure 4.16: *Bias and standard deviation when varying the height of the camera. The camera is parallel with the tag and the x and y position is 0.*

However, a variance model is constructed for the camera system since it is required by the EKF. New data is generated for the variance model where the camera is moved to positions in the area spanned by the range of detection. Additional uncertainty is introduced in the attitude of the camera to emulate a more realistic system. This uncertainty increases the variance of the position estimate, but the bias remains the same. Linear regression is used to generate a variance model for the x-,y- and z-estimate. From the previous results it is concluded that the variance of the x-estimate is coupled with the altitude and the offset in the x-direction. The analogous applies for the y-estimate. The variance of the z-estimate is coupled with the horizontal distance and the altitude. The type of regressors are chosen based on the shape of the variance surface. The regressors can be seen in Table 4.3. An example of the variance surface and the prediction of the corresponding model can be seen in Figure 4.18.



(a) *Large tag at a height of 10 m.* ***(b)*** *Small tag at a height of 5 m.*

Figure 4.17: *Bias and standard deviation when varying the x-position of the camera at a constant height. The camera is parallel with the tag, the y position is 0.*

Table 4.3: *Linear regressors for the variance in the x-,y- and z-direction. The variable r represent the horizontal offset.*

Variance Axis	Regressors
x	$1, \quad x, \quad z, \quad x \cdot z, \quad x^2, \quad z^2$
y	$1, \quad y, \quad z, \quad y \cdot z, \quad y^2, \quad z^2$
z	$1, \quad r^2, \quad z^2, \quad z^4$

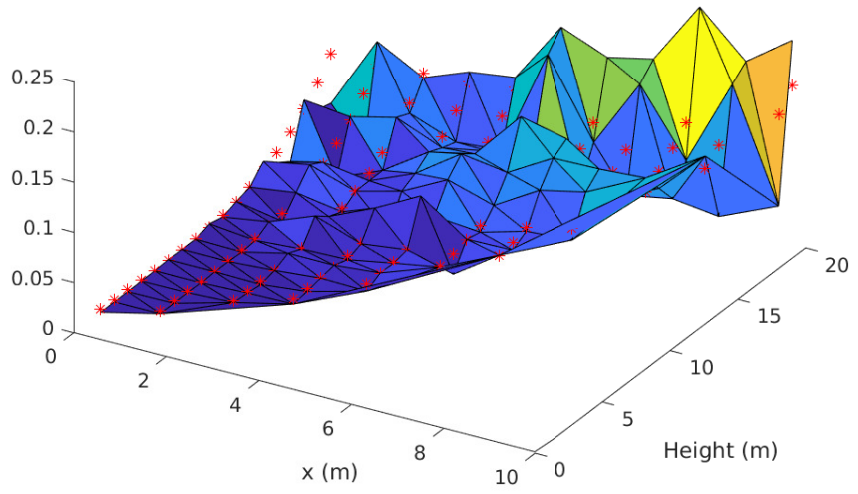


Figure 4.18: *Illustration of the variance surface when the x- and z-position changes. The red stars indicate the prediction from the linear regression model.*

4.6 Accelerometer

The estimation system uses relative acceleration data as input in each time update step. In order to obtain relative acceleration data, both the UAV and the UGV are equipped with an IMU. The IMU contains an accelerometer, which measures acceleration in the vehicle's local coordinate frame. The acceleration data is low-pass filtered and down-sampled from 400 Hz to 50 Hz before being passed to the EKF. An overview of the relative acceleration computation can be seen in Figure 4.19

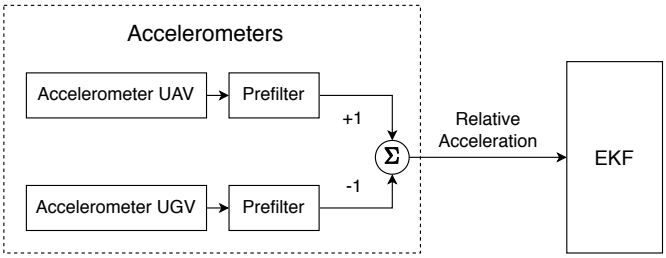


Figure 4.19: *Overview of the acceleration data processing. The two IMU's attached to the vehicles acquired acceleration measurements in 400 Hz. The prefilter block represent the low-pass filtering and downsampling of the measurements. After the prefilter block the relative acceleration is computed and passed to the EKF.*

4.7 Reference Barometer

A barometer measures the pressure in the atmosphere, which in turn can be converted to an altitude and vice versa, see Section 3.9. See Figure 4.20 for an overview of the relative altitude estimate. The relative altitude measurement is used to aid the position estimate from the UWB sensors.

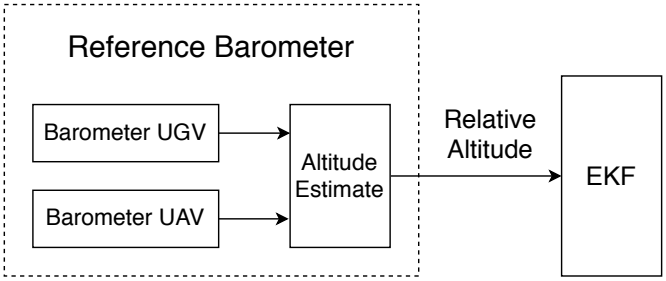


Figure 4.20: *Barometer data from the UAV and the UGV is to compute an estimate of the relative altitude, which in turn is passed to the EKF.*

4.7.1 Sensor Model

The measured pressure data from the barometers are converted to a relative altitude according to (3.26), which is restated here

$$z_{\text{rel}} = z_{\text{a}} - z_{\text{b}} = \frac{T}{L}\left[\left(\frac{P_{\text{a}}}{P_{\text{b}}}\right)^{-\frac{LR}{g}-1}\right] = \mathbf{h}(P_a, P_b, T_b)\,. \tag{4.27}$$

where P_a is the pressure from the UAV's barometer, P_b is the pressure from the UGV's barometer and T is the air temperature. It is assumed that the converted altitude is under the influence of AWGN. The sensor model is

$$\mathbf{z} = z_{\text{rel}} = [\mathbf{0}_{1\times 2}\ 1\ \mathbf{0}_{1\times 3}]\,\mathbf{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, R_{\text{baro}})\,, \tag{4.28}$$

where R_{baro} is the covariance matrix.

4.7.2 Error Model

In order to estimate the variance of the noise in the sensor model, see (4.28), the noise of the real barometers are investigated. The sensor test is performed as follows:

- Both barometers are put on the floor for 100 seconds.
- The first barometer is lifted to a height of 1.94 meters and is left there for 100 seconds.
- The first barometer is then placed on the floor again and is left there for another 100 seconds.

Figure 4.21 shows the data from the test. The pressure data is used to compute a height estimate which is presented in Figure 4.22. Table 4.4 presents statistical data from the relative height estimate. From the data presented in the table, R_{baro} is set to 0.2 m^2.

For simulation purposes it is also interesting to know the variance of the barometer. The variance of the first 1000 data points from the pressure data gives a variance of 11.56 and 13.89 for barometer 1 and 2 respectively.

Table 4.4: *Statistics of the relative barometer height estimates.*

Interval	Mean Altitude (m)	Mean Deviation (m)	Variance (m^2)
1	-0.0934	0.0934	0.2105
2	1.6894	0.2506	0.2371
3	-0.0199	0.0199	0.1891

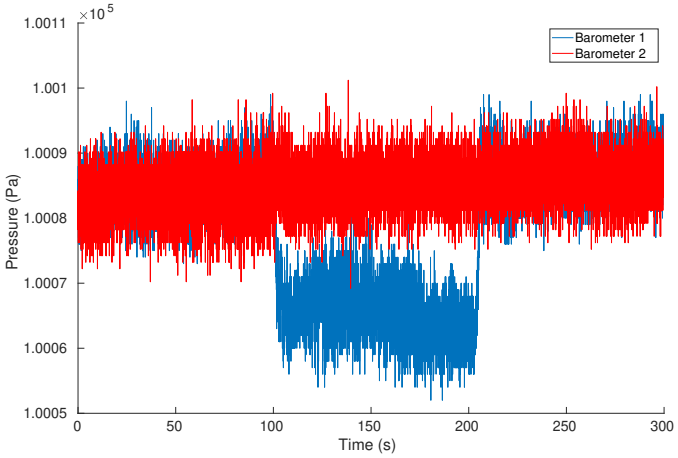


Figure 4.21: *Pressure data from the two barometers. Red data correspond to the barometer on the ground and blue data correspond the barometer that is lifted after 100 seconds.*

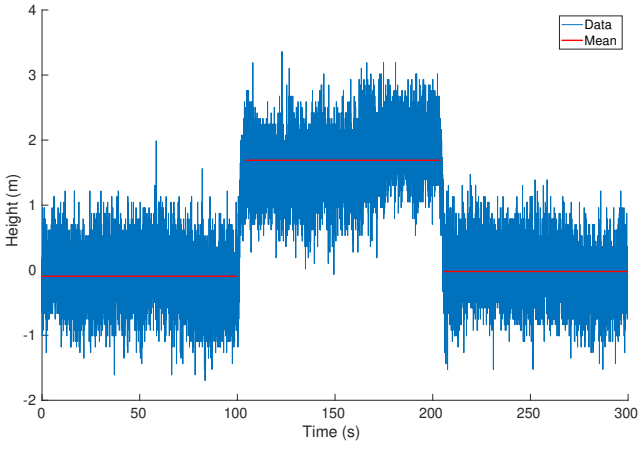


Figure 4.22: *Relative height estimates computed from the pressure data in figure 4.21. The red lines indicate the mean height for each 100 s time interval.*

5

Control System

This chapter describes the subsystem that is responsible for computing appropriate control outputs for the UAV to solve the landing task. The subsystem is named control system.

5.1 Control System Description

The overall control goal of the control system is to land the UAV on top of the UGV. In terms of quantities this is the same as saying that the position of the UAV should be equal to the position of the UGV or equivalently that the relative distance between the vehicles is zero. Let $\mathbf{p}_{\text{UAV}} \in \mathbb{R}^3$ define the position of the UAV in the {W}-frame and $\mathbf{p}_{\text{UGV}} \in \mathbb{R}^3$ define the position of the UGV in the {W}-frame. Then let the relative position vector between the vehicles be defined as

$$\mathbf{p}_{\text{rel}} = \mathbf{p}_{\text{UAV}} - \mathbf{p}_{\text{UGV}} \,. \tag{5.1}$$

The 2-norm of the relative position between the vehicles is a natural choice of error signal for the control system. Let e represent the error signal, i.e.

$$e = \|\mathbf{p}_{\text{rel}}\|_2 \,. \tag{5.2}$$

The underlying control goal is then to minimise e.

Since the landing manoeuvre is assumed to take place in an open, relatively flat environment the control system does not need to account for any obstacles except the ground and the UGV. This means that the UAV can move in any direction as long as it is above the ground and the UGV. Hence, no obstacle avoidance algorithm is implemented in this work. However, to ensure that the UAV does not descend too early and collides with the UGV when approaching it, a reference offset is introduced. The error signal is therefore extended as

$$e = \|\mathbf{p}_{\text{d}} - \mathbf{p}_{\text{rel}}\|_2 \,, \tag{5.3}$$

where $\mathbf{p}_d = [0, 0, h_d]^T$ and h_d is an altitude offset which is piecewise constant. The altitude offset h_d is set to 0 when the UAV can descend safely onto the UGV.

As mentioned in Section 2.3.2 the UAV runs the open source autopilot software ArduCopter. The autopilot accepts attitude targets and a climbing rate[9] which are translated into low level control actions, i.e. the rotational speed of each motor. These quantities are controlled via feedback from the sensors attached to the flight control unit.

The attitude target is accepted as a normalised quaternion vector $\mathbf{q}$ that describes the orientation of the {UAV}-frame relative to the {W}-frame. When the {UAV}-frame is aligned with the {W}-frame the quaternion vector $\mathbf{q} = [0, 0, 0, 1]^T$. The vector $\mathbf{q}$ can be transformed into the Euler angles roll (ϕ), pitch (θ) and yaw (ψ) which have a more direct interpretation. As described in Section 3.3 a roll angle and a pitch angle cause movements in the y and x-direction respectively in the {UAV}-frame. If the target yaw angle is zero, a pitch and a roll angle in the {UAV}-frame correspond to the same roll and pitch angles in the {W}-frame. Hence, with a yaw angle of zero the movements in the {UAV}-frame equal the movements in the {W}-frame. Additionally, the yaw angle does not have an impact on the control or estimation system. Therefore, the yaw angle is set to zero in the control system.

The climbing rate c_r describes the vertical movements of the UAV and should be in the interval $[0, 1]$. The values of c_r have the following interpretation

$$\begin{cases} c_r < 0.5: \text{ Descending, maximum descent rate when } c_r = 0 \\ c_r = 0.5: \text{ Hovering} \\ c_r > 0.5: \text{ Ascending, maximum ascent rate when } c_r = 1 \end{cases}$$

The maximum ascent and descent velocity is determined by a selectable parameter in the autopilot software.

Hence, the task of the control system is to compute the control commands (ϕ, θ, c_r) from the current relative position, $\mathbf{p}_{\text{rel}}$ together with the desired height, h_d and then pass them to the ArduCopter autopilot. Since the autopilot requires a quaternion vector the Euler angles are converted before they are passed onwards, see (3.5a)-(3.5d). Figure 5.1 shows an overview of the control system.

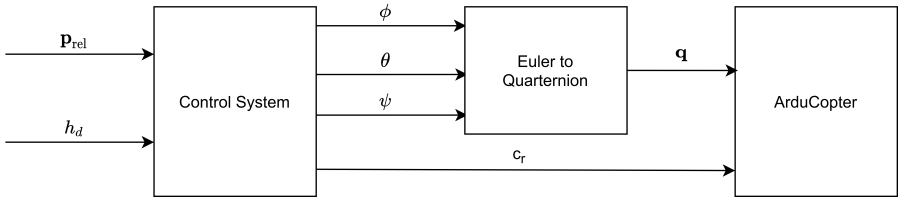


Figure 5.1: *Overview of the control system.*

To summarise, the control system should drive the relative position between

[9] When used in the `GUIDED_NOGPS` flight mode, however, ArduCopter supports a variety of flight modes, see https://ardupilot.org/copter/docs/flight-modes.html.

the vehicles to zero by controlling the attitude and climbing rate of the UAV. The altitude can be directly controlled through changing the climbing rate. The movements in the horizontal plane are indirectly controlled via the attitude targets. Without the autopilot the UAV would lose upwards thrust when tilting in any direction since the total thrust would be split in the horizontal plane as well. However, the autopilot handles this since it is regulating the vertical velocity. The entire control system can therefore be divided into a vertical and a horizontal controller.

5.2 State Machine

The context of the control problem changes with the distance between the vehicles, which motivates the use of different control laws at different distances. A state machine has therefore been implemented where each state corresponds to a unique control law. The overall strategy of the state machine is described as follows:

- The UAV is flying at constant altitude and moves towards the UGV at the maximum allowed velocity. The altitude is high enough to account for inaccuracies in the height estimation in order to prevent collision with the ground. When the UAV is approaching the UGV it starts to slow down in order to prevent an overshoot.

- When the UAV is in close proximity of the UGV it starts to match the UGV's movements and its horizontal position. This requires a faster response time.

- When the UAV closely follows the movements of the UGV it starts descending while keeping the horizontal positions as close as possible.

The above explained actions are each captured in a unique state. The states are called: `APPROACH`, `FOLLOW` and `DESCEND`. The idea is that the state machine starts in the `APPROACH` state and transitions to the other states in the presented order as the distance between the vehicles gets shorter. A transition occurs if a condition is met where the governing transition quantity is the relative distance between the vehicles in the horizontal plane. A state transition can take place in both directions. To prevent the state machine from switching back and forth between two states, a deadband is introduced such that a greater distance is required in order to transition backwards. An overview of the state machine and its conditions can be seen in Figure 5.2. The following three subsections describes and motivates each state.

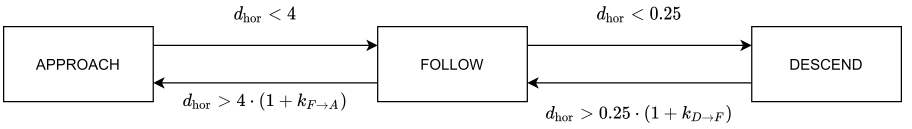


Figure 5.2: *The states and state transition conditions of state machine. The variable d_{hor} represents the relative distance in the horizontal plane between the vehicles, $k_{F \to A}$ and $k_{D \to F}$ represent the additional distance required to transition backwards. The distances are given in meters and $k_{F \to A} = 0.2$ and $k_{D \to F} = 1.0$.*

5.2.1 Approach

The APPROACH state is the initial state of the state machine. In this state the UAV flies at a constant altitude of approximately 10 meters[10] while moving towards the UGV in the horizontal plane. The main goal is to close the distance between the vehicles as fast and efficiently as possible. Since the distance to the target is longest in this state the estimation of the relative position difference, based on the UWB measurements, exhibits the largest errors. Also, the distance implies that the movements of the UGV does not affect the direction of the relative position vector as much compared to the other states. These two facts entail that the control strategy in this state should be to guide the UAV in a coarse direction towards the UGV, i.e. it should neglect fast movement variations while aiming for a future point of collision.

As stated in Section 3.8 guidance laws are control laws designed to guide a pursuer such that it reaches a target object. A common application for guidance laws is therefore in missile systems. However, Gautam et al. [38] investigates the possibility to use guidance laws for UAV landing. They compare the performance of three classical guidance laws: pure pursuit, line-of-sight and proportional navigation. Gautam et al. concluded that the proportional navigation (PN) law is the most efficient in terms of time and required acceleration. Borowczyk et al. [8] test the PN strategy in a real world scenario with positive results. This motivates the use of a PN-controller in the APPROACH state.

As described in Section 3.8 the PN-control law provides an acceleration which is perpendicular to the LOS-vector between the vehicles. Hence, the control law only steers the vehicle. The law assumes that the vehicle is propelled forward from another source such as a constant acceleration or an initial velocity, which is the case for a missile. Therefore, a complementary controller is required. However, compared to a missile the approaching UAV poses additional constraints on the relative velocity. The UAV should slow down before reaching the target in order to not overshoot. Hence, the complementary controller should allow the UAV to fly with the maximum allowed velocity, saturated attitude angles, until it is in the proximity of the UGV. At that point, it should start to slow down to prevent an overshoot. The implementation of a PN controller that accounts for

[10] Chosen as a safety precaution in consultation with the staff at FOI.

this is presented in Section 5.3.3.

5.2.2 Follow

The FOLLOW state is used when the UAV is closer than 4 meters from the UGV in the horizontal plane. The goal of the state is to position the UAV directly above the UGV, i.e. a zero relative difference in the horizontal plane. In order to preserve the matching horizontal position and to track the motions of the UGV it is important with a fast response time. In this state the desired altitude is lowered to 5 m.

In [39] it is concluded that a PN-controller generates oscillative control actions at close distances and therefore become inefficient, which is solved by switching to a PD-controller. Borowczyk et al. [8] build on this result in their approach. They also state that a PD-controller is easier to tune in order to achieve fast response time and accuracy.

This motivates the use of a PD-controller controller in the FOLLOW state. However, in this work the PD-controller is extended to a PID-controller to mitigate static errors due to wind or the UGV moving too fast.

5.2.3 Descend

The DESCEND state is the final state in the state-machine. The underlying controller is the same as in the FOLLOW state, i.e. a PID-controller. The difference is that the desired altitude is set to zero. The UAV will therefore start to descend further towards the UGV.

5.3 Implementation

This section describes how the control laws in the control system are implemented. The implementation is based on the theory described in sections 3.7–3.8. As mentioned in Section 5.1, the control signals allow the control system to be divided into a vertical and horizontal controller. This is also viable since the error signal $e = \|\mathbf{p}_\text{d} - \mathbf{p}_\text{rel}\|_2$ is minimised when each of its components are minimised, which can be done independently. The control system is therefore split into a vertical and horizontal controller. The control algorithms used in the control system require the relative position and velocity of the UAV with respect to the UGV. The state vector $\mathbf{x}$ is therefore

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_\text{rel} \\ \dot{\mathbf{p}}_\text{rel} \end{bmatrix} \in \mathbb{R}^6 , \tag{5.4}$$

where $\dot{\mathbf{p}}_\text{rel}$ denotes the time derivative of $\mathbf{p}_\text{rel}$. The overall structure of the control system is presented in Figure 5.3, which is an extension of the `Control System` block in Figure 5.1.

The vertical controller takes the vertical position and velocity $(z, \dot{z})$ as well as the desired height h_d as inputs and outputs a climbing rate c_r with the goal of reaching h_d. The implementation of the vertical controller is described in Section 5.3.1.

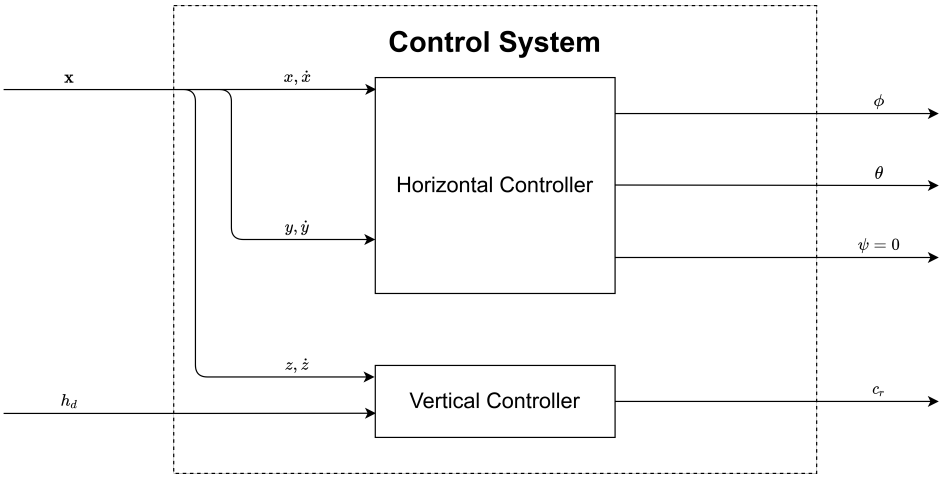


Figure 5.3: *Overall structure of the implementation of the UAV control system. Inputs are the state vector* $\mathbf{x}$ *and the desired height* h_d*. The control system outputs control commands in the form of attitude angles* (ϕ, θ, ψ) *and a climbing rate* c_r*.*

The horizontal controller block has the goal of steering the UAV towards the UGV to attain a relative horizontal distance of zero. The inputs of the controller are x, $\dot{x}$, y and $\dot{y}$. The controller then outputs the attitude target (ϕ, θ). As stated earlier ψ is set to 0 throughout the entire landing task.[11] The horizontal controller uses two different control structures: a proportional-integral-derivative (PID) structure (`FOLLOW` and `DESCEND`) and a proportional navigation (PN) structure (`APPROACH`). It is the task of the state machine to decide which of the two control structures to use.

The control signals that are passed to the autopilot software must be within certain boundaries. The attitude angles ϕ and θ must be within the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and the climbing rate $c_r \in [0, 1]$. However, further restrictions of the steering commands are used as safety precautions to make sure that the UAV cannot accelerate too aggressively during flight. The restrictions are

$$\phi \in [-\alpha_{\max}, \alpha_{\max}], \quad \theta \in [-\alpha_{\max}, \alpha_{\max}], \quad c_r \in [c_{r,\min}, 1], \tag{5.5}$$

where $\alpha_{\max} = 0.3$ rad and $c_{r,\min} = 0.3$. For convenience, the saturation function f_{sat} is defined as

$$f_{\text{sat}}(x,\ x_{min},\ x_{max}) := \begin{cases} x_{max} & \text{if } x \geq x_{max} \\ x & \text{if } x_{min} < x < x_{max} \\ x_{min} & \text{if } x \leq x_{min} \end{cases}. \tag{5.6}$$

[11] The choice of $\psi = 0$ is for simplicity's sake and is not a restriction of the system. To fly with an arbitrary yaw angle ψ the attitude vector $\begin{bmatrix}\phi & \theta\end{bmatrix}^T$ is simply rotated ψ radian before being sent to ArduCopter.

5.3.1 Vertical Controller

This subsection aims to present how the vertical controller is implemented. As stated earlier, the goal of the vertical controller is to output a climbing rate $c_r \in [c_{r,\min}, 1]$ that minimises the error term $(h_d - z)^2$, i.e. make sure that the UAV flies h_d meters above the UGV. The climbing rate c_r is controlled via feedback using a PD-control structure, see Section 3.7. The required inputs are therefore z and $\dot{z}$. The vertical control problem is the same throughout the entire landing task. Hence, the same vertical controller can be used in all of the state machine's states. The vertical controller is summarised in Figure 5.4.

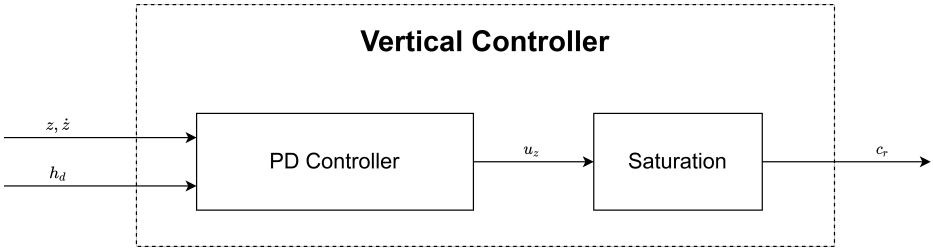


Figure 5.4: *Overall structure of the implementation of the vertical controller. Inputs are $(z, \dot{z})$ and the desired height h_d. The vertical controller outputs a saturated climbing rate c_r.*

The error signal e_z used by the PD-controller is chosen as $e_z = h_d - z$. With only a proportional part in the controller this choice of error signal would output a positive c_r if the UAV is below the desired altitude, i.e. ascend. And vice versa if the UAV is above the desired altitude. Thus, the choice of error signal as $e_x = h_d - z$ minimises $(h_d - z)^2$. The output from the PD controller, u_z, is saturated, returning the outgoing climbing rate:

$$c_r = f_{\text{sat}}(u_z,\ c_{r,\min},\ 1). \tag{5.7}$$

5.3.2 Horizontal PID Controller

This subsection aims to describe how the horizontal PID controller is implemented. The control goal of driving the relative horizontal distance to zero is equivalent to minimising the relative distance in the x- and y-axis independently. Hence, two decoupled PID-controllers can be used, each steering the UAV in the x- and y-direction respectively. An overview of the control structure is presented in Figure 5.5.

The controller has the goal of minimising x^2 and y^2 through changes in the attitude angles (ϕ, θ). To avoid repetition when explaining the PID-controllers, consider the case of the controller in the x-direction. According to Section 3.3, there is a relation between attitude angles and acceleration. Therefore, if the PID-controller generates an acceleration, it can subsequently be converted to an attitude angle. Suppose the error signal e_x for the PID-controller in the x-direction is

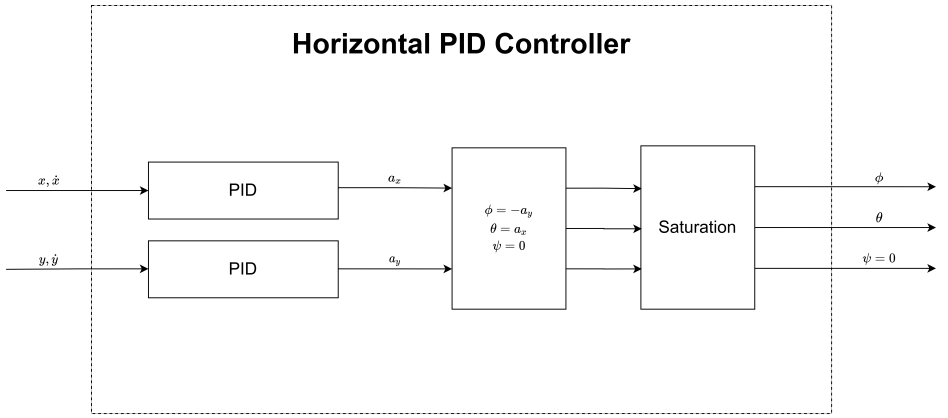


Figure 5.5: *Overall structure of the implementation of the horizontal PID controller. Inputs are the horizontal position and velocity and the output is the attitude angles (ϕ, θ, ψ).*

chosen as $e_x = -x$. With only a proportional term in the controller this error signal would result in a desired negative acceleration when the relative x-distance is positive, i.e. the UAV accelerates towards the UGV. Vice versa when the relative x-distance is negative. Thus, the choice of error signal as $e_x = -x$ minimises x^2. The PID controllers return the horizontal accelerations (a_x, a_y).

Since the PID controller is implemented in discrete time, the integral in (3.24) must be approximated. A common approximation is derived using the Euler method, and gives

$$\int_0^t e_x(\tau)d\tau = T_s \sum_{j=0}^{k} e_{x,j}\,, \tag{5.8}$$

where k is the sample at time t and T_s is the period time of the controller. However, when this approximation is coupled with control signal saturation it can lead to a integrator windup [28]. To avoid this phenomenon, the error sum is saturated and multiplied with a decay factor $d_\mathrm{f} < 1$. The saturation prevents the I-part from being too large and the decay factor makes sure that older errors do not have as much impact as newer ones. The error sum at sample k, $S_{x,k}$, is calculated as

$$S_{x,k} = d_\mathrm{f} \cdot f_\mathrm{sat}(S_{x,k-1} + e_{x,k},\ -S_{max},\ S_{max})\,, \tag{5.9}$$

where S_{max} is the saturation limit and $S_{x,0} = 0$. The I-part at sample k, $I_{x,k}$, is calculated as

$$I_{x,k} = K_i \cdot T_s \cdot S_{x,k}\,, \tag{5.10}$$

where K_i is the integral gain.

When converting the horizontal acceleration signals (a_x, a_y) to the attitude angles (ϕ, θ), the model of UAV flight described in Section 3.3 is simplified. Firstly,

the yaw angle ψ is set to zero. Inserting $\psi = 0$ in (3.8) gives

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \begin{bmatrix} \cos\phi \sin\theta \\ -\sin\phi \\ \cos\phi\cos\theta \end{bmatrix} \frac{T}{m}. \tag{5.11}$$

The model in (5.11) can be simplified further.

As described in Section 5.3, the maximum allowed attitude angle in the control system is ± 0.3 rad ($\approx \pm 17°$). The attitude angles of the UAV will therefore be relatively small, and the small-angle approximation ($\sin v \approx v$, $\cos v \approx 1$, $\tan v \approx v$) is applicable. Applying the small-angle approximation to (5.11) results in

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \theta \cdot T' \\ -\phi \cdot T' \\ -g + T' \end{bmatrix}, \tag{5.12}$$

where $T' = T/m$. Equation (5.12) describes that a change in the pitch angle θ, leads to a proportional change of acceleration in the x-direction and vice versa for $-\phi$ and y. Therefore, the control signals are mapped to attitude angles as follows:

$$\begin{aligned} \phi &= -a_y \\ \theta &= a_x \end{aligned} \tag{5.13}$$

However, the outputted attitude angles must be saturated to $[-\alpha_{\max}, \alpha_{\max}]$, resulting in

$$\begin{aligned} \phi &= f_{\text{sat}}(-a_y, -\alpha_{\max}, \alpha_{\max}) \\ \theta &= f_{\text{sat}}(a_x, -\alpha_{\max}, \alpha_{\max}). \end{aligned} \tag{5.14}$$

5.3.3 Proportional navigation Control

As stated in section 5.2.1 the PN control law only provides an acceleration vector perpendicular to the LOS-vector between the vehicles. To address this, Borowczyk et al. complement their PN controller with a PD controller that outputs an acceleration parallel with the LOS-vector [8]. The same control structure is used in this work but with a modification in the PD controller. The perpendicular acceleration $\mathbf{a}_\perp \in \mathbb{R}^2$ and a parallel acceleration $\mathbf{a}_\parallel \in \mathbb{R}^2$ are combined to an acceleration $\mathbf{a}_{\text{xy}}$ before converted to the attitude angles (ϕ,θ), which are the final outputs of the horizontal PN controller. The overall structure of the horizontal PN controller is presented in Figure 5.6.

The PD-structure that computes $\mathbf{a}_\parallel$ is based on the theory described in Section 3.7. Let $\mathbf{p}_{\text{xy}} = \mathbf{x}_{1:2}$ and $\mathbf{v}_{\text{xy}} = \mathbf{x}_{4:5}$, i.e. the relative position and velocity between the UAV and the UGV in the horizontal plane. Then let $\mathbf{e}_p = -\mathbf{p}_{\text{xy}}$ be the position error. Since $\mathbf{v}_{\text{xy}}$ is not always in the same direction as the LOS vector, $\mathbf{v}_{\text{xy}}$ is projected onto the LOS vector as follows

$$\mathbf{v}_{\parallel\text{LOS}} = \frac{\mathbf{p}_{\text{xy}} \cdot \mathbf{v}_{\text{xy}}}{\mathbf{p}_{\text{xy}} \cdot \mathbf{p}_{\text{xy}}} \mathbf{p}_{\text{xy}}. \tag{5.15}$$

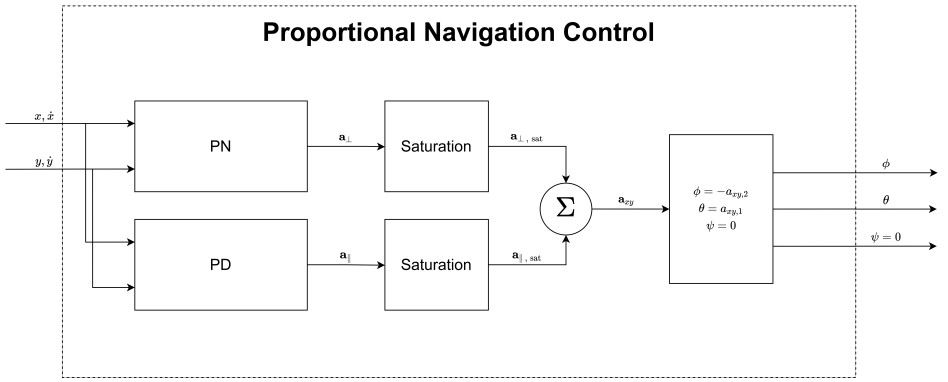


Figure 5.6: *Overall structure of the implementation of the horizontal PN-controller. Inputs are the horizontal position and velocity and the output is the attitude angles (ϕ, θ, ψ).*

The velocity error $\mathbf{e}_v$ is then defined as

$$\mathbf{e}_v = k_{\text{approach}} \cdot \frac{\mathbf{v}_{\parallel \text{LOS}}}{\left\|\mathbf{v}_{\parallel \text{LOS}}\right\|_2} - \mathbf{v}_{\parallel \text{LOS}}\,, \tag{5.16}$$

where k_{approach} is positive parameter that describes a desired approaching velocity. The parallel acceleration $\mathbf{a}_\parallel$ is finally computed as

$$\mathbf{a}_\parallel = K_p \mathbf{e}_p + K_d \mathbf{e}_v\,, \tag{5.17}$$

where K_p and K_d are positive tuneable parameters. The parameters are chosen such that the proportional term dominates when the UAV is far away from the UGV, to make sure that the UAV approaches the UGV with the maximum allowed velocity. When the UAV approaches the UGV the proportional error will shrink, and the relative velocity will start to dominate and make sure that the UAV slows down to prevent an overshoot.

In order to compute $\mathbf{a}_\perp$, (3.25) is used as follows

$$\mathbf{a}'_\perp = \lambda |\mathbf{v}| \frac{\mathbf{p}}{|\mathbf{p}|} \times \boldsymbol{\Omega}\,, \quad \boldsymbol{\Omega} = \frac{\mathbf{p} \times \mathbf{v}}{\mathbf{p} \cdot \mathbf{p}}\,, \tag{5.18}$$

where $\mathbf{p} = \begin{bmatrix}\mathbf{x}_{1:2}^T & 0\end{bmatrix}^T$ and $\mathbf{v} = \begin{bmatrix}\mathbf{x}_{4:5}^T & 0\end{bmatrix}^T$. The third component is set to zero since an acceleration in the horizontal plane is desired. The sign of the equation has changed since $\mathbf{x}_{1:2}$ = {Pursuer position} – {Target position} and $\mathbf{x}_{4:5}$ = {Pursuer velocity} – {Target velocity}, which is opposite from the theory. The requested parallel acceleration $\mathbf{a}_\perp$ is the first two components of $\mathbf{a}'_\perp$.

From experiments it has been noted that the length of $\mathbf{a}_\parallel$ is much larger than $\mathbf{a}_\perp$. Changing the value of the K_p-parameter does not solve this since $\mathbf{a}_\parallel$ is proportional to the relative distance, which varies a lot during the `APPROACH` phase. This becomes an issue when the accelerations are converted to attitude angles

according to (5.14). The saturation performed in (5.14) takes place in the x- and y-direction. If $\mathbf{a}_{\parallel}$ and $\mathbf{a}_{\perp}$ are not aligned with those axes, the much larger $\mathbf{a}_{\parallel}$-vector would dominate and surpass the saturation limit, such that $\mathbf{a}_{\perp}$ have no effect. To solve this, the saturation is performed on the individual accelerations before they are combined. The saturation is implemented as follows

$$\mathbf{a}_{\parallel,\,\text{sat}} = f_{\text{sat}}(|\mathbf{a}_{\parallel}|,\ -\alpha_{\max},\ \alpha_{\max}) \cdot \frac{\mathbf{a}_{\parallel}}{|\mathbf{a}_{\parallel}|} \tag{5.19a}$$

$$\mathbf{a}_{\perp,\,\text{sat}} = f_{\text{sat}}(|\mathbf{a}_{\perp}|,\ -\alpha_{\max},\ \alpha_{\max}) \cdot \frac{\mathbf{a}_{\perp}}{|\mathbf{a}_{\perp}|} \tag{5.19b}$$

Thereafter, the accelerations are combined in the following manner

$$\mathbf{a}_{xy} = \alpha_{\max} \cdot \frac{\mathbf{a}_{\perp,\,\text{sat}} + \mathbf{a}_{\parallel,\,\text{sat}}}{\max(\|\mathbf{a}_{\perp,\,\text{sat}} + \mathbf{a}_{\parallel,\,\text{sat}}\|_{\infty},\ \alpha_{\max})}\,, \tag{5.20}$$

which guarantees that each element of $\mathbf{a}_{xy}$ is within the interval $[-\alpha_{\max},\ \alpha_{\max}]$. Finally, the attitude angles can be computed using (5.13) as follows

$$\phi = -\mathbf{a}_{xy,2}\,, \quad \theta = \mathbf{a}_{xy,1}\,. \tag{5.21}$$

5.4 Control System Evaluations

The control system's performance is evaluated in the simulation environment described in Section 2.4. During this test the control system use true position and velocity data when computing the control commands. A simulated test is performed as follows:

- The UAV is spawned at the position $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$. The UGV is randomly placed on a circle with radius 50 m and the UGV's heading is randomly selected.
- When the flight controller is initialised, which takes about 10 s, the UAV starts to ascend and the EKF is initialised. Simultaneously, the UGV starts moving in the horizontal plane with a velocity of 4 m/s according to the pattern described in Appendix C.3.
- When the UAV reaches an altitude of 9 m it starts using the outputs of the control system in order to approach and land on top of the UGV.

During the entire simulation the UAV is under the influence of an external wind force, see Appendix C.1.

The simulation is repeated 100 times. Figure 5.7 shows the UGV's movements during the 100 runs. The shape of the trajectories is determined by the UGV's starting position, its heading at the start and how the UGV is manoeuvring during the simulation. Additionally, the length of the trajectory is determined by the time it takes for the UAV to land, which is influenced by the UGV's movements and the direction and amplitude of the wind. Figure 5.8 illustrates the UAV's flight path together with the movements of the UGV from the previous figure.

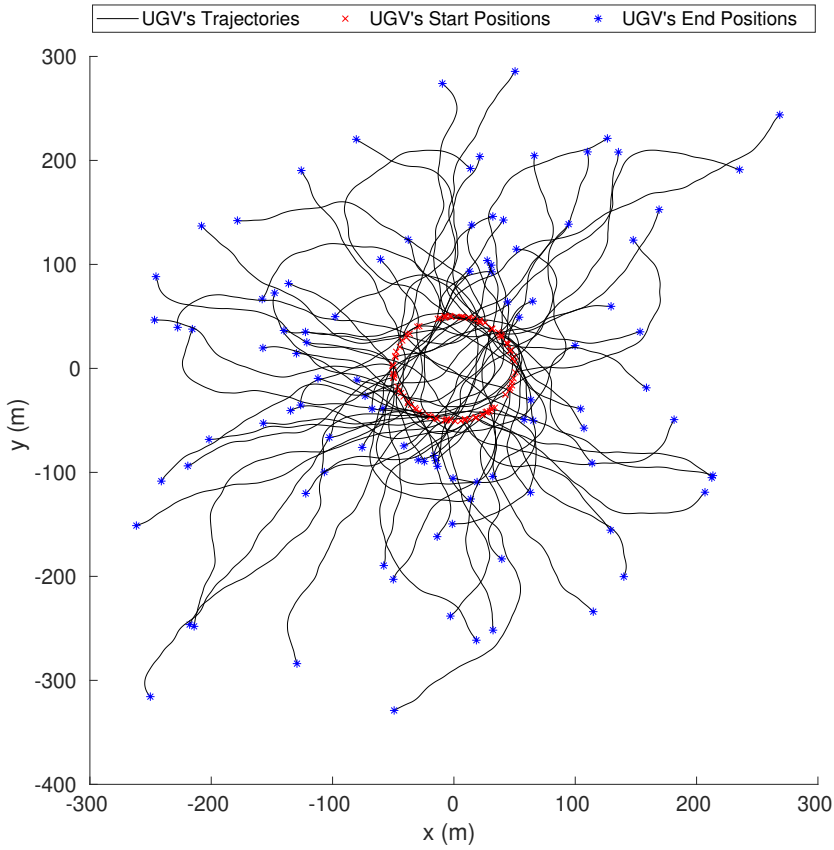


Figure 5.7: *Movement trajectories for UGV during 100 simulation runs. The red crosses indicates the starting positions of the UGV, the black line shows the trajectories and the blue stars represent the position of the UGV when the UAV has landed on top of it.*

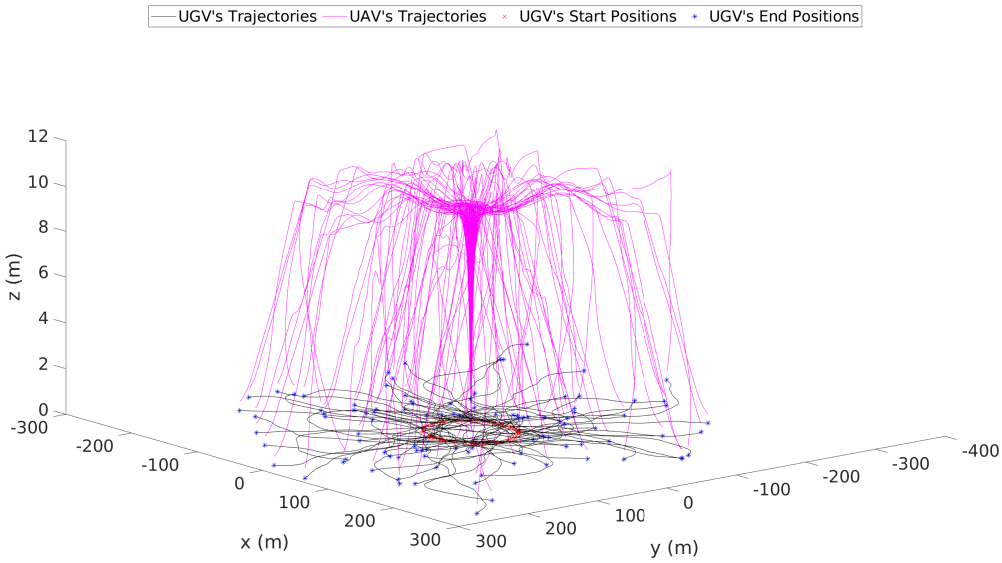


Figure 5.8: *The UGV's and the UAV's movement paths during the 100 runs. The UAV's flight paths are represented with the purple lines.*

The UAV lands on top of the UGV in each simulation. The landing positions of the UAV on the UGV platform can be seen in Figure 5.9 together with a 95 % confidence interval. Additionally, Figure 5.9 presents 100 landing positions when the K_i constant is set to zero for both horizontal PID-controllers, i.e. no I-part during landing phase. Without an I-part the landing positions are shifted backwards, which makes sense since the UGV is moving in the opposite direction. Additionally, the landing positions are more spread out when no I-part is being used. In this idealised case, the landing accuracy is high and 95 % of the landings occur less than 20 cm from the centre of the landing pad.

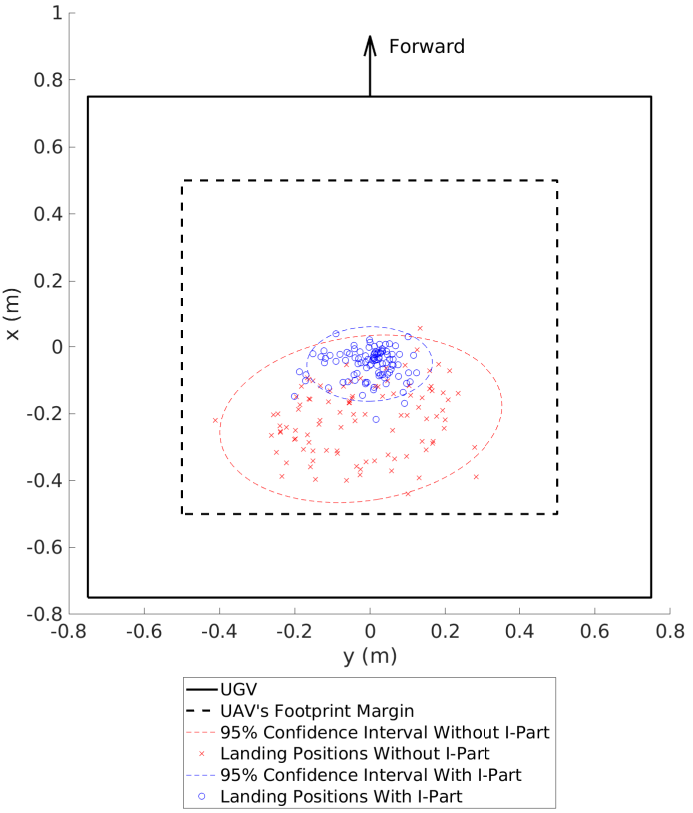


Figure 5.9: *UAV landing positions on the UGV, with and without an I-part in the horizontal PID-controller. The black line indicates the border of the landing platform on the UGV, the dashed black line illustrates the region where the UAV can land safely in order to have all four legs on the platform and the black arrow shows the direction the UGV is moving in. The blue circles show the landing positions when an I-part is used, and the red crosses show the landing position when no I-part is used. Two 95 % confidence ellipses are presented alongside the landing positions.*

The time distribution for the simulations are studied, in particular the elapsed time for all of the states, the APPROACH state as well as the FOLLOW and DESCEND state. The median, min and max time of each scenario is presented in Table 5.1, which is based on the distributions presented in Appendix D.1. The variations of elapsed time during an entire mission seem to be caused by variations in the APPROACH phase. Further investigation of these variations conclude that the main cause is the environmental circumstances of a simulation, i.e. the path the UGV traverses as well as the direction of the wind.

The fluctuations in elapsed time in the FOLLOW and DESCEND states is significantly smaller when compared to the APPROACH state. The extremes of these states were studied further, and it was confirmed that the additional time was caused by so-called *retakes*. If the UAV is moving too far away from the center of the UGV during the DESCEND state, it results in a transition back to the FOLLOW state, or a retake. Retakes were observed in 12 of the 100 simulations. The retakes are not caused by estimation errors since the control system uses true position and velocity data. Factors causing a retake is instead sudden change of the UGV's movements or shift in amplitude of the wind.

Table 5.1: *Time distribution from 100 simulations. Median, min and max of the elapsed time in all of the states, the APPROACH state as well as the FOLLOW and DESCEND state.*

States	Median (s)	Min (s)	Max (s)
All	35.0	22.2	87.9
APPROACH	15.9	2.7	68.8
FOLLOW and DESCEND	19.0	17.4	31.0

In order to further illustrate how the control system operates, data from one of the 100 simulations is presented. Figure 5.10 shows the UAV's and the UGV's horizontal movements and Figure 5.11 shows the vehicles' respective vertical positions during one simulation. In Figure 5.10 it can be seen that the UAV is drifting in the wind direction while it is ascending. When the UAV reaches an altitude of 9 m it starts using the outputs from the control system and therefore starts moving towards the UGV. When the UAV transitions from the APPROACH state to the FOLLOW state the desired altitude changes from 10 m to 5 m, which Figure 5.11 shows. The UAV transitions to the DESCEND state before it reaches an altitude of 5 m and will therefore continue to descend until it has landed on the UGV.

To get an understanding of how well the UAV is able to track the UGV's velocity in the horizontal plane the velocities in the x-direction are presented in Figure 5.12. It can be seen that the vehicles' velocities are different during the APPROACH state, but as the UAV transition to the FOLLOW and DESCEND states the UAV tries to match the UGV's velocity. The UAV's velocity exhibits a slightly oscillative behaviour during the FOLLOW and DESCEND states. An explanation for this is that the current implementation of the PID-controller only reacts on the current states and does not predict future state values.

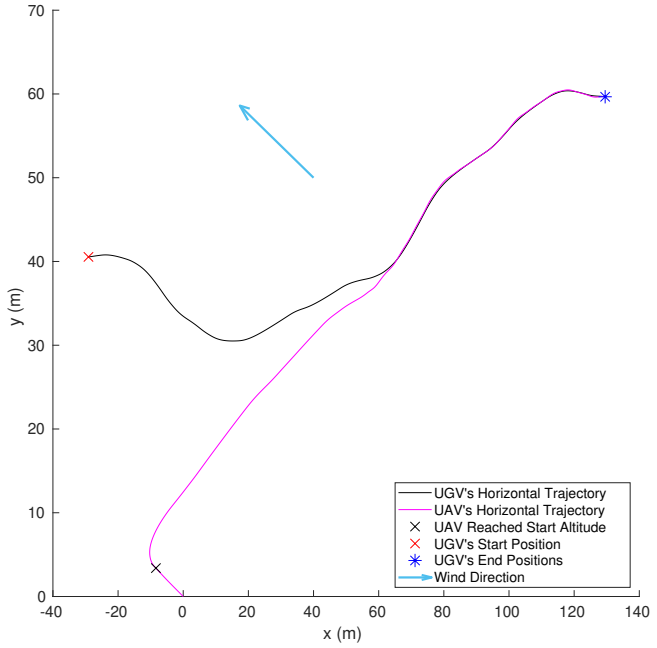


Figure 5.10: *The UAV's and the UGV's horizontal movements during one simulation. The blue arrow shows the direction of the wind and the black cross the position where the UAV reaches an altitude of 9 m.*

Figure 5.13 shows the attitude angles, roll and pitch, from the control system during the simulation. When the control system transitions from the `APPROACH` to the `FOLLOW` state, a steep change in control signals is observed. These changes are caused by the switch between controllers in the state transition. This behaviour has been reduced in the control system tuning but is evidently still present. Furthermore, the control signals show oscillative behaviour in the `DESCEND` state, see Figure 5.13. The cause of the oscillations is that the UAV has the goal of matching its position and velocity with the UGV's. With the UGV varying direction regularly as well as changes in the wind amplitude, the control signal has difficulty of getting to a steady state.

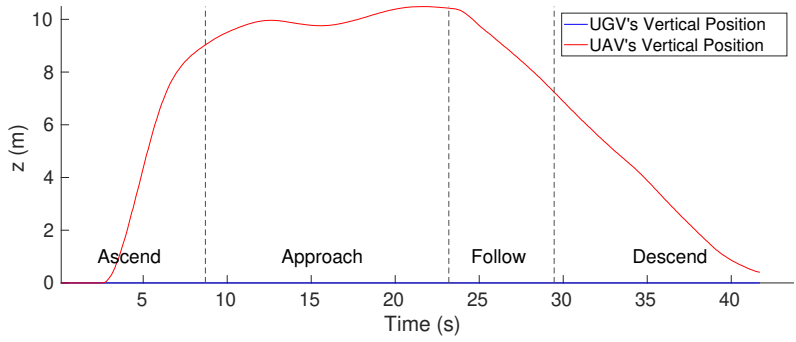


Figure 5.11: *The UAV's and the UGV's vertical position during one simulation. The leftmost dotted line indicates when the UAV reaches an altitude of 9 m. The other two dotted lines indicate transition in the state machine.*

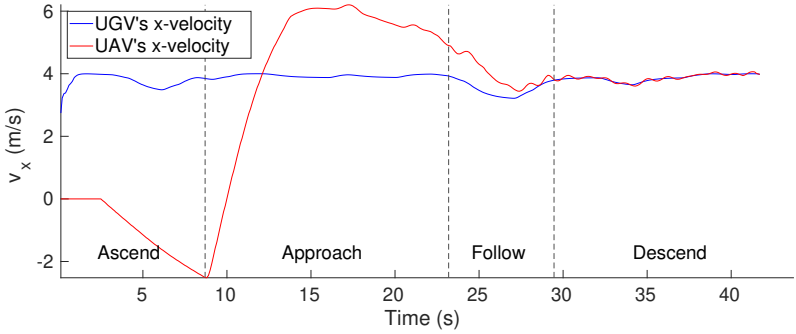


Figure 5.12: *The UAV and UGV velocity in the x-direction during one simulation.*

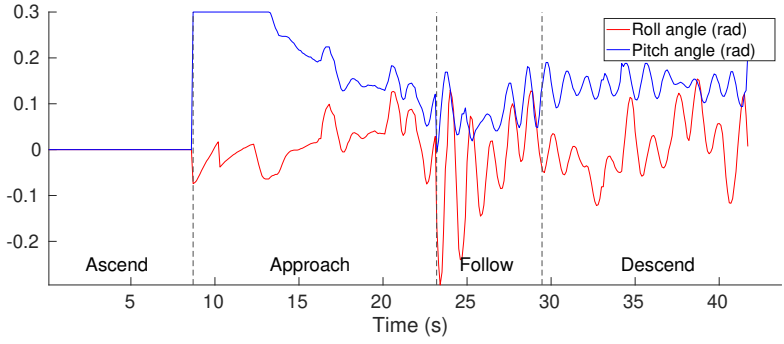


Figure 5.13: *Attitude output (roll and pitch angles) from the control system during one simulation.*

6

Landing System Evaluation

This chapter presents the results from the evaluation of the landing system. In Section 6.1 the results from simulated landings using both the estimation and the control system are presented. An experimental validation of the estimation system is presented in Section 6.2.

6.1 Landing System Simulation

As presented in Section 5.4, the control system was capable of executing 100 consecutive landings when using true state data. The next step is to combine the control system and the estimation system and evaluate their combined performances, i.e. let the control system compute control commands using estimated position and velocity data. The analysis is carried out using the same simulation environment with identical preliminaries as in Section 5.4, with the exception that the control system uses estimated states.

Figure 6.1 illustrates the trajectories of both the UAV and UGV. The flight paths in Figure 6.1 do not show any new behaviour compared to the paths in Figure 5.8, the UAV is able to land on the UAV during every simulation in this case as well. The landing positions of the UAV can be seen together with a 95 % confidence interval in Figure 6.2. Compared to the landing positions in Figure 5.9, the landing positions are somewhat more spread out, but still within the UAV footprint margin. The UAV lands within approximately 3 dm from the centre of the landing pad in all simulations.

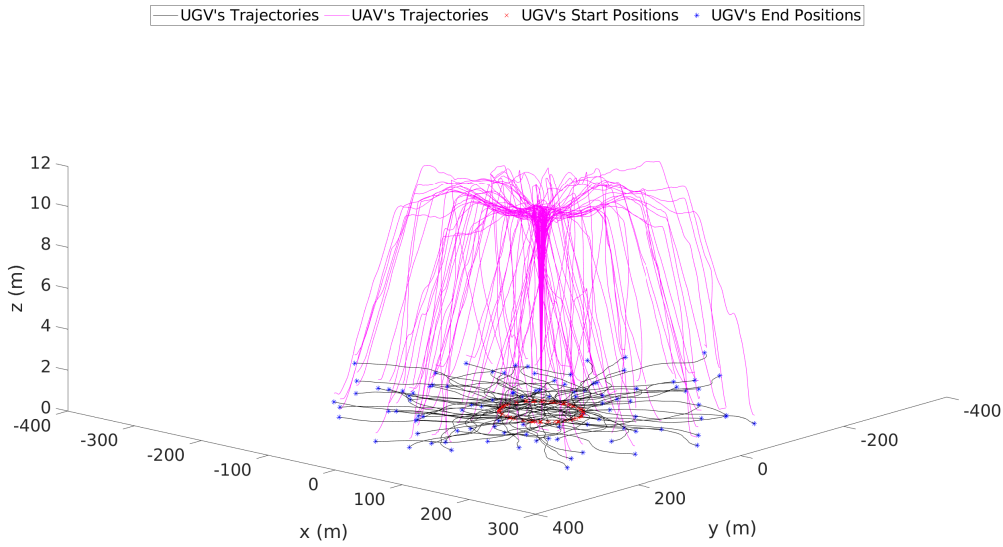


Figure 6.1: *UAV and UGV trajectories over 100 simulations.*

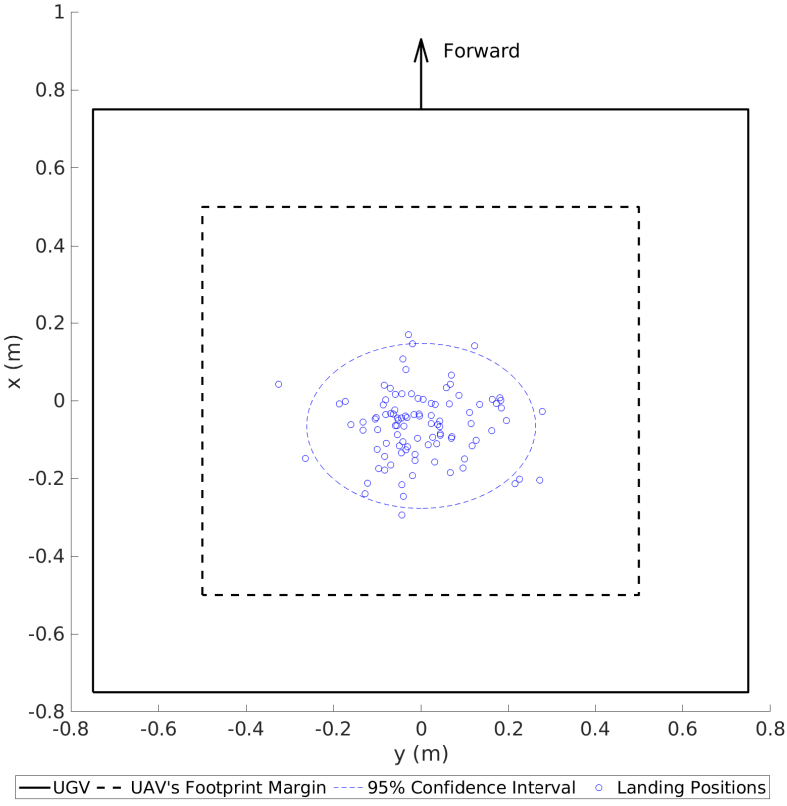


Figure 6.2: *Landing positions of the UAV with respect to the {UGV}-frame. The positions of 100 runs as well as 95 % confidence interval of the landing positions are shown. The black dashed line represents the footprint margin of the UAV, i.e. the area in which the UAV can land safely.*

The time distribution of the 100 runs can be seen in Table 6.1. The table is generated from the data presented in Appendix D.1. Compared to Table 5.1 it can be seen that the median time is slightly larger when using estimated data, which is expected. The median time in the APPROACH state is similar between the runs. However, the max time when using true data is about 10 s larger compared to when using estimated data. This indicates that the extreme cases in the APPROACH state is mostly determined by the environmental circumstances rather than the estimation quality. On the contrary, the extreme cases in the FOLLOW and the DESCEND states suggest that the governing factor is the estimation quality. The median time is not substantially increased; however, the max time is approximately 50 % larger using estimated data. The UAV exhibits 21 retakes when using estimated data compared to the 12 retakes with true data. The additional retakes together with the larger max time indicate a more uncertain behaviour during the FOLLOW and DESCEND when estimated data is being used.

Table 6.1: *Time distribution from 100 simulations. Median, min and max of the elapsed time in all of the states, the APPROACH state as well as the FOLLOW and DESCEND state.*

States	Median (s)	Min (s)	Max (s)
All	36.8	22.5	95.1
APPROACH	16.4	0.7	54.6
FOLLOW and DESCEND	19.9	17.9	46.4

The performance of the estimation system is measured using the root mean square error (RMSE) of the estimated states. The estimated states are $\{x, y, z, v_x, v_y, v_z\}$. Since the direction of the x- and y-axis are determined by the underlying reference system, which is decoupled from the vehicles orientation, a horizontal distance and velocity is used instead in the evaluations. The horizontal distance r is defined as $r = \sqrt{x^2 + y^2}$ and the horizontal velocity v_r is defined as $v_r = \sqrt{v_x^2 + v_y^2}$.

During a simulation, the relative distance between the vehicles varies. In the APPROACH state the relative horizontal distance ranges from 4 to 100 m while in the FOLLOW state and the DESCEND state the horizontal distance is below 4 m. The relative distance affects the estimation quality and the RMSE is therefore calculated separately for the APPROACH state and landing phase.

Table 6.2 and Table 6.3 presents the mean, min and max RMSE for the 100 runs for position and velocity estimates respectively. Additionally, the tables present data from simulations where the camera is excluded. The distributions of the RMSE in simulation is presented further in Appendix D.2.

First, it can be seen that without the camera, the RMSE for the horizontal and vertical position estimate is slightly lower in the APPROACH state but higher in FOLLOW and DESCEND states compared to when the camera is used. In the APPROACH state the UAV flies at the maximum allowed altitude, 10 m, and in the FOLLOW and DESCEND states the UAV immediately starts to descend to an alti-

tude of 5 m and 0 m respectively. From Section 4.5.5 it is stated that the bias in the camera system increases with the altitude and that the bias is not accounted for in the estimate. This suggests that the bias in the camera system's estimate will have most impact in the APPROACH state and decrease in the FOLLOW and DESCEND states, which the results reflect. Furthermore, the camera system only provides position estimates in the very end of the APPROACH phase when the vehicles are close to each other. The velocity estimate is not affected by bias in the position estimate, which also can be seen in Table 6.3.

Secondly, the horizontal position and velocity have a much larger mean RMSE compared to the vertical quantities in the APPROACH state. The explanation is that the system mainly uses measurements from the UWB sensor network when estimating the horizontal quantities, while the reference barometer is used for the vertical estimates. From Section 4.4.3 it is concluded that the UWB sensor network exhibits a larger position error at longer distances. Comparing the results from the APPROACH state with the results from the FOLLOW- and the DESCEND states it can be seen that the estimation errors are significantly lower for the latter. The accuracy of the UWB system improves during the FOLLOW- and DESCEND-phase due to a more favourable geometry. Furthermore, the camera-based positioning system can also provide measurements in these phases.

Table 6.2: *Mean, min and max RMSE for horizontal and vertical position during 100 simulations. The RMSE is divided into the APPROACH state and the FOLLOW and DESCEND states. Additionally, RMSE without the camera system is presented.*

State	Camera	Horizontal Position			Vertical Position		
		Mean	Min	Max	Mean	Min	Max
APPROACH	Yes	4.54	2.8	7.29	0.11	0.06	0.24
	No	4.46	2.56	6.69	0.12	0.07	0.27
FOLLOW and DESCEND	Yes	0.11	0.06	0.19	0.05	0.03	0.1
	No	0.18	0.08	0.4	0.06	0.03	0.19

Table 6.3: *Mean, min and max RMSE for horizontal and vertical velocity during 100 simulations. The RMSE is divided into the APPROACH state and the FOLLOW and DESCEND states. Additionally, RMSE without the camera system is presented.*

State	Camera	Horizontal Velocity			Vertical Velocity		
		Mean	Min	Max	Mean	Min	Max
APPROACH	Yes	0.88	0.38	1.56	0.12	0.06	0.25
	No	0.92	0.44	2.66	0.13	0.06	0.25
FOLLOW and DESCEND	Yes	0.17	0.08	0.54	0.08	0.05	0.23
	No	0.20	0.1	0.86	0.1	0.05	0.46

To illustrate the impact of the camera system during the landing phase, the landing positions of the UAV without a camera are presented in Figure 6.3. The

landing positions are more spread out compared to Figure 6.2. Also, a few of the landing positions are outside the safety margins. This does not necessarily mean that one of the UAV's legs is outside the platform, but there is a risk. Additionally, the UAV performs retakes in 65 of the 100 simulations. This indicates that the landing system is less robust during landing when omitting measurements from the camera system.

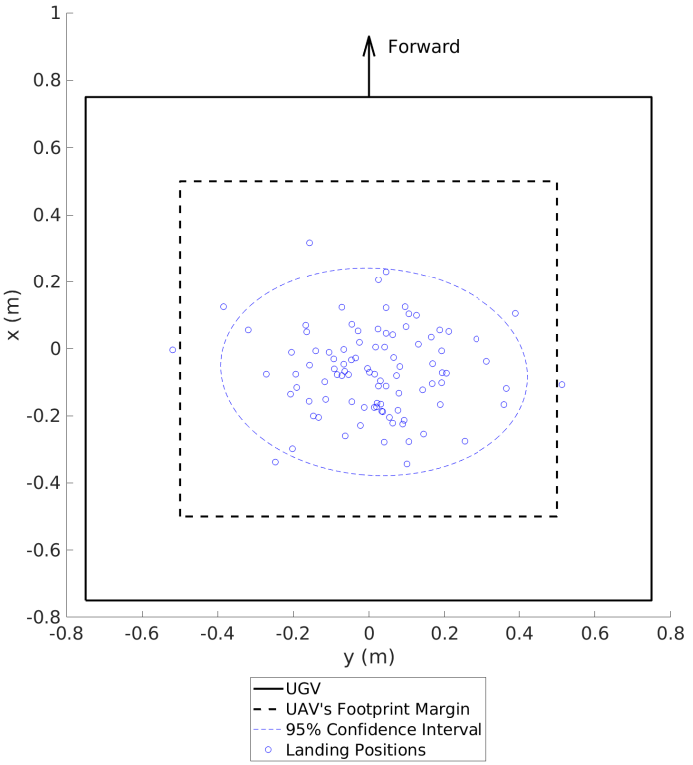


Figure 6.3: *Landing positions of the UAV with respect to the {UGV}-frame without the camera system. The positions of 100 runs as well as 95 % confidence interval of the landing position are shown. The black dashed line represents the footprint margin of the UAV, i.e. the area in which the UAV can land safely.*

6.2 Estimation System Validation

Since the landing system has proven to work in a realistic simulation environment, the next step is to validate it experimentally. To make sure that every subsystem is working as intended, the validation is performed incrementally. A part of the validation process, which was executed in this thesis, is to validate the estimation system with physical sensors. The results from the test are presented in this section.

The validation test was performed in the Visionen laboratory environment at Linköping University. Visionen is equipped with a camera positioning system that can provide a position and velocity estimate with millimetre precision using Qualisys camera tracking system[12]. The positioning environment is in an arena with physical dimensions similar to the volume in which the UAV executes its landing phase. These experiments are therefore used to validate the accuracy of the estimation system during the landing phase, with the estimates from the Qualisys camera system used as ground truth.

The UGV was stationary during the test and lacked a communication link with the UAV. Hence, the attitude, accelerometer or barometer data from the IMU on the UGV could not be utilised. The attitude of the UGV was instead taken from the Qualisys camera system at the start of the simulation, and the UGV's accelerations were set to zero. Due to the time variance of the air pressure readings from the barometer, caused by naturally varying air pressure and sensor bias drift, the height estimate from the single barometer deviated more than expected. Therefore, the height measurement was instead simulated using height data from Qualisys. In accordance with the reference barometer error model described in Section 4.7.2, AWGN with variance 0.2 m^2 was added to the height measurements. Additionally, a bias of 0.2 was added which is a conservative mean of the deviation in Table 4.4.

The test setup was as follows:

- The UGV was placed in the arena. The positions of the UGV mounted UWB-anchors were noted.
- The Jetson on the UAV was powered on and started recording sensor data from the camera, IMU, UWB-tag and Pixhawk. The ground truth pose and velocity for each vehicle was recorded.
- The UAV's engines were powered on.

After the setup, the UAV took off and the pilot started flying the UAV around in the room, while the Jetson continued to record data. Figure 6.4 presents the UAV and UGV during the test.

The flight test lasted for 300 s in total and during that time, a number of manual landing manoeuvres were performed to provide data similar to the `FOLLOW` and `DESCEND` states in simulation but without any actual UAV touch-downs. Three of these landing manoeuvres were extracted and further analysed.

[12] For more information on Qualisys, see: https://www.qualisys.com/software/qualisys-track-manager/

Figure 6.4: *The UAV and UGV during the Visionen test.*

The recorded data from the manoeuvres, as well as the artificial sensor measurements from the UGV, are post-processed in the estimation system. The post-processing is performed using two sensor setups: all sensors used in the estimation system and all sensors except the camera system used in the estimation system. The second sensor setup is motivated by the goal of finding out if a landing can be achieved covertly and independent of the time of day. Such a landing could not rely on the camera system in its current state since it requires visibility of the tag which is not possible at night. Solutions such as lighting up the landing pad would not meet the covert landing criteria. From both setups, the positions and velocities in each manoeuvre are estimated. The UAV trajectory (estimated as well as ground truth) from the second flight manoeuvre is presented in Figure 6.5 to illustrate the performance of the estimation.

To further investigate the estimation accuracy for both sensor setups, the RMSE, maximum absolute error and a 95th percentile of the absolute error for the position and velocity estimate are determined, see Table 6.4 and Table 6.5. The results show that the RMSE for the horizontal position estimate is almost doubled when the camera system is excluded. The results are also compared with the re-

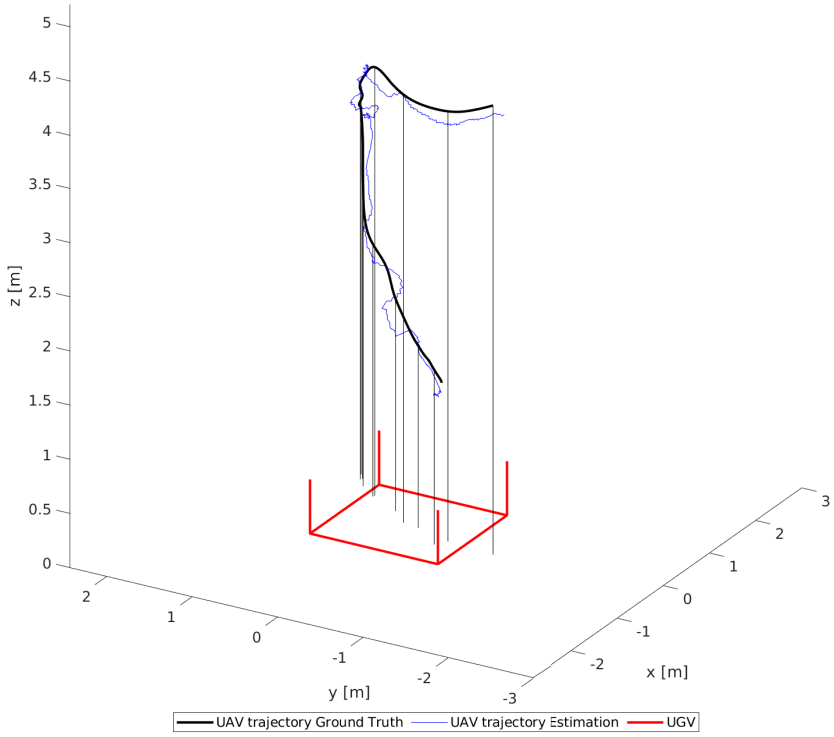


Figure 6.5: *UAV trajectory (ground truth and estimation) from the second flight manoeuvre in the Visionen test. The landing pad of the UGV with four beams pointing upwards is shown as well.*

sults in Table 6.2 and Table 6.3 for the `FOLLOW` and `DESCEND` states. The individual RMSE from the experimental tests are less than or equal to the corresponding maximum RMSE in simulation. This suggests that there exist simulations where the estimation quality is similar to the experimental tests.

The maximum absolute error in Table 6.4 and 6.5 is around 2 to 3 times larger than the RMSE. The greatest discrepancy between RMSE and the maximum is seen in landing manoeuvre 1 and 3. In these manoeuvres, the large error was caused by the camera losing sight of the tag. In the case of flight manoeuvre 2, the camera has sight of the tag during the entirety of the manoeuvre, see Figure 6.6. This explains the significantly lower maximum error when the camera is used. Comparing the estimation error without the camera system, the maximum absolute error is similar during all three manoeuvres. The maximum error without the camera system is also similar to the maximum error during flight 1 and 3, which confirms that the large error is caused by the camera system losing track of the tag.

The estimation without the camera system was investigated further. It was concluded that the large errors can occur during the entire manoeuvre. Hence, with only a UWB sensor network and an IMU for estimation, a relatively high horizontal estimation error can appear at close distances to the UGV (~1-2 m).

Table 6.4: *RMSE, maximum absolute error as well as the 95th percentile of the absolute error (in m) for horizontal and vertical position for three different landing manoeuvres during the Visionen test. Results are shown with and without the camera system.*

		Horizontal Position			Vertical Position		
Landing Manoeuvre	Camera	RMSE	Max	95th	RMSE	Max	95th
1	Yes	0.19	0.54	0.48	0.10	0.19	0.15
2	Yes	0.09	0.19	0.16	0.08	0.18	0.14
3	Yes	0.18	0.65	0.41	0.07	0.29	0.13
1	No	0.21	0.54	0.45	0.10	0.23	0.18
2	No	0.39	0.65	0.61	0.12	0.21	0.18
3	No	0.29	0.56	0.47	0.07	0.17	0.14

Table 6.5: *RMSE, maximum absolute error as well as the 95th percentile of the absolute error (in m/s) for horizontal and vertical velocity for three different landing manoeuvres during the Visionen test. Results are shown with and without the camera system.*

		Horizontal Velocity			Vertical Velocity		
Landing Manoeuvre	Camera	RMSE	Max	95th	RMSE	Max	95th
1	Yes	0.16	0.41	0.35	0.17	0.4	0.34
2	Yes	0.14	0.54	0.32	0.11	0.38	0.20
3	Yes	0.25	0.73	0.65	0.10	0.27	0.20
1	No	0.19	0.53	0.39	0.18	0.39	0.34
2	No	0.16	0.41	0.32	0.12	0.34	0.23
3	No	0.17	0.44	0.31	0.10	0.39	0.19

The sensor measurements are also analysed to get further knowledge about

the estimation system. When studying the accelerometer data from the test, unexpected behaviour was found. From the time of engine start, the raw data was very noisy, which continued throughout the test. When the UAV landed and the engines were cut-off, an oscillation transient could be seen in the data as well. It was suspected that the oscillations are caused by resonance in the sensor platform, caused by the vibrations from the engines. To reduce resonance in future tests, the mounting of the platform could be revised. Another unwanted behaviour from the accelerometer is that the data contained outliers. With the current implementation, these outliers could not be detected by the NIS-test as described in Section 4.2.

To illustrate the performance of the camera system, its estimated position during landing manoeuvre 2 is studied, see Figure 6.6. The camera system mostly uses the small tag for the estimation, since the UAV is too close to get the entire large tag in the FOV of its camera. Camera data is acquired during the entire landing manoeuvre.

To study the performance of the UWB sensor network, the ground truth distance was calculated for each anchor-tag pair based on the anchor position as well as the position of the UAV and UGV from Qualisys. Figure 6.7 presents the performance of anchor-tag pairs 1 and 4 during the complete flight: pair 1 follows the ground truth nicely, while pair 4 exhibit some serious outliers. This motivates the use of an outlier rejection algorithm. It has been confirmed that the NIS-test, see Section 4.2, manage to reject this kind of outliers. To study the performance of all anchor-tag pairs further, the RMSE, maximum absolute error as well as the 95th percentile of the absolute error is calculated, see Table 6.6. The measurements from anchor-tag pair 3 had outliers of similar size as with anchor-tag pair 4. Despite these outliers, 95 % of the measurements from all anchor-tag pairs have errors below 30 cm. These errors are both caused by the performance of the UWB radios as well as inaccuracy when measuring the sensor placement.

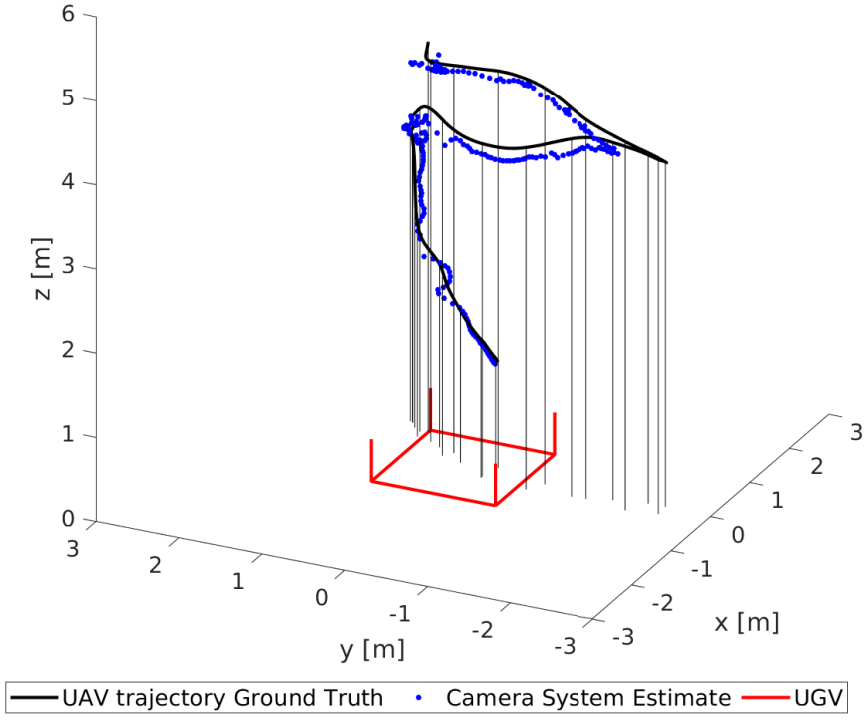


Figure 6.6: *Position estimation from the camera system in the second flight manoeuvre from the Visionen test. The landing pad of the UGV with four beams pointing upwards is shown as well.*

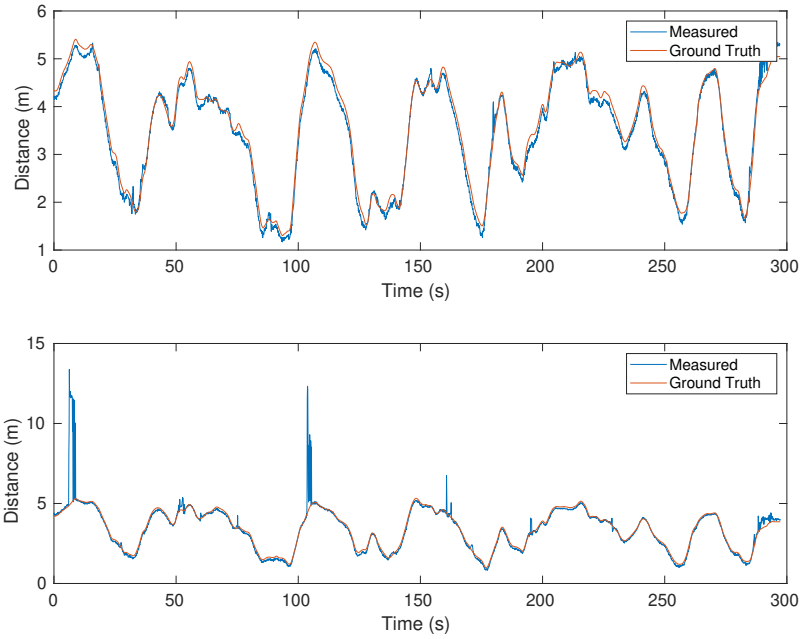


Figure 6.7: *UWB sensor performance for anchor-tag pairs 1 and 4 (top and bottom). Measured distance is compared to ground truth.*

Table 6.6: *RMSE, maximum absolute error as well as the 95th percentile of the absolute error (in m) of the range measurements from all anchor-tag pairs.*

Anchor-tag pair	RMSE	Max	95th
1	0.15	0.79	0.24
2	0.13	0.62	0.26
3	0.24	7.26	0.29
4	0.63	8.48	0.23

7

Conclusion and Future Work

This chapter presents the conclusions of this thesis, with regards to the problem formulation presented in the introduction, together with suggested future work.

7.1 Conclusions

A combined estimation and control system capable of landing a UAV on a mobile UGV has been presented. The combined system has been tested and evaluated in a realistic simulation environment, developed with the aid of Gazebo, ArduCopter SITL and ROS. The individual sensors used in the system have been evaluated separately. Finally, the estimation system has been validated through experiments.

Initially, the performance of the UWB radios were evaluated with physical sensors. Based on the evaluation, three different placements of the UWB anchors were analysed with CRLB. The anchor placements were decided based on the results. The accuracy of the camera system was investigated with a simulated camera, to get an understanding of the bias and variance of its position estimate. From these evaluations, as well as small-scale experiments indicating the accuracy of the accelerometers and barometers, the estimation system was designed. The implementation of the sensors in the simulation environment were based on these results.

The control system proposed in this work utilises two control structures: a PN controller is used to approach the moving UGV and a PID controller is used to land on top of it. This structure is based on [8], however, the implementation of the PN controller is modified and a PID controller is used instead of a PD controller. The control system has been evaluated in simulation and has shown positive results. The system managed to safely land the UAV during all the 100 simulations. The feasibility of the control system cannot be confirmed before a test is conducted with physical hardware. However, since a SITL implementation

of the flight controller has been used during simulation, the control system will most likely perform similarly in such an experiment. Despite the successful landings, it was noted that the control system showed a slight oscillative behaviour during the landing phase, which results in a less robust system. One probable factor to this behaviour is that the control system does not account for future movements of the UGV.

The estimation system has been evaluated during simulated landings as well. Combined with the control system, landings were achieved in 100 of 100 attempts, which confirms a fully functional landing system. However, when compared to the landing results using true data, the landing positions on the UGV are more spread out and the UAV is more prone to restart its attempts. A UAV landing at night is also of interest, therefore landings have been simulated without the use of the camera system as well. The results show a much more uncertain behaviour during the landing phase, behaviours that would be unwelcome in a physical landing.

To validate the results from simulation, the estimation was evaluated experimentally using the hardware platforms. The experiment was conducted in an environment similar to the landing phase, except with a stationary UGV. The estimation accuracy during the experiment was comparable to the accuracy in simulation. This indicates the feasibility of the estimation system with real hardware during the landing phase. The estimation system was also experimentally evaluated without the use of the camera system. Results show that the accuracy is significantly lower when the camera system is not used, the estimation error is doubled to tripled in size. When considering the behaviour in simulation presented previously as well, we conclude that a physical landing without a camera is improbable.

7.2 Future Work

The long term intention of the landing system is to deploy it on real hardware. Further experiments are required to get a better understanding of the validity of the simulation results. First, the estimation system should be tested at longer distances resembling the APPROACH-phase. Secondly, the control system should be implemented and evaluated on the real UAV platform.

The performance of the control system during the landing phase could potentially by improved utilising knowledge about the movements of the UGV. Implementing such a feature would most likely reduce the oscillate behaviour during the landing phase, which in turn would allow for a more robust system. The movements of the UGV could either be predicted or provided, assuming a communication link between the UAV and the UGV. Baca et al. propose a movement prediction strategy in [9], which has shown to work in a variety of real-world experiments.

A future goal is to have an estimation system functioning independent of time of day. Thus, the current version of the estimation system needs to be modified since an ordinary camera is used. Results show that the system performs signif-

icantly worse without a camera and a substitute is most likely needed. Further research should therefore investigate if an IR-camera can be used instead.

Appendix

A

Hardware Drivers

This appendix describes the hardware drivers used for communication between the sensors and the Jetson Nano.

A.1 XSens Driver

The official Xsens ROS-package *xsens_mti_driver*[13] is used to provide IMU data to the ROS network.

A.2 Camera Driver

The ROS-package *gscam*[14] is used on the jetson processor to handle the CSI communication with the camera. The package also provides an interface to the ROS-network and publishes the raw camera data to it with a rate of 10 Hz. The parameters of the camera model is published alongside. The ROS-package *image_proc*[15] is used to rectify the raw image data using the camera model.

A.3 Decawave data extraction

The DWM1001 Decawave UWB modules used in the landing system has a pre-built firmware library [40]. The firmware library supports a shell mode which accepts a list of commands. Via this shell mode it is possible to extract range data from Decawave modules acting as tags in a sensor network. The commands can

[13] Package at http://wiki.ros.org/xsens_mti_driver.
[14] Package at http://wiki.ros.org/gscam.
[15] Package at http://wiki.ros.org/image_proc

be sent to a Decawave module through serial communication. In this case UART is used with a baud rate of 115200. There are a number of shell-commands but the one of interest in this case is the `lec`-command. This command enables a mode which starts sending range data over UART with a CSV format. The data is provided with a frequency of 10 Hz and is given in the format:

[DIST, # Anchors, ID Anchor 1, X, Y, Z, Range, ID Anchor 2, ...]

B

Simulated Sensors

This appendix describes how the sensors were simulated.

B.1 Attitude

The standard IMU-sensor in Gazebo is used to obtain attitude data. The data is noise-free. In order to get a more realistic simulation, noise is added to the attitude data before it is used. The noise is applied as an additional rotational matrix, which is called $\mathcal{R}_{noise}$. The $\mathcal{R}_{noise}$ matrix is a rotational matrix generated from randomly selected Euler angles. The Euler angles: roll, pitch and yaw are drawn from a Gaussian distribution with zero mean and 1° variance. On top of the randomly drawn angles a bias is added. The bias is randomly chosen as $[0.7°, -0.5°, 0.6°]$[16], where each element correspond to an Euler angle. The attitude data is used to compute a rotation matrix ${}^{\mathrm{W}}\mathcal{R}_{\mathrm{b}}$, which then is contaminated with noise as follows

$$ {}^{\mathrm{W}}\mathcal{R}_{\mathrm{b,noise}} = \mathcal{R}_{noise}\,{}^{\mathrm{W}}\mathcal{R}_{\mathrm{b}}\,, \tag{B.1} $$

where ${}^{\mathrm{W}}\mathcal{R}_{\mathrm{b,noise}}$ is the rotation matrix used in simulation.

B.2 Ultra-Wide Band Sensor Network

The UWB sensor network is simulated in Gazebo and used during the simulated landing task. The network consists of four anchors placed on the UGV according to the configuration chosen in Section 4.4.3. One tag is placed at the centre of

[16] Based on the RMS values of a XSens MTi-600 IMU, see: https://www.xsens.com/hubfs/Downloads/Leaflets/MTi%20600-series%20Datasheet.pdf

the UAV as well. For each anchor-tag pair, the measured distance is calculated as presented in (4.15), with the noise variance determined in Section 4.4.2.

However, as described in Table 4.1 in Section 4.4.2, the parameters a and b can vary between sensors. Therefore, to get a more realistic measurement a and b are modelled as random variables with a uniform distribution $\mathcal{U}$. The scaling factor a is bounded to the interval $[1.0028, 1.0036]$ and the bias b is bounded to the interval $[0.01, 0.1]$, i.e.

$$a \sim \mathcal{U}(1.0028, 1.0036), \quad b \sim \mathcal{U}(0.01, 0.1) \tag{B.2}$$

The parameters a and b are generated for each anchor-tag pair at the start of a simulation.

B.3 Camera

Gazebo is capable of simulating a camera sensor[17]. The sensor is attached to a model in Gazebo and outputs a simulated camera stream based on the pose of the model. The camera parameters calculated in Section 4.5.1 are used with the simulated camera to allow a more realistic simulation. Pixelwise AWGN is also added to the simulated image.

B.4 Accelerometer

Gazebo has an implementation of an IMU-sensor[18]. The sensor provides the linear acceleration of the body it is attached to and the data is represented in a body-fixed coordinate system. The simulated sensor supports modelling of AWGN.

B.5 Reference Barometer

The barometers are simulated in Gazebo and used during the simulated landing task. The pressure generated by the simulated sensors comes from the inverse of equation (3.26), which is presented here

$$P = P_0 \cdot \left[1 + \frac{L(z - z_0)}{T}\right]. \tag{B.3}$$

The lower altitude z_0 is set to zero. P_0 and T are set to 101.325 kPa and 288 K respectively, which are the ISA standard values for pressure and temperature at sea level, see Section 3.9. The altitude z is the true altitude of the model the sensor is attached to in Gazebo. In order to get a more realistic pressure measurement, AWGN is added to the generated pressure data. The variance corresponds to the variance of the real barometer sensors, which is chosen as 14 based on the result Section 4.7.2.

[17] A description of the Gazebo camera sensor can be found at: http://sdformat.org/spec?ver=1.7&elem=sensor

[18] See http://gazebosim.org/tutorials?tut=ros_gzplugins.

C

Gazebo Plugins

This appendix describes the Gazebo plugin developed for the simulation environment.

C.1 Wind Force Plugin

A plugin was created to apply a wind to the UAV in the simulation. The wind was modelled as a force vector. The wind force is applied in the horizontal plane and the magnitude of the force vector is characterised as a telegraph process. The two states of the telegraph process are 0.5 N and 1.0 N. The mean time before the processes switches to the other state is 6 s. When the plugin is initialised a random wind direction is chosen. During each iteration of the simulation noise is added to the direction. An illustration of the wind force can be seen in Figure C.1.

C.2 Air Resistance Plugin

Gazebo do not apply air resistance to the UAV model during flight, therefore a plugin was created to simulate it. The plugin applies a force in the opposite direction of the velocity of the UAV. The force is computed equally in the x-, y- and z-direction. In the case of the x-direction, the force is calculated as follows

$$F_{\text{wind,x}} = -\text{sign}(v_x) \cdot k_x \cdot v_x^2 \,, \tag{C.1}$$

where v_x is the velocity in the x-direction and k_x a positive constant.

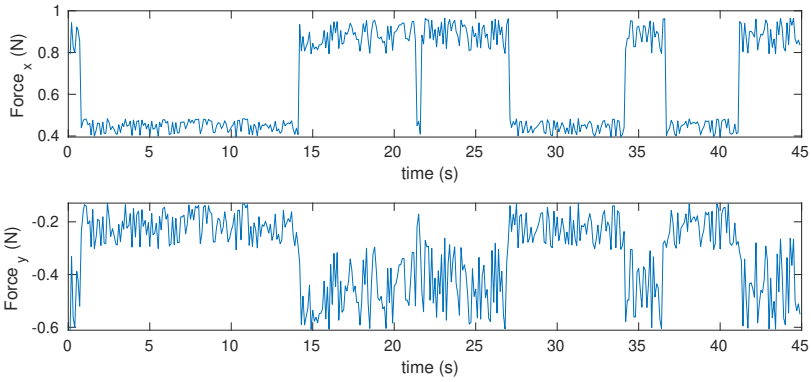


Figure C.1: *Example of the horizontal wind force during a simulation run.*

C.3 Random Trajectory Plugin

A plugin was created to enable the UGV to drive in a random pattern during simulations. When the simulation is initialised the UGV starts moving forward at a velocity of 4 m/s. Following this, the direction of the UGV either stays constant, turns +0.2 rad or turns −0.2 rad every 4 s. Each of the direction changes have equal probability. An outcome of what the resulting trajectory of the UGV could look like can be seen in Figure C.2.

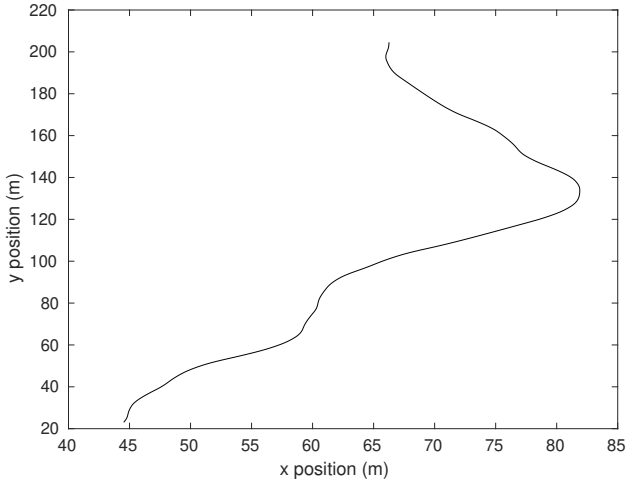


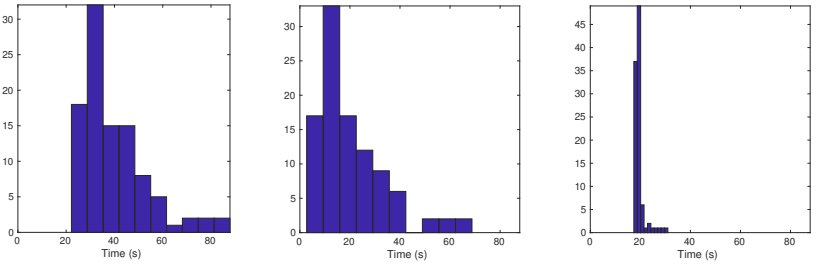
Figure C.2: *Example of the UGV's trajectory when using the Random Trajectory plugin.*

D

Additional Results

This appendix describes additional results from Section 5.4 and 6.1.

D.1 Time Distributions

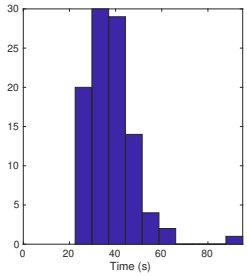


(a) *All states.*
Mean: 39.5 s.
Median: 35.0 s.

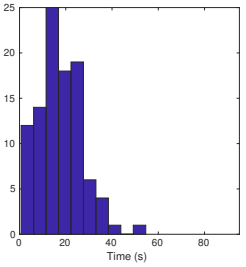
(b) `APPROACH`*.*
Mean: 20.0 s.
Median: 15.9 s.

(c) `FOLLOW` *and* `DESCEND`*.*
Mean: 19.5 s.
Median: 19.0 s.

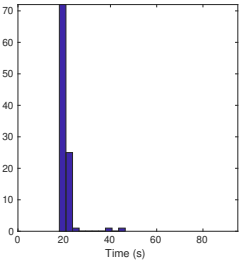
Figure D.1: *Time distributions for the 100 simulation runs using true states. The first subfigure* ***(a)*** *shows the total time distribution of the landing manuever which is during the* `APPROACH`*-,* `FOLLOW`*- and* `DESCEND`*-state. The second subfigure* ***(b)*** *shows the distribution for the time spent in the* `APPROACH`*-state. The third* ***(c)*** *subfigure show the time distribution for the landing phase (*`FOLLOW`*- and* `DESCEND`*-state).*



(a) *All states.*
Mean: 37.9 s.
Median: 36.8 s.



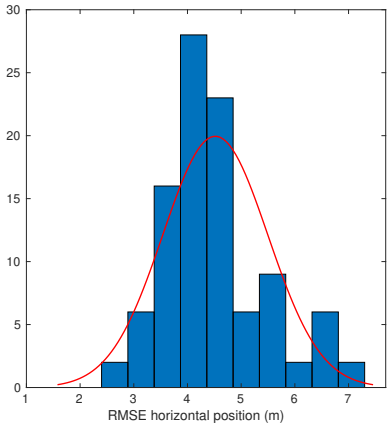
(b) *`Approach`.*
Mean: 17.4 s.
Median: 16.4 s.



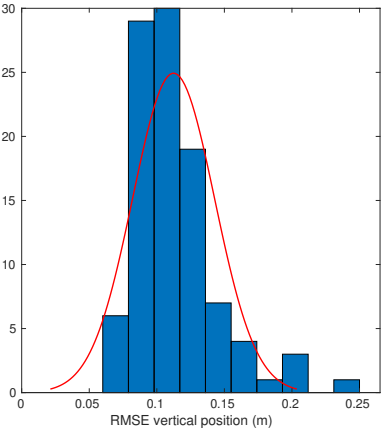
(c) *`Follow` and `Descend`.*
Mean: 20.5 s.
Median: 19.9 s.

Figure D.2: *Time distributions for the 100 simulation runs using estimated states states. The first subfigure **(a)** shows the total time distribution of the landing manuever which is during the `APPROACH`-, `FOLLOW`- and `DESCEND`-state. The second subfigure **(b)** shows the distribution for the time spent in the `APPROACH`-state. The third **(c)** subfigure show the time distribution for the landing phase (`FOLLOW`- and `DESCEND`-state).*

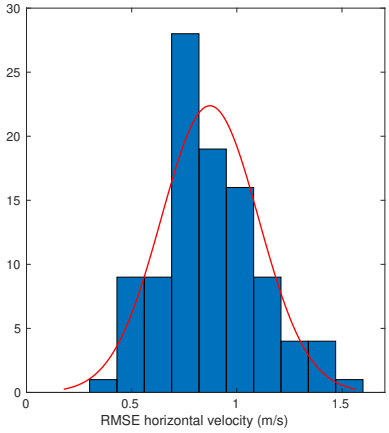
D.2 Root Mean Square Error



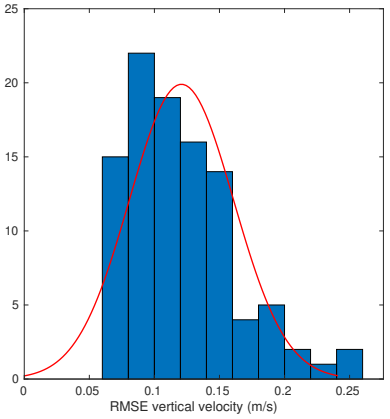
(a) RMSE horizontal position.
Mean: 4.54 m.
Standard Deviation: 0.98 m.



(b) RMSE vertical position.
Mean: 0.112 m. Standard Deviation: 0.030 m.

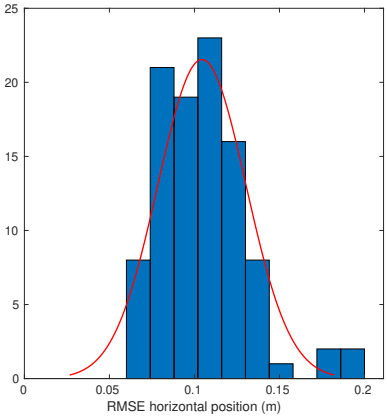


(c) RMSE horizontal velocity.
Mean: 0.877 m/s.
Standard Deviation: 0.223 m/s.

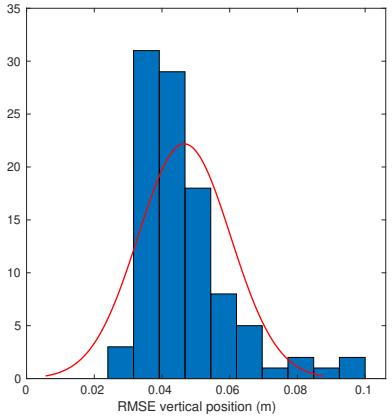


(d) RMSE vertical velocity.
Mean: 0.121 m/s.
Standard Deviation: 0.040 m/s.

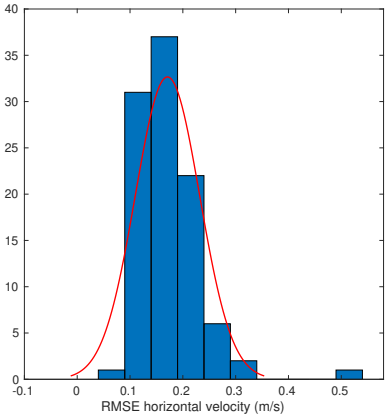
Figure D.3: *Distribution of root mean square error (RMSE) for both horizontal and vertical position and velocity, during the* `Approach`*-state over 100 runs using the camera system.*



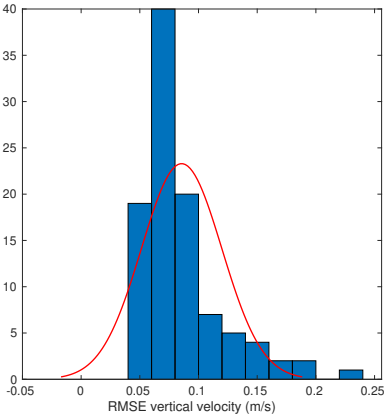
(a) *RMSE horizontal position.*
Mean: 0.105 m.
Standard Deviation: 0.027 m.



(b) *RMSE vertical position.*
Mean: 0.046 m.
Standard Deviation: 0.013 m.

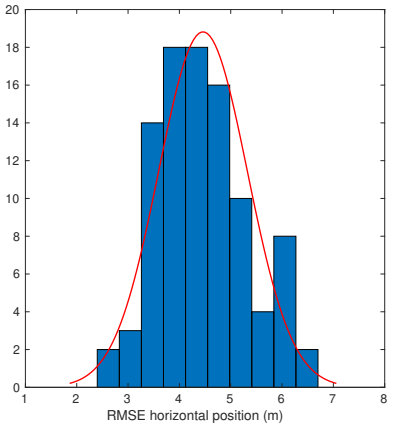


(c) *RMSE horizontal velocity.*
Mean: 0.170 m/s.
Standard Deviation: 0.061 m/s.

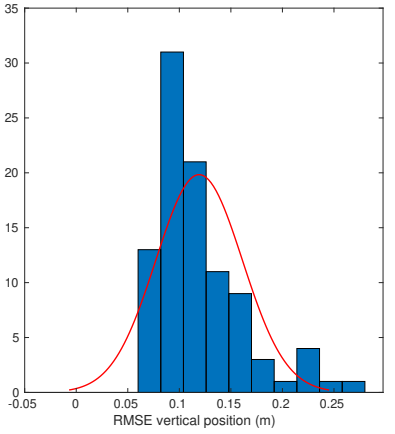


(d) *RMSE vertical velocity.*
Mean: 0.085 m/s.
Standard Deviation: 0.034 m/s.

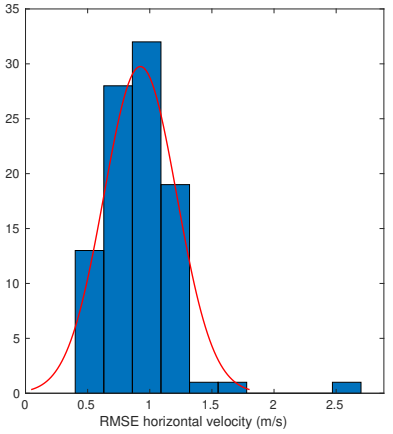
Figure D.4: *Distribution of root mean square error (RMSE) for both horizontal and vertical position and velocity, during the Landing phase over 100 runs using the camera system.*



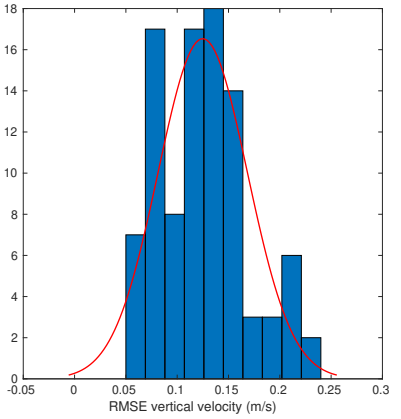
(a) *RMSE horizontal position.*
Mean: 4.46 m.
Standard Deviation: 0.87 m.



(b) *RMSE vertical position.*
Mean: 0.112 m.
Standard Deviation: 0.042 m.

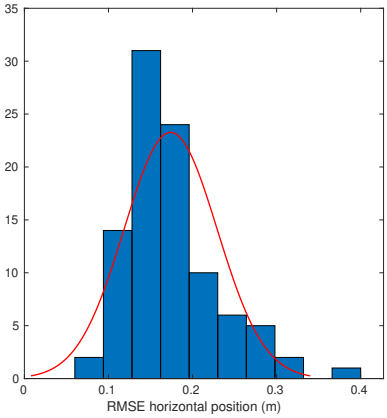


(c) *RMSE horizontal velocity.*
Mean: 0.924 m/s.
Standard Deviation: 0.293 m/s.

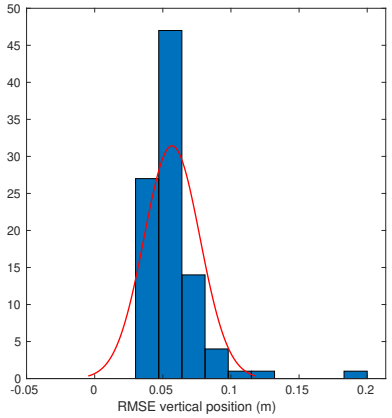


(d) *RMSE vertical velocity.*
Mean: 0.125 m/s.
Standard Deviation: 0.044 m/s.

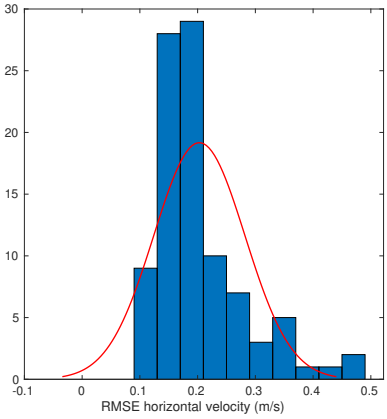
Figure D.5: *Distribution of root mean square error (RMSE) for both horizontal and vertical position and velocity, during the* `Approach`*-state over 100 runs without the camera system.*



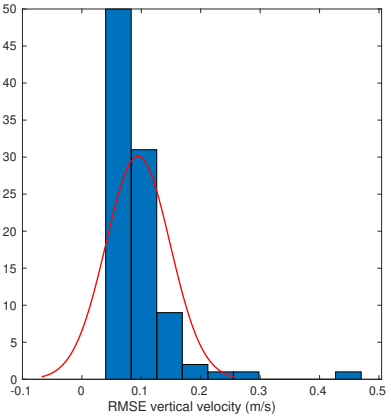
(a) *RMSE horizontal position.*
Mean: 0.174 m.
Standard Deviation: 0.055 m.



(b) *RMSE vertical position.*
Mean: 0.057 m.
Standard Deviation: 0.020 m.



(c) *RMSE horizontal velocity.*
Mean: 0.203 m/s.
Standard Deviation: 0.079 m/s.



(d) *RMSE vertical velocity.*
Mean: 0.094 m/s.
Standard Deviation: 0.054 m/s.

Figure D.6: *Distribution of root mean square error (RMSE) for both horizontal and vertical position and velocity, during the Landing phase over 100 runs without the camera system.*

Bibliography

[1] Nikolas Giakoumidis, Jin U. Bak, Javier V. Gómez, Arber Llenga, and Nikolaos Mavridis. Pilot-scale development of a UAV-UGV hybrid with air-based UGV path planning. *Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012*, (December):204–208, 2012. doi: 10.1109/FIT.2012.43.

[2] Jouni Rantakokko, Erik Axell, Niklas Stenberg, Jonas Nygårds, and Joakim Rydell. Tekniker för navigering i urbana och störda GNSS-miljöer. Technical Report FOI-R–4907–SE, Ledningssystem, Totalförsvarets Forskningsinstitut (FOI), December 2019.

[3] Andrew J. Kerns, Daniel P. Shepard, Jahshan A. Bhatti, and Todd E. Humphreys. Unmanned Aircraft Capture and Control Via GPS Spoofing. Journal of Field Robotics. *Journal of Field Robotics*, 31(4):617–636, 2014. doi: 10.1109/ICIF.2008.4632328.

[4] Sinan Gezici, Zhi Tian, Georgios B. Giannakis, Hisashi Kobayashi, Andreas F. Molisch, H. Vincent Poor, and Zafer Sahinoglu. Localization via ultra-wideband radios: A look at positioning aspects of future sensor networks. *IEEE Signal Processing Magazine*, 22(4):70–84, 2005. ISSN 10535888. doi: 10.1109/MSP.2005.1458289.

[5] Fabrizio Lazzari, Alice Buffi, Paolo Nepa, and Sandro Lazzari. Numerical investigation of an UWB localization technique for unmanned aerial vehicles in outdoor scenarios. *IEEE Sensors Journal*, 17(9):2896–2903, 2017.

[6] Linnea Persson. Cooperative Control for Landing a Fixed-Wing Unmanned Aerial Vehicle on a Ground Vehicle. Master's thesis, KTH Royal Institute of Technology, 2016.

[7] Linnea Persson. *Autonomous and Cooperative Landings Using Model Predictive Control*. Licentiate thesis, KTH Royal Institute of Technology, Stockholm, 2019.

[8] Alexandre Borowczyk, Duc Tien Nguyen, André Phu Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous land-

ing of a quadcopter on a high-speed ground vehicle. *Journal of Guidance, Control, and Dynamics*, 40(9):2373–2380, 2017. ISSN 07315090. doi: 10.2514/1.G002703.

[9] Tomas Baca, Petr Stepan, Vojtech Spurny, Daniel Hert, Robert Penicka, Martin Saska, Justin Thomas, Giuseppe Loianno, and Vijay Kumar. Autonomous landing on a moving vehicle with an unmanned aerial vehicle. *Journal of Field Robotics*, 36(5):874–891, 2019. ISSN 15564967. doi: 10.1002/rob.21858.

[10] Sven Lange, Niko Sünderhauf, and Peter Protzel. A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments. *2009 International Conference on Advanced Robotics, ICAR 2009*, 2009.

[11] Karl Engelbert Wenzel, Andreas Masselli, and Andreas Zell. Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 61 (1-4):221–238, 2011. ISSN 09210296. doi: 10.1007/s10846-010-9473-0.

[12] Stephen Chaves, Ryan Wolcott, and Ryan Eustice. NEEC Research: Toward GPS-denied Landing of Unmanned Aerial Vehicles on Ships at Sea. *Naval Engineers Journal*, 127(1):23–35, 2015. ISSN 0028-1425.

[13] Oualid Araar, Nabil Aouf, and Ivan Vitanov. Vision Based Autonomous Landing of Multirotor UAV on Moving Platform. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 85(2):369–384, 2017. ISSN 15730409. doi: 10.1007/s10846-016-0399-z. URL `http://dx.doi.org/10.1007/s10846-016-0399-z`.

[14] Janis Tiemann, Florian Schweikowski, and Christian Wietfeld. Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments. *2015 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2015*, pages 1–7, 2015. doi: 10.1109/IPIN.2015.7346960.

[15] Thien Minh Nguyen, Abdul Hanif Zaini, Kexin Guo, and Lihua Xie. An Ultra-Wideband-based Multi-UAV Localization System in GPS-denied environments. In *International Micro Air Vehicles Conference*, 2016.

[16] P. Nordin and J. Nygårds. Local navigation using traversability maps. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 7(PART 1):324–329, 2010. ISSN 14746670. doi: 10.3182/20100906-3-it-2019.00057.

[17] Peter Nordin, Lars Andersson, and Jonas Nygårds. Results of the tais/prerunners-project. In *Fourth Swedish Workshop on Autonomous Robotics SWAR'09*, pages 60–61, 2009.

[18] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58:1–35, 2006. ISSN 14602431. doi: 10.1093/jxb/erm298. URL `ftp://sbai2009.ene.unb.br/Projects/GPS-IMU/George/arquivos/Bibliografia/79.pdf`.

[19] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation Magazine*, 19(3):20–32, 2012. ISSN 10709932. doi: 10.1109/MRA.2012.2206474.

[20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 1960.

[21] Andrew H. Jazwinski. *Stochastic processes and filtering theory*. Number 64 in Mathematics in science and engineering. Academic Press, New York, 1970. ISBN 0123815509.

[22] Guobin Chang. Robust Kalman filtering based on Mahalanobis distance as outlier judging criterion. *Journal of Geodesy*, 88(4):391–401, 2014. ISSN 14321394. doi: 10.1007/s00190-013-0690-8.

[23] Yoichi Morales, Eijiro Takeuchi, and Takashi Tsubouchi. Vehicle localization in outdoor woodland environments with sensor fault detection. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 449–454, 2008. ISSN 10504729. doi: 10.1109/ROBOT.2008.4543248.

[24] Fredrik Gustafsson. *Statistical sensor fusion*. Studentlitteratur, 2010.

[25] Amir Beck, Petre Stoica, and Jian Li. Exact and approximate solutions of source localization problems. *IEEE Transactions on Signal Processing*, 56 (5):1770–1778, 2008. ISSN 1053587X. doi: 10.1109/TSP.2007.909342.

[26] OpenCV. Camera Calibration and 3D Reconstruction, 2019. URL `https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`.

[27] Duane Brown. Close-range camera calibration, 1971.

[28] Martin Enqvist, Torkel Glad, Svante Gunnarsson, Peter Lindskog, Lennart Ljung, Johan Löfberg, Tomas McKelvey, Anders Stenman, and Jan-Erik Strömberg. *Industriell reglerteknik kurskompendium*. Linköpings Universitet, Linköping, 2014.

[29] Elsevier Science & Technology. *Missile Guidance and Pursuit : Kinematics, Dynamics and Control*. Elsevier Science & Technology, 1998. ISBN 9781904275374.

[30] Guangwen Liu, Masayuki Iwai, Yoshito Tobe, Dunstan Matekenya, Khan Muhammad Asif Hossain, Masaki Ito, and Kaoru Sezaki. Beyond horizontal location context: Measuring elevation using smartphone's barometer. *UbiComp 2014 - Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 459–468, 2014. doi: 10.1145/2638728.2641670.

[31] Mustafa Cavcar. International Standard Atmosphere, in BS. *Fluid Mechanics in Channel, Pipe and Aerodynamic Design Geometries 2*, pages 251–258, 2018. doi: 10.1002/9781119457008.app5.

[32] Kexin Guo, Zhirong Qiu, Cunxiao Miao, Abdul Hanif Zaini, Chun-Lin Chen, Wei Meng, and Lihua Xie. Ultra-Wideband-Based Localization for Quadcopter Navigation. *Unmanned Systems*, 04(01):23–34, 2016. ISSN 2301-3850. doi: 10.1142/s2301385016400033.

[33] James Bowman and Patrick Mihelich. How to Calibrate a Monocular Camera, 2019. URL `http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration`.

[34] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011. ISSN 10504729. doi: 10.1109/ICRA.2011.5979561.

[35] John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. *IEEE International Conference on Intelligent Robots and Systems*, 2016-Novem:4193–4198, 2016. ISSN 21530866. doi: 10.1109/IROS.2016.7759617.

[36] Danylo Malyuta. Guidance, Navigation, Control and Mission Logic for Quadrotor Full-cycle Autonomy. Master's thesis, ETH Zurich, 2018.

[37] Maximilian Krogius, Acshi Haggenmiller, and Edwin Olson. Flexible Layouts for Fiducial Tags. *IEEE International Conference on Intelligent Robots and Systems*, pages 1898–1903, 2019. ISSN 21530866. doi: 10.1109/IROS40897.2019.8967787.

[38] Alvika Gautam, P. B. Sujit, and Srikanth Saripalli. Application of guidance laws to quadrotor landing. *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pages 372–379, 2015. doi: 10.1109/ICUAS.2015.7152312.

[39] Ruoyu Tan and Manish Kumar. Proportional navigation (PN) based tracking of ground targets by quadrotor UAVs. *ASME 2013 Dynamic Systems and Control Conference, DSCC 2013*, 1(October 2013), 2013. doi: 10.1115/DSCC2013-3887.

[40] Decawave. DWM1001 Firmware Application Programming Interface (Api) Guide. pages 1–74, 2017.