NTNU – Trondheim
Norwegian University of
Science and Technology

Cooperative Control and RTK Navigation System for Multirotors

Jon-Håkon Bøe Røli

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Thor Inge Fossen, ITK
Co-supervisor: Kristian Klausen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Jon-Håkon Bøe Røli
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Samarbeidende reguleringssystem og RTK navigasjonssystem for multirotorer
Thesis title (English):	Cooperative Control and RTK Navigation System for Multirotors

Thesis Description: The purpose of the thesis is to develop a cooperative control system for multirotors, using RTK GPS for accurate outdoor navigation.

The following items should be considered:

1. Define the scope of the thesis and clarify what your contributions are.
2. Literature overview on cooperative formation control.
3. Development and testing of RTK GPS for multirotors.
4. Develop a passivity-based method for cooperative formation control.
5. Develop a multirotor platform and demonstrate cooperative control with formation flying.
6. Conclude your findings in a report and discuss the weaknesses of the system, and how these can be resolved.

Start date:	2015-01-05
Due date:	2015-06-18

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor:	Kristian Klausen, Dept. of Eng. Cybernetics, NTNU

Abstract

This thesis considers the implementation of cooperative control on small, unmanned, multirotor systems. More specifically, the problem of distributed formation control is handled, as well as the necessary high precision navigation. The control of multiple Unmanned Aerial Vehicles (UAV) has seen a lot of development in the last years, but most advanced implementations use central processing and rely on motion capture systems for precision navigation, or simply cope with the low accuracy of a Global Navigation Satellite System (GNSS). This thesis uses an established passivity-based framework to achieve decentralized formation control, and investigates the use of Real-Time Kinematics (RTK) to augment the accuracy of a GNSS to the centimetre level. The development of a suitable multirotor platform is presented, and control is implemented with the combination of the open-source APM:Copter platform on a commercial autopilot and the DUNE: Uniform Navigational Environment on a payload computer. The resulting distributed system of UAVs is capable of fulfilling a flexible group objective while sustaining a desired formation, which is demonstrated in both simulations and field experiments.

Keywords: Unmanned Aerial Vehicle, Multirotor, Multicopter, Hexacopter, Global Navigation Satellite System, Real-Time Kinematics, Cooperative Control, Passivity, DUNE, Software-in-the-Loop, APM:Copter, Pixhawk, Piksi

Sammendrag

(Norwegian translation of the abstract)

Denne avhandlingen tar for seg implementeringen av et samarbeidene reguleringssystem for små, ubemannede multirotorsystemer. Mer spesifikt er problemet med distribuert formasjonskontroll taklet, i tillegg til nødvendig høy-presisjons navigasjon. Regulering av flere fjernstyrte, ubemannde luftfartøy (UAV), også kjent som *droner*, har sett stor utvikling de siste årene, men de fleste avanserte systemene bruker "motion capture" og prosessering på en sentral datamaskin. Eventuelt baserer de seg kun på den lave nøyaktigheten satelittbaserte navigasjonssystmer (GNSS) kan tilby. Denne avhandlingen bruker et etablert, passivitetsbasert rammeverk til å oppnå desentralisert formasjonskontroll, samt utforsker bruken av "Real-Time Kinematics" (RTK) til å utvide nøyaktigheten fra et GNSS til centimeternivå. Utviklingen av en passende multirotorplatform blir presentert, og regulering oppnås med kombinasjonen av en kommersiell autopilot med åpen kildekode (APM:Copter) og et navigasjonsmiljø (DUNE: Uniform Navigational Environment) kjørende på en datamaskin i nyttelasten. Det resulterende, distribuerte systemet av fartøy kan løse et fleksibelt, samlet mål mens gruppen opprettholder en ønsket formasjon, og er demonstrert med både simuleringer og feltforsøk.

Preface

This is a master's thesis concerning work done in the spring of 2015 at the Department of Engineering Cybernetics, submitted in partial fulfillment of the requirements for the degree Master of Science (MSc) at the Norwegian University of Science and Technology (NTNU), in the field of Engineering Cybernetics. The initiative for the thesis came from the Centre for Autonomous Marine Operations and Systems (AMOS), more precisely as part of AMOS' *Project 5: Autonomous aerial systems for marine monitoring and data collection.* It is assumed that the target audience has prior basic knowledge in the field of systems and control.

The development of the multirotor platform presented in Chapter 2, as well as the execution of all field experiments presented in Chapter 7, was performed in collaboration with fellow MSc. student Recep Cetin.

Trondheim, 2015-06-17

Acknowledgements

I would like to thank my supervisor, Professor Thor I. Fossen, for his valuable guidance and feedback, and for introducing me to this exciting project. A special thanks to Lars Semb from the Dept. of Engineering Cybernetics at NTNU, for his unmatched piloting skills. Furthermore, I am very grateful towards the technical staff at the electronic and mechanical workshops of the department, for their aid and open-door policy. I would also like to extend my gratitude to Nadezda Sokolova and Aiden Morrison from the Dept. of Communication Systems at SINTEF, for their insight and recommendations. A huge thanks goes to my fellow MSc student, Recep Cetin, for an excellent joint effort throughout this project. Moreover, this project would never have been possible without the extraordinary devotion from my co-supervisor, PhD candidate Kristian Klausen. Your inspirational leadership and avid interest for the project has helped me create something of which I am truly proud of.

Finally, I would like to thank my parents and siblings, for their never-ending support and affection.

Sincerely,

Jon-Håkon Bøe Røli

Contents

List of Figures

List of Tables

List of Abbreviations

AMOS Centre for Autonomous Marine Operations and Systems.

BBB BeagleBone Black.

CAD Computer-Aided Design.

CCW Counterclockwise.

CPU Central Processing Unit.

CW Clockwise.

DGNSS Differential GNSS.

DOF Degrees of Freedom.

DUNE DUNE: Uniform Navigational Environment.

ESC Electronic Speed Controller.

FPGA Field-Programmable Gate Array.

GCS Ground Control Station.

GDOP Geometric Dilution of Precision.

GLONASS Globalnaya Navigatsionnaya Sputnikovaya Sistema (Global Navigation Satellite System).

GNSS Global Navigation Satellite System.

GPIO General-Purpose Input/Output.

GPS Global Positioning System.

GUI Graphical User Interface.

IAR Integer Ambiguity Resolution.

IMC Inter-Module Communication.

IMU Inertial Measurement Unit.

IP Internet Protocol.

LSTS Laboratório de Sistemas e Tecnologias Subaquáticas (Underwater Systems and Technology Laboratory).

MCU Micro Computer Unit.

NAVSTAR Navigation System using Timing And Ranging.

NED North-East-Down.

NTNU Norges teknisk-naturvitenskapelige universitet (Norwegian University of Science and Technology).

OS Operating System.

PVT Position/Velocity/Time.

PWM Pulse-Width Modulation.

RC Radio Controller.

RF Radio Frequency.

RTK Real-Time Kinematics.

SBP Swift Navigation Binary Protocol.

SIL Software-in-the-Loop.

SINTEF Stiftelsen for industriell og teknisk forskning (The Foundation for Scientific and Industrial Research).

TCP Transmission Control Protocol.

UAV Unmanned Aerial Vehicle.

UAV-Lab Unmanned Aerial Vehicle Laboratory of AMOS.

UDP User Datagram Protocol.

Chapter 1

Introduction

1.1 Background and Motivation

The term cooperative control applies to a group of control problems where multiple autonomous agents work together to achieve some common goal. This area has seen extensive research in the last decades, as many benefits can be obtained when a single complicated agent is replaced by multiple, yet simpler agents. Multiple cheap agents can be more cost-efficient than one expensive, give increased redundancy to agent failures and often simply perform tasks more efficiently, e.g. through increased area coverage in a search or reduced drag by formation flying.

A distinction between a *centralized* and a *distributed* approach to the control of multiple agents is often made. In the centralized approach, the control of all agents are computed at a central station working with global information, essentially just an extension of the traditional single-agent control. By contrast, each agent in the distributed approach only acts on local information from sensing or communication devices. Consequently, information sharing plays a central role in the resulting behaviour in the distributed approach and the communication topology becomes essential to designing the controller for each agent.

An example of distributed cooperative control is the *agreement problem*, where the goal is for the agents to reach a certain common agreement over a variable of interest, e.g. a heading, attitude, position, phase etc. This is also known as *consensus* or *synchronization*. The main focus of this thesis is *formation control*, where the objective is to coordinate a group of agents to achieve some desired relative distance or position.

High precision navigation is necessary for advanced cooperative control, and many implementations use motion capture systems that limit applications outside the laboratory. Instead, this thesis has investigated the use of carrier phase differential measurements, also known as Real-Time Kinematics (RTK) navigation, to augment the accuracy

of a Global Navigation Satellite System (GNSS) to the centimetre level.

1.2 Previous Work

The simplest implementation of formation control is a master-slave strategy, where the master possibly follows some desired trajectory while the slaves only position themselves relative to the master. A semi-autonomous example of this is given in (Cheung, Chung, and Coleman 2009) where a leader robot is selected and teleoperated by an operator and follower robots are autonomously coordinated to make a formation, without intercommunication. However, the more common implementation is using synchronization by regarding formation control as a shifted agreement problem.

(Pettersen, Gravdahl, and Nijmeijer 2006) presents many interesting contributions from a workshop with focus on control theoretic challenges related to group coordination and cooperative control. An excellent overview of the past two decades of research in control of multi-agent systems is given by (Olfati-Saber, Fax, and Murray 2007; Murray 2007; Cao et al. 2013). Common for most applications is the use of graph theory to present the communication topology, and furthermore to analyze the resulting behaviour of the complete system (Wu and Chua 1995). In (Ren, Beard, et al. 2005; Moreau 2005), certain requirements to interaction graphs are shown to lead to asymptotic information consensus, even with unreliable and changing interaction topology. Further, a tutorial overview of strategies in information consensus is given in (Ren, Beard, and Atkins 2007).

A framework for coordinated control of multiple UAVs is presented in (Kaminer et al. 2007) and further extended in (Xargay et al. 2013), where the vehicles negotiate their speeds along their respective paths in response to information exchange. This forms a strategy for time-critical cooperative missions that yield robust behaviour against external disturbances. A similar approach is used in (Ghabcheloo et al. 2009), where coordinated path-following is achieved by adjusting speed profiles of virtual targets following assigned paths.

One approach to formation control mentioned by (Murray 2007) is to formulate it as an optimization problem by introducing cumulative formation errors between neighbours as cost functions. (Dunbar and Murray 2006) solves this problem using receding horizon optimal control. Another approach mentioned by (Murray 2007) is to shape the dynamics of the formation using potential fields. In this case the control law for individual vehicles is the gradient of a potential function based on the state of the vehicle and its neighbours. The work of (Leonard and Fiorelli 2001) uses a concept of artificial potentials inspired by biological systems. However, their resulting controller is criticized for not guaranteeing convergence to a unique desired formation (Olfati-Saber and Murray 2002). Instead, (Olfati-Saber and Murray 2002) proposes a framework using potential functions obtained from the structural constraints of a desired formation. A virtual structure approach is used in (Broek, Wouw, and Nijmeijer 2009), where mutual coupling between unicycle mobile robots is used to achieve formation control.

A passivity-based design tool for cooperative control was proposed in (Arcak 2007), where bidirectional communication topology and internal feedback is used to transform the interconnected system into a cascaded set of passive systems. This leads to a systematic construction of Lyapunov functions for stability analysis. The resulting framework was further developed, and a good summary of this research is presented in (Bai, Arcak, and Wen 2011), along with several examples of implementations. The formation problem is solved by synchronizing relative positions or distances between communicating agents, while a common mission velocity in the internal feedback achieves the group objective. A second approach to formation control based on (Ihle, Arcak, and Fossen 2007) is also given, where the desired formation is achieved in a similar fashion to (Ghabcheloo et al. 2009), but passivity is used for proving stability. Moreover, the flexibility offered by the framework is proved useful in developing adaptive feedback laws, for instance when the mission velocity is only known to the leader.

Formation control and path following for underactuated marine surface vessels is analyzed in (Børhaug et al. 2011; Belleter and Pettersen 2014) using nonlinear cascaded systems theory.

As noted in (Cao et al. 2013), much of the research on consensus only consider simple agent dynamics like single integrator kinematics or double integrator dynamics. Indeed the latter is used in both (Leonard and Fiorelli 2001) and (Olfati-Saber and Murray 2002). An extension from the simple dynamics to general linear dynamics is given in (Qu, Wang, and Hull 2008), using a framework based on matrix theory. Furthermore, the proposed method in (Chung and Slotine 2009) permits highly nonlinear dynamics, using contraction to prove exponential stability. The passivity approach in (Bai, Arcak, and Wen 2011) also allows systems of high order and nonlinear dynamics. In fact, the framework is applicable as long as a passive controller can be made. This is trivial for Newtonian systems, but can also be accomplished for Lagrangian and Hamiltonian systems.

Research on the effects of time delays on synchronization is also well covered in (Cao et al. 2013). An interesting result mentioned for systems with single integrator kinematics is that only input delay, and not communication delay, affect the convergence and performance of consensus, referred to as *consensusability*. The main research question is to determine an upper bound of the time delay under which time delay does not affect the consensusability. Generally speaking, consensus with time delays for nonlinear systems is more challenging, but still, consensus can be achieved under a relatively large communication delay (Cao et al. 2013).

1.3 Contribution and Scope of This Thesis

The overall goal of this thesis is to design and implement a system of multirotors capable of performing tasks in a cooperative manner. One intended future application is cooperative lifting of suspended loads as presented in (Klausen, Fossen, Johansen, et al. 2014).

This report covers the development of the multirotor platform aided by RTK navigation, as well as the implementation of the necessary cooperative control. The passivity-based framework presented in (Bai, Arcak, and Wen 2011) will be used to solve the latter, as the strict implications from passivity, combined with the modularity and flexibility of the framework, would allow adaptation of the resulting system to solve a large range of cooperative tasks. Further, RTK navigation will be achieved using a new receiver called *Piksi*, chosen because its low price and open-source software is promising for the broadening of RTK applications. This project is also an important first step for the Unmanned Aerial Vehicle Laboratory of AMOS (UAV-Lab), as no field experiments with cooperative control of multiple UAVs have yet been attempted.

To summarize, the purpose of the project can be described by the following objectives:

Multirotor Platform
: Establish the necessary software and hardware to form a multirotor platform suitable for cooperative control.

RTK GPS
: Explore the use of Piksi as a high precision navigation system for multirotors.

Formation Control
: Implement distributed formation control where multirotors converge to a given formation.

Common Objective
: The group of multirotors should be able to solve a common objective, e.g. moving to a target location.

The complete system will be verified through simulations, and further demonstrated in field experiments, with the formation flight of multiple vehicles. Note that the multirotor platform is developed in cooperation with another MSc student, Recep Cetin (Cetin 2015). In addition, suspended load control and an alternative indoor navigation system is presented in (Cetin 2015).

To limit the scope of this thesis, the following assumptions are made:

- A low-level controller for the multirotor is already available, capable of achieving desired attitudes and rates.
- All multirotors are able to communicate with each other, i.e. the communication topology is known and constant.

- Collision avoidance is disregarded (although a short discussion is given in Section 4.2.4). This should not be a problem if initial positions are taken into consideration with the desired formation.

1.4 Organization of This Thesis

This introductory chapter ends with a list of notations and useful definitions used throughout the thesis.

Chapter 2 presents the multirotor platform used in this project. This involves the different hardware components and the design of a custom payload module. An introduction to the necessary software tools is also given, before the mathematical model is derived.

An introduction to RTK navigation is given in Chapter 3, followed by a short discussion about its application for UAVs. Furthermore, Piksi is presented, with a discussion of issues, in addition to several experiments investigating accuracy and robustness.

Cooperative control is presented in Chapter 4, where the passivity-based approach from (Bai, Arcak, and Wen 2011) is applied to the multirotor platform. Moreover, the resulting feedback laws are verified by simulation.

Chapter 5 combines the three previous chapters into a realization of distributed formation control for multirotors. Further, Software-in-the-Loop (SIL) simulations of the complete system is conferred in Chapter 6, before finally presenting field experiments with formation flights in Chapter 7.

Results from the preceding chapters are briefly summarized in Chapter 8, which generally concludes the thesis. Lastly, a short discussion of future work is given.

A collection of supplementary figures is presented in Appendix A.

The digital appendix provided with this thesis contains data from simulations and field experiments, as well as the code used for extraction, processing and plotting. Animations of the data sets presented in Chapter 6 and 7 is also given. However, the code developed as part of the implementation in Chapter 5 is not included, as this is available at the UAV-Lab Git repository, under the branch *copter-dev* (AMOS 2015).

1.5 Notation and Definitions

- The set of all *real* numbers is denoted $\mathbb{R}$.
- All vectors in this thesis are column vectors. $\mathbb{R}^p$ denotes the set of $p \times 1$ real vectors, while the set of real matrices with p rows and q columns are denoted $\mathbb{R}^{p\times q}$. In addition, all vectors and matrices are given in bold font (e.g. $x \in \mathbb{R}$ is a scalar while $\boldsymbol{x} \in \mathbb{R}^p$ is a vector).
- $\boldsymbol{A}^\mathsf{T} \in \mathbb{R}^{q\times p}$ denotes the *transpose* of the matrix $\boldsymbol{A} \in \mathbb{R}^{p\times q}$.
- The inverse of an invertible matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is denoted $\boldsymbol{A}^{-1} \in \mathbb{R}^{n\times n}$.
- The superscript † of a matrix $\boldsymbol{A} \in \mathbb{R}^{p\times q}$ denotes its Moore–Penrose *pseudoinverse*:

$$\boldsymbol{A}^\dagger := \boldsymbol{A}^\mathsf{T}(\boldsymbol{A}\boldsymbol{A}^\mathsf{T})^{-1} \in \mathbb{R}^{p\times q} \tag{1.1}$$

- For a vector $\boldsymbol{x} \in \mathbb{R}^p$, $|\boldsymbol{x}|$ denotes its 2-norm, i.e. $|\boldsymbol{x}| = \sqrt{\boldsymbol{x}^\mathsf{T}\boldsymbol{x}}$.
- $\mathscr{R}(\boldsymbol{A})$ and $\mathscr{N}(\boldsymbol{A})$ denotes the *range space* and *null space* of a matrix $\boldsymbol{A}$, respectively, while the empty set is denoted by $\emptyset$.
- $\boldsymbol{I}_p$ is the $p \times p$ *identity matrix* and $\boldsymbol{1}_p$ is the $p \times 1$ column vector of ones, while 0 is always a scalar, vector or matrix of zeros with a compatible dimension.
- The notation $\text{diag}\{\boldsymbol{K}_1, \boldsymbol{K}_2, \cdots, \boldsymbol{K}_n\}$, where $\boldsymbol{K}_i \in \mathbb{R}^{p\times q},\ i \in \{1, \ldots, n\}$, is used to denote the block diagonal matrix:

$$\begin{bmatrix} \boldsymbol{K}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{K}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{K}_n \end{bmatrix} \in \mathbb{R}^{np\times nq} \tag{1.2}$$

- The labeling of reference frames are done by a lower-case letter in curly brackets, and vectors represented in said reference frame is displayed by the same letter in superscript (e.g. $\boldsymbol{x}^n$ is a vector represented in {n}).
- A star in superscript denotes a desired value, e.g. $\boldsymbol{x}^\star$ is a vector of desired values for $\boldsymbol{x}$. Likewise, a zero in superscript denotes an initial value.
- The superscript F denotes a formation specification, e.g. $\boldsymbol{x}_i^F$ defines the desired relative formation for agent i.
- The subscript *sat* indicates the saturation value for the absolute value (2-norm) of the respective scalar (vector), e.g. the maximum speed of the velocity vector $\boldsymbol{v} \in \mathbb{R}^3$ is denoted by $v_{sat} =: max(|\boldsymbol{v}|) \in \mathbb{R}$.

- The Kronecker product of two matrices $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{B} \in \mathbb{R}^{p\times q}$ is defined as:

$$\boldsymbol{A}\otimes\boldsymbol{B} := \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{mp\times nq} \tag{1.3}$$

  and satisfies the properties:

$$(\boldsymbol{A}\otimes\boldsymbol{B})^{\mathsf{T}} = \boldsymbol{A}^{\mathsf{T}}\otimes\boldsymbol{B}^{\mathsf{T}} \tag{1.4a}$$

$$(\boldsymbol{A}\otimes\boldsymbol{I}_p)(\boldsymbol{C}\otimes\boldsymbol{I}_p) = (\boldsymbol{A}\boldsymbol{C})\otimes\boldsymbol{I}_p \tag{1.4b}$$

  where $\boldsymbol{A}$ and $\boldsymbol{C}$ are assumed to be compatible for multiplication.

- A function is said to be C^k if its partial derivatives exist and are continuous up to order k or higher.

- Given a C^2 function $P : \mathbb{R}^p \to \mathbb{R}$, ∇P and $\nabla^2 P$ denotes its gradient vector and *Hessian* matrix, respectively.

- $\boldsymbol{S}(\cdot)$ is the cross-product matrix operator defined such that the cross-product of two vectors $\boldsymbol{\lambda} \in \mathbb{R}^3, \boldsymbol{x} \in \mathbb{R}^3$ can be represented as:

$$\boldsymbol{\lambda}\times\boldsymbol{x} := \boldsymbol{S}(\boldsymbol{\lambda})\boldsymbol{x} \tag{1.5}$$

  where $\boldsymbol{S}$ is skew-symmetric and defined as:

$$\boldsymbol{S}(\boldsymbol{\lambda}) = -\boldsymbol{S}^{\mathsf{T}}(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix} \tag{1.6}$$

Chapter 2

The Multirotor Platform

This chapter presents the multirotor platform used in the thesis, as well as in (Cetin 2015). Firstly, an overview is given, before presenting the individual system components. Next, the design of a custom payload is discussed, followed by an introduction to the different software tools used. Finally, the dynamics of the vehicle is derived.

2.1 Overview

The complete system can be divided into three segments:

- A base station
- The multirotors with payload
- Mission control with the Ground Control Station (GCS) and pilots

As will be discussed in Chapter 3, a base station is necessary for providing observations for Piksi on each multirotor, solving the high-precision relative navigation of the vehicles. Figure 2.3 gives an overview of the full system and the interaction of its hardware and software components, while a picture of the complete multirotor platform and base station can be seen in Figure 2.1 and Figure 2.2, respectively.

Figure 2.1: The complete multirotor with the custom payload and all components.

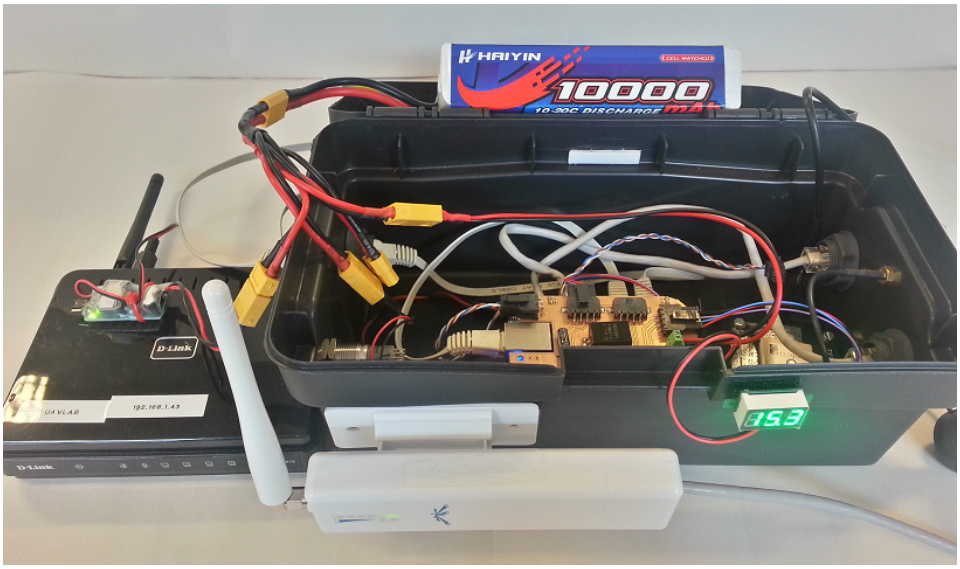


Figure 2.2: The base station, together with a switch and access point.

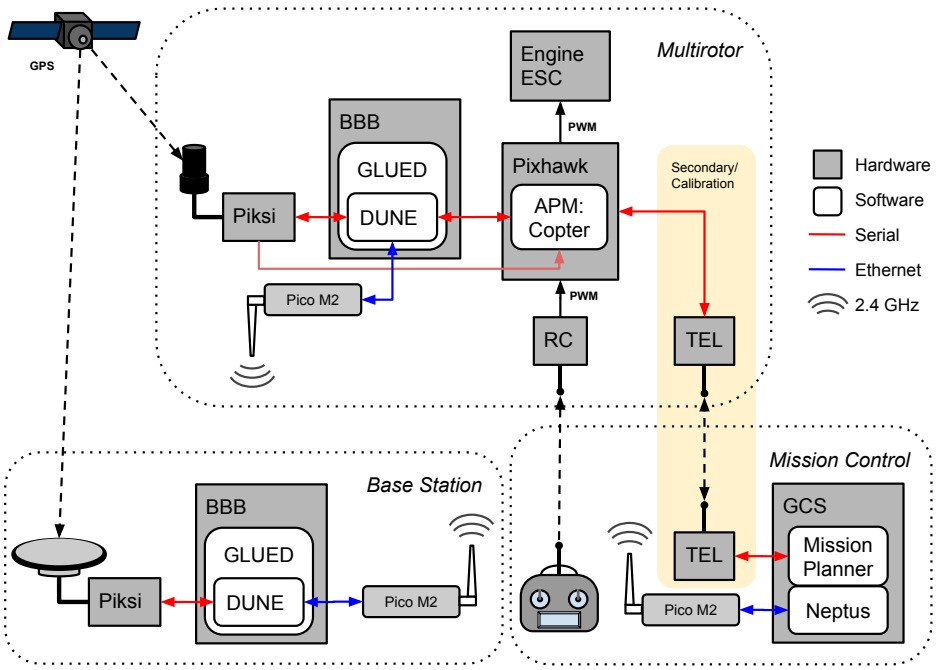


Figure 2.3: Overview of the complete system. A solid line illustrates a wired connection, while a dashed line illustrates a wireless connection. Arrowheads indicate the direction of information flow.

2.2 System Components

2.2.1 Multirotor

A hexacopter, a multirotor with six arms and rotors, is considered to be a good compromise between size, cost and weight. The ArduCopter Hexa B (3D Robotics 2014a), shown in Figure 2.4, has been used with great success in previous projects at NTNU (Steen 2014; Andersen 2014), and was used as the base multirotor platform in this thesis.

Figure 2.4: The ArduCopter Hexa B from 3D Robotics. The blue colored support arms define the forward direction of the airframe. *Image courtesy of* 3drobotics.com.

2.2.2 Autopilot

The autopilot is responsible for controlling the speed of the 6 rotors of the multirotor through Pulse-Width Modulation (PWM) signals to their individual Electronic Speed Controller (ESC). Pixhawk, shown in Figure 2.5a, was chosen as autopilot and it runs the APM:Copter software, described in Section 2.5, to achieve the necessary low-level control.

(a) The Pixhawk autopilot. *Image courtesy of* 3drobotics.com.

(b) The GPS/compass module. *Image courtesy of* 3drobotics.com.

Figure 2.5: The Pixhawk autopilot, designed by the PX4 open-hardware project and manufactured by 3D Robotics, along with the external GPS/compass module.

Pixhawk also contains sensors required for basic navigation: An Inertial Measurement Unit (IMU) to determine the rotation and acceleration, a barometer for relative altitude measurements and the external GPS/compass module shown in Figure 2.5b (not illustrated in Figure 2.3) to determine the position and heading (3D Robotics 2014c).

2.2.3 Precision navigation

For high-precision positioning of the multirotor, this thesis has used a Real-Time Kinematics (RTK) receiver called Piksi. Piksi, shown in Figure 2.6, outputs high-precision position and velocity relative to the base station on each multirotor. RTK and Piksi is further presented in Chapter 3, where a discussion of its antennas is also given.

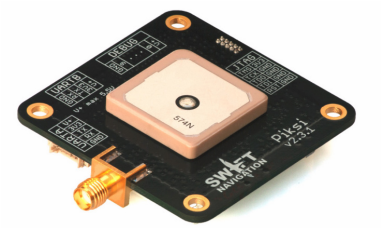


Figure 2.6: The Piksi RTK receiver from Swift Navigation. *Image courtesy of* `swiftnav.com`.

2.2.4 Radio controller

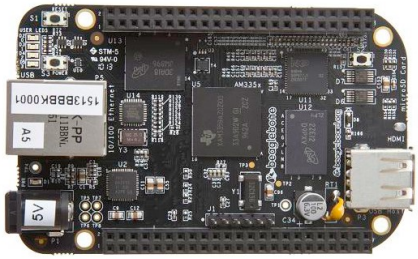
A Radio Controller (RC) is the typical tool used by a pilot for manual control of a UAV. The Spektrum DX7s shown in Figure 2.7 was used in this thesis. It sends a PWM signal to a receiver on the hexacopter, giving commands to Pixhawk. The autopilot then does the actual control of the hexacopter, as it is impossible for a human operator to stabilize it by commanding the speed of the individual motors.

Figure 2.7: The Spektrum DX7s with its receiver. *Image courtesy of* `hobbyfly.com`.

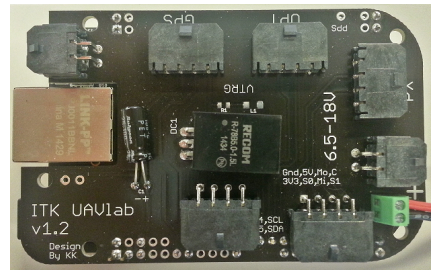
2.2.5 Payload computer

A small, low-cost platform for developers called BeagleBone Black (BBB) (BeagleBoard 2014) was chosen as the payload computer. Its use in a multirotor payload was advised by (Steen 2014), and (Klausen 2013) used it with great success on small autonomous robots.

The BBB is the brain connecting all the important components together. It provides guidance for the multirotor through Pixhawk and interfaces the on-board Piksi to receive precision navigation. BBB runs tasks in DUNE to achieve this, on a minimal Linux distribution called GLUED (see Section 2.4.1). A BBB is also used to interface Piksi at the base station.



(a) The BBB. *Image courtesy of* beagleboard.org.



(b) The BBB with the custom made add-on board attached.

Figure 2.8: The payload computer, BeagleBone Black, with and without the custom made add-on board that provides power and simple connectivity to peripherals.

To supply the BBB with power and simplify connections to other components, a custom add-on board was made. The BBB, as well as the add-on board, can be seen in Figure 2.8.

2.2.6 Ground control station

The Ground Control Station (GCS) runs a software called Neptus to interact with DUNE on the payload computer and perform command and control. Neptus is also the tool responsible for data collection and analysis. A short introduction is given in Section 2.4.2.

The yellow shaded link in Figure 2.3 is an optional telemetry link from Pixhawk, used to interface another control software called Mission Planner (3D Robotics 2014b) on the GCS. A short description of Mission Planner is given in Section 2.5.1, but the only intended use in the system is as a simple tool to calibrate Pixhawk and configure settings in APM:Copter.

2.2.7 Communication

The PicoStation M2 HP displayed in Figure 2.9 is used to provide wireless network communication between the the different units running DUNE or Neptus. It is a compact, but powerful 2.4 GHz radio used as the main communication link in this project (Ubiquiti Networks 2014). With the unit on the base station configured as an access point and the other as clients, all the units can communicate seamlessly with each other through TCP/IP.

Figure 2.9: The PicoStation M2 HP from Ubiquiti Networks, with and without its plastic housing.

The secondary calibration link between Mission Planner and Pixhawk is a set of two simple 433 Mhz radios from 3D Robotics (3D Robotics 2015), shown in Figure 2.10.

Figure 2.10: The 433 MHz radio from 3D Robotics. *Image courtesy of* 3drobotics.com.

2.2.8 Batteries

Figure 2.11: The three different lithium polymer batteries used.

Three different lithium polymer (LiPo) batteries, shown in Figure 2.11, was used:

- An 800 mAh *payload battery* connected to the add-on board in Figure 2.8b, providing the BBB and the other payload components on the multirotor with power.
- A 4000 mAh *rotor battery* connected to the ESCs, providing the rotors with power.
- A 10000 mAh *nest battery*, providing the base station with power in the same manner as the payload battery on the multirotor.

With the payload powered by a separate battery, the system avoids being reset each time the rotor battery is depleted. This was a big advantage as the rotor battery only provided a flight time of approximately 5 to 10 minutes. The Pixhawk is in fact powered by both the payload and rotor battery, as it provides essential voltage monitoring for the latter.

2.3 Custom Payload

The ArduCopter Hexa B has a lot of room for payloads both beneath and above the hexacopter body. The hexacopter comes with a removable top carrier plate with the possibility to add smaller extra levels using separators (see Figure 2.12). These provide simple mounting of payload for many uses. Together with a custom made bottom payload, (Steen 2014) and (Andersen 2014) used the standard top payload plates. In this project however, most of the payload had to fit on top of the hexacopter. This was because of the intended future application in cooperative lifting, mentioned in Section 1.3, where connection mechanisms and other relevant equipment will need the space below the hexacopter body. The design also had to conform with the needs of the cooperative project (Cetin 2015), where the mounting of an additional complicated part was necessary: The *Atmel kit* shown in Figure 2.19a.

Consequently, with a lot of parts to fit and some being nontrivial to mount, the need to design custom top payload plates emerged. Note that all dimensions in this chapter are given in millimeters.

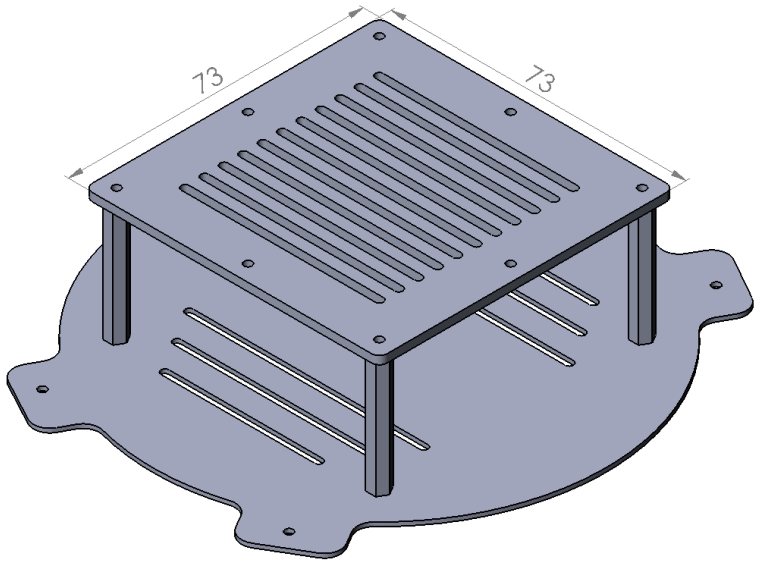


Figure 2.12: Original top carrier plate of the 3DR Hexa with one extra level attached with separators. *Model courtesy of Kristian Klausen.*

2.3.1 Specifications

The top payload plate had to provide space and proper mounting for the following components:

- Pixhawk
- BeagleBone Black
- Piksi
- Atmel kit
- PicoStation M2 HP
- Helix antenna for Piksi
- GPS/Compass for Pixhawk
- Payload battery

Other small parts, like the receiver for the RC or the secondary calibration link was simply attached to the arms of the hexacopter with Velcro.

Further, the following properties were considered:

Lightweight
Less weight means better endurance and maneuverability.

Symmetry and balance
The hexacopter should be kept as symmetrical as possible for optimal balance and hence, maneuverability. Some balancing correction can be achieved when mounting the rotor battery.

GPS antennas and magnetic compass
These should be mounted above all other components for an unobstructed view of the sky, and far from noise heavy components to minimize disturbance. They should also be placed far from each other to prevent mutual coupling (more than half the wavelength, i.e. > 9.5 cm).

Atmel and Picostation antennas
As these are fixed to the body of the components, the placement of the components themselves had to be taken into consideration. The antennas should be placed far from each other as to minimize interference, as well as far out on the hexacopter body itself to get the best possible reception in all directions.

Pixhawk
This had to be placed as close as possible to the center of gravity of the hexacopter and aligned with the axes of the axes of the {b} frame (see Section 2.7.1), as this is where the IMU is located.

Accessibility
All components should be kept as accessible as possible for easy maintenance.

2.3.2 Design

A CAD software (SolidWorks 2014) was used to design the necessary parts. Co-supervisor, PhD candidate Kristian Klausen, had already made models of the original Hexa B, which together with models and mock-ups of the payload components simplified the design job.

A design with three separate levels was chosen. First a new carrier plate that was directly connected to the rest of the hexacopter body, replacing the original. Then, two smaller plates were designed to be mounted on top of the carrier plate, using the original separators.

Carrier plate

The Pixhawk was naturally aligned with the axes of the {b} frame and placed in the center of the carrier plate, while the placement of the other components was less trivial. Several different placements and designs were made (see Figure 2.13), before ending up with the current version shown in Figure 2.15. This has a mount for the PicoStation, two antenna mast placements (numbered 1 and 2) and a special groove to fit an Atmel kit mount, shown in Figure 2.19b. Figure 2.14 shows a top down view, with the placement of the components indicated.

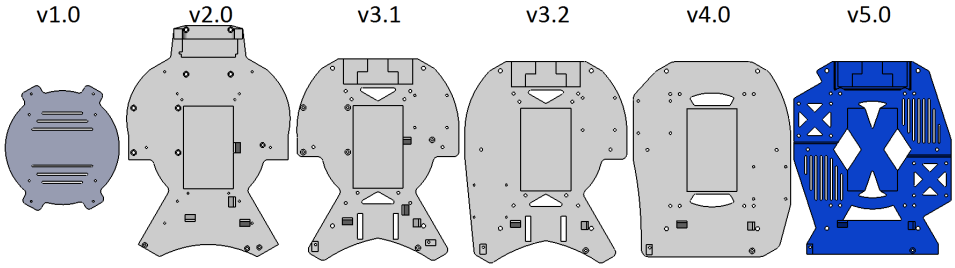


Figure 2.13: Different generations of the carrier plate, all in relative scale. From the original v1.0 from 3D Robotics to the final v5.0 used in this thesis.

The Atmel kit and the PicoStation was placed in opposite ends of the carrier plate, such that their antennas could point down and outside the hexacopter body. Cutouts were made to reduce weight and provide holes for cables. Two ribbed surfaces (marked A and B in Figure 2.14) provide possible placements for the payload battery. Because the electronics of the stripped (to reduce weight and size) PicoStation became very exposed on the carrier plate, a simple cover was added, which is fastened by connecting the antenna and the RJ-45 Ethernet plug. Furthermore, this cover and the mount for PicoStation was covered by copper tape in an attempt to mitigate problems of interference with Piksi. The shielded housing can be seen in Figure 3.8, and the interference problem is further discussed in Section 3.5.

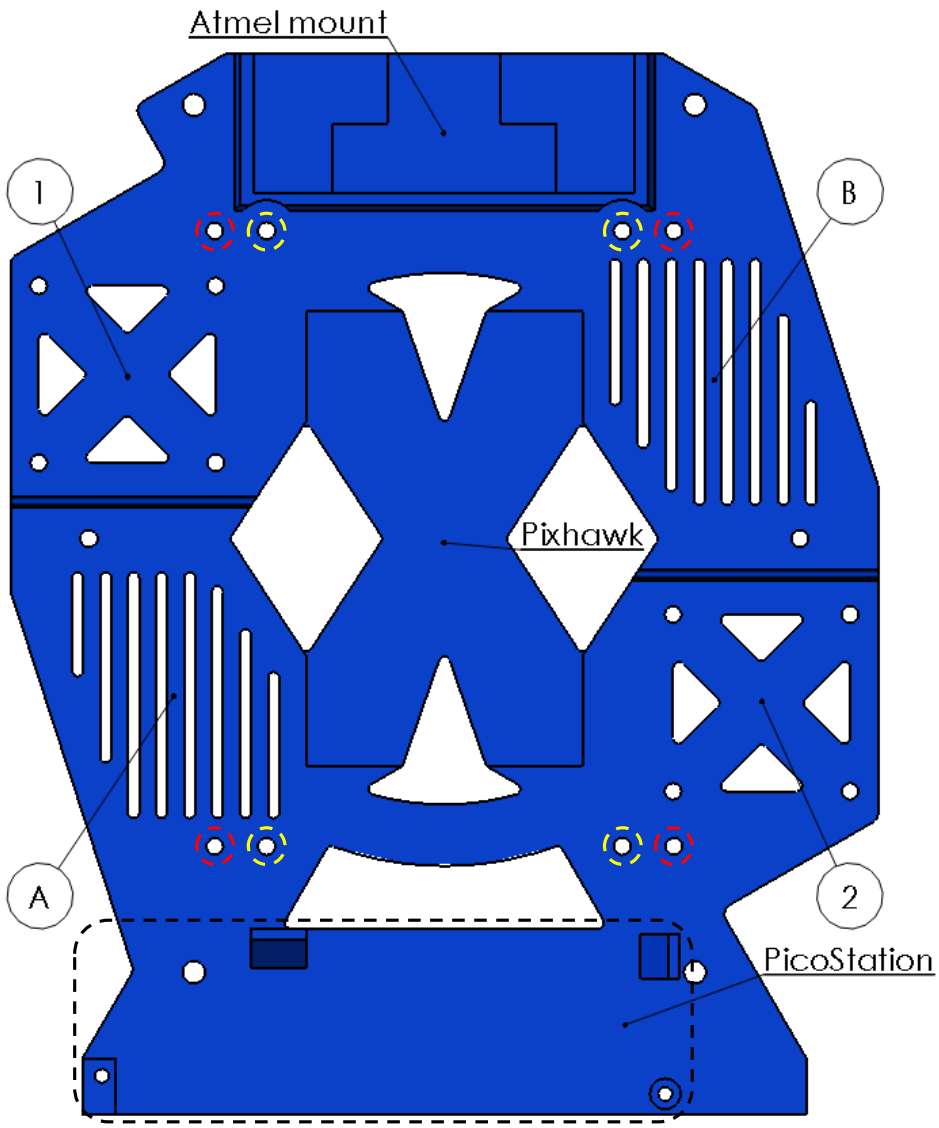


Figure 2.14: Layout of the final version of the carrier plate. Antenna mast and battery placements are marked by numbers and letters, respectively. The dashed red and yellow circles indicate the attachment points for the separators and the attachment to the hexacopter body, respectively.

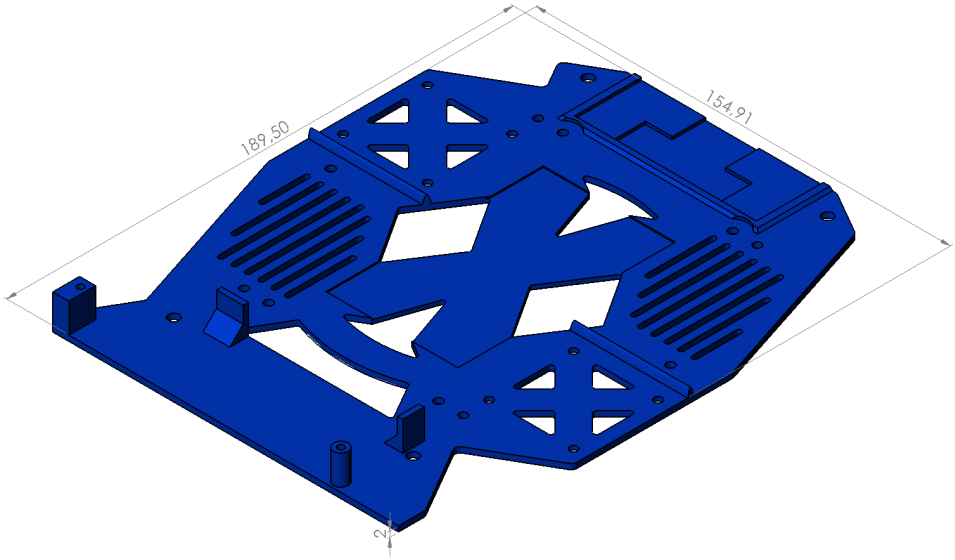


Figure 2.15: Dimensions of the final version of the carrier plate.

MCU plate

The payload computer, BeagleBone Black (BBB) was placed on the second level plate, which appropriately was named the Micro Computer Unit (MCU) plate. Piksi was also placed at this level.

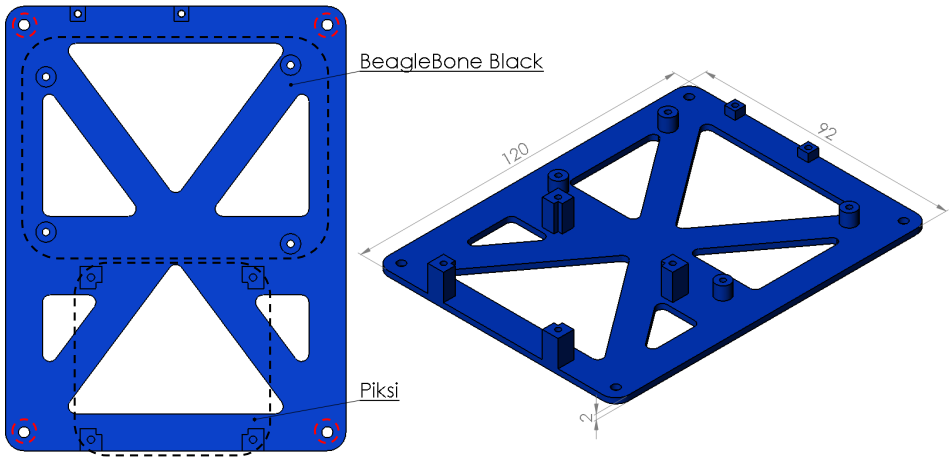
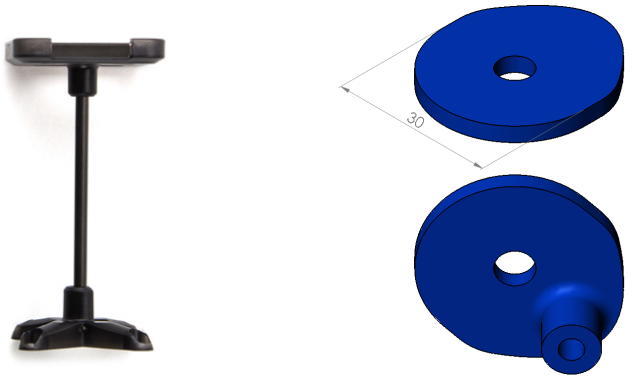


Figure 2.16: The MCU plate. A mount for a voltage display is visible on the top edge. The dashed red circles indicate the attachment points for the separators.

GPS plate and antenna masts

A GPS mast sold by 3D Robotics (Figure 2.17a) was used to mount both the helix GNSS antenna for the Piksi and the combined GPS/compass for the Pixhawk. This mast is designed to fit the latter, but replacing the top platform with the simple, custom designed mount shown in Figure 2.17b, it also provided a simple way to attach the helix antenna.



(a) The GPS mast. *Image courtesy of 3drobotics.com*.

(b) The helix antenna mount.

Figure 2.17: The GPS mast sold by 3D Robotics for the GPS/compass module for Pixhawk (a) and the helix antenna mount (b).

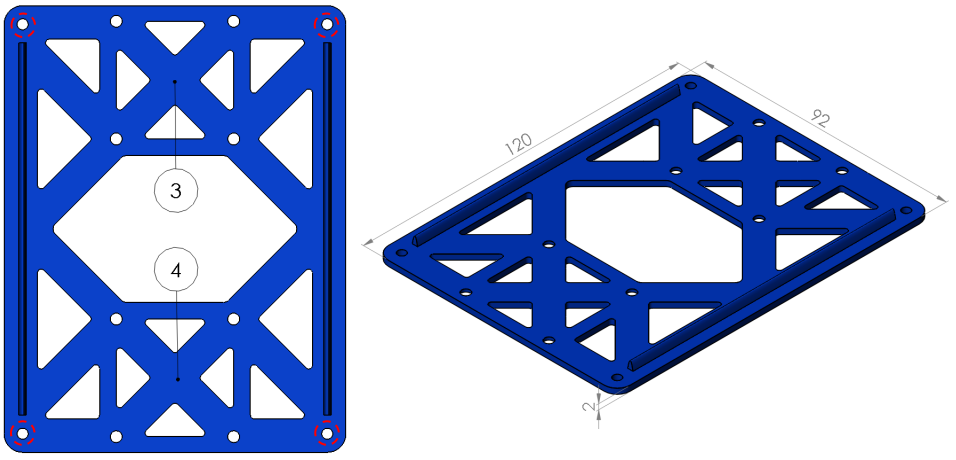
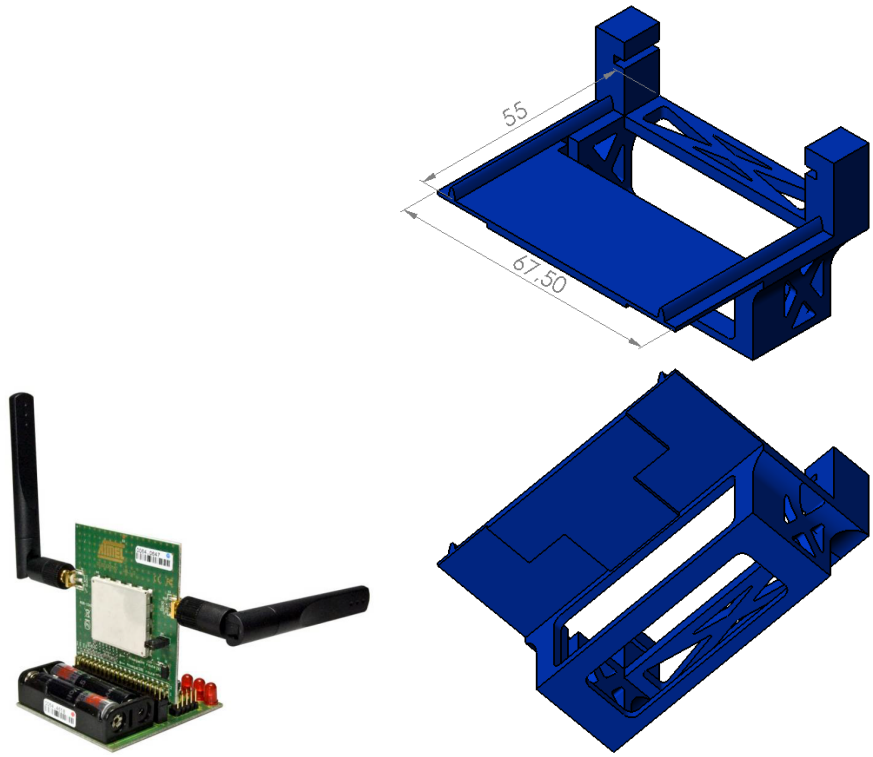


Figure 2.18: The GPS plate with antenna mast placements 3 and 4. The dashed red circles indicate the attachment points for the separators.

These antenna masts were initially placed in opposite "corners" of the carrier plate (i.e. 1 and 2 in Figure 2.14), as a good distance between them prevented both mutual coupling and physically blocking GPS signals for each other. However, after encountering the interference from PicoStation, discussed in Section 3.5, a third level with two more placements for antenna masts was designed and mounted on top of the MCU plate. This *GPS plate* is shown in Figure 2.18.

Atmel kit mount

Designing a proper way to attach the Atmel kit proved a challenge. Whereas the other components had screw holes, providing a simple means of attachment, the Atmel kit did not. A box-like container, exploiting its two-part connection to lock it in place, was designed. Because this shape would make the carrier plate complex to print (see the next section), it was designed as a separate, third part of the payload plate, to be glued to the carrier plate instead. The finished design is shown in Figure 2.19b.



(a) The Atmel kit. *Image courtesy of* atmel.com.

(b) The designed mount. The bottom view clearly shows the track to fit in the carrier plate.

Figure 2.19: The Atmel kit used by (Cetin 2015) and its designed mount.

2.3.3 Production and assembly

The plates were printed in plastic using a 3D printer. The optimal material would probably have been carbon fiber, providing extreme durability compared to its weight. However, plastic was lightweight and durable enough. Moreover, the ability of rapid prototyping with a 3D printer was a big advantage.

A 3D printer builds parts by adding successive layers of material. It is therefore convenient if parts only have extruding features on one side, limiting the amount of support structure needed and, hence, the time required for printing. This benefit was the reason for designing the mount for the Atmel kit as a separate part, at the expense of structural integrity.

The completed payload with custom printed plates and components mounted can be seen mounted on the multirotor in Figure 2.1, where all parts fit nicely into their designed placement.

2.3.4 Improvements to future designs

The designs were made with accessibility in mind for easy prototyping, neglecting protection and covering. This is beneficial as changes are often made along the way, and the need for a new connection here and there can emerge. It would be pretty trivial in future designs to build covers and lids around the exposed components on the carrier plate and maybe turning the MCU plate into an MCU box.

2.4 Introduction to the LSTS Software Toolchain

The Laboratório de Sistemas e Tecnologias Subaquáticas (Underwater Systems and Technology Laboratory) (LSTS) is a an interdisciplinary research laboratory specializing on the development of unmanned underwater, surface and air vehicles, as well as tools and technologies for the deployment of networked systems of said vehicles (LSTS 2015f). LSTS has developed a software toolchain consisting of the runtime environment DUNE, the command and control software Neptus and the IMC protocol.

Powerful and open-source (LSTS 2015d), the LSTS toolchain was used as the software backbone for the multirotor platform in this thesis and an introduction to each of its components is given below.

2.4.1 DUNE

DUNE: Uniform Navigational Environment (DUNE) is the on-board software running at the heart of each vehicle, interfacing all peripherals. The system is both CPU architecture and OS independent, making it highly portable (LSTS 2015a).

Written in C++, DUNE is based on the idea of defining libraries of tasks, each responsible for controlling one element on-board the vehicle, e.g. different sensors, actuators, autopilots or communication devices. At runtime the desired set of tasks (and their parameters) are defined in a simple *initialization file* for each vehicle, causing them to be run on separate threads or processes, while using the concept of message passing to communicate with each other or tasks running on other vehicles.

GLUED

Targeted at embedded systems, GLUED is a minimal Linux distribution designed by the LSTS for easily configurable cross compilation (LSTS 2015b). It is the OS running on the BBB used in this project, where DUNE is executed.

2.4.2 Neptus

Neptus is the command and control infrastructure running on the GCS, solving the problem of operator-vehicle interaction with a GUI (LSTS 2015e). The software has three main features:

- Planning
- Execution
- Review and Analysis

By adding profiles for each vehicle, the software can be used in planning operations by simulating and validating before execution. During the execution, the software can visualize real-time data and send mission plans to multiple vehicles. Neptus is also used for post-mission data extraction and analysis from vehicles.

Neptus is written in Java and supports the development of independent plug-ins, making it a very flexible and customizable tool.

2.4.3 IMC

The Inter-Module Communication (IMC) protocol is used for all communication between the different tasks internally in DUNE, as well as between different vehicles and Neptus at the GCS (LSTS 2015c). It provides a large set of shared message definitions that can be serialized and transferred over different media. The protocol can easily be extended by adding new message definitions.

2.5 Introduction to APM:Copter

APM:Copter is an open-source platform for multirotors and helicopters developed by the DIY Drones community (DIY Drones 2015b). Based on the Arduino platform (from which its name originates; ArduPilotMega), it offers a wide range of autopilot solutions that comprise both remote and autonomous flight control. The latter includes waypoint navigation, mission planning and telemetry on a ground station for monitoring. The modularity and the capabilities of the software is extensive, and the interested reader is referred to (DIY Drones 2015a). There are several flight modes available in APM:Copter, out of which four was used in this project:

Stabilize
: Pilot input to the RC is interpreted as desired angles in roll and pitch, and the average speed of the rotors. Without input the copter will automatically level itself. This is the mode allowing the most aggressive maneuvering from the pilot, while still aiding with stability.

Loiter
: The copter tries to maintain a consistent location and orientation. RC input from the pilot is interpreted as the adjustment of said location and orientation, resulting in simpler and smoother control. Unlike Stabilize, this mode is dependant on a GPS fix.

Guided
: Also GPS dependant, Guided is an autonomous mode where the user can dynamically guide the copter to travel to a target location or achieve a desired velocity. This mode is used when DUNE provides said desired locations or velocities to achieve higher-level control.

Land
: Another autonomous mode that makes the copter perform a controlled descent. This mode is set to activate upon detection of low battery voltage to avoid crashing as a result of engine power failure.

2.5.1 Mission Planner

Mission Planner is the GCS of APM:Copter. It can be connected to the vehicle using a telemetry radio and facilitate a command and control center for the vehicle. Mission Planner enables the use of point-and-click waypoint entries, using Google maps or other local or global maps. Missions containing a set of commands can be saved and executed easily. Real-time vehicle telemetry is also shown, including the output from the Pixhawk's serial terminal. Moreover, it offers an interface with a PC flight simulator. However, the most interesting part for this thesis is the configuration of the APM settings for our airframe. Mission Planner includes an easy and intuitive GUI to tune parameters and optimize the autopilot's behaviour.

2.5.2 Micro Air Vehicle Link

Pixhawk uses a header-only message marshalling library called MAVLink (DIY Drones 2015e). This Micro Air Vehicle communication protocol serves as the backbone for the internal communication, as well as communication with DUNE and the ground link.

2.6 Piksi Software

2.6.1 Swift Navigation Binary Protocol

The Swift Navigation Binary Protocol (SBP) is the native protocol used by Piksi to transmit solutions, observations, status and debugging information (SwiftNAV 2015e). A portable C implementation was provided as a part of *libswiftnav*, a platform independent library of GNSS related functions and algorithms for use by software-defined receivers and other software (SwiftNAV 2015b). This has since been moved to its own library, *libsbp* (SwiftNAV 2015c), although this was after the former version of libswiftnav was used to interface Piksi in this project (see Section 5.1).

SBP is primarily used as a fast and simple protocol to send the binary representation of C structs across serial links, without any error correction and without delivery guarantees.

2.6.2 Piksi Console

The Piksi Console is a GUI written in Python to interface Piksi through SBP. It allows Piksi to communicate with a computer through any serial port and display satellite tracking, baseline, position solutions and status information, in addition to observations sent and received between modules. The console has the same features as a configuration utility. It is the simplest way to download and install new firmware on Piksi and various settings of Piksi can be accessed and altered on-the-fly.

During the RTK experiments presented in Section 3.4, the Piksi Console was used for monitoring and saving data for analysis. However, for the rest the project, the console was only used for configuration.

2.7 Multirotor Dynamics

The dynamics of the multirotor platform is derived in this section, using similar notation and nomenclature as (Fossen 2011). The geometrical aspects of motion, also known as the *kinematics*, is first presented, before analyzing the forces causing the motion, i.e. the *kinetics* (Fossen 2011).

2.7.1 Kinematics

A multirotor has six degrees of freedom (6 DOF) as it can both move and rotate in 3D space. This thesis uses two different geographic reference frames to represent these, illustrated in Figure 2.20.

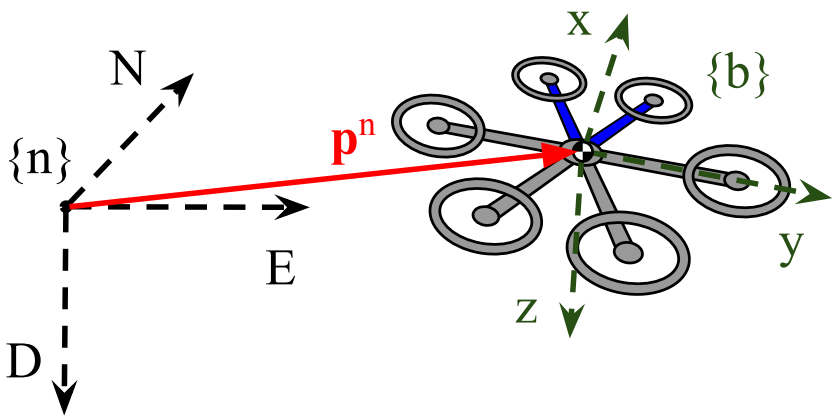


Figure 2.20: Illustration of the inertial {n} and body-fixed {b} reference frames, used to represent the vehicle kinematics. Note that for the hexacopter used in this thesis, the x-axis in {b} is centered between the two blue front airframe support arms.

First, let {n} denote the (right-hand) *north-east-down* (NED) coordinate system defined relative to the Earth's reference ellipsoid (Smith 1987). Further, {n} is assumed to be inertial such that Newton's laws still apply (Fossen 2011). The position of the vehicle can then be represented by the vector:

$$\boldsymbol{p}^n := \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathbb{R}^3 \tag{2.1}$$

Next, let {b} denote the (right-hand) *body-fixed* reference frame that is a moving coordinate frame fixed to the center of mass of the vehicle: The x- and y-axis point in the

forward and starboard direction of the vehicle, while the z-axis points downwards along the normal axis to complete the right-hand convention. The attitude of the vehicle can then be described by the relative orientation of {b} with respect to {n}. Using the Euler angles roll (ϕ), pitch (θ) and yaw (ψ) that describe a principal rotation about the x, y and z axes, respectively, the attitude can be represented as:

$$\boldsymbol{\Theta} := \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \in \mathbb{R}^3 \tag{2.2}$$

The linear and angular velocity of the vehicle can then also be represented in {b} relative to {n} by the respective vectors:

$$\boldsymbol{\nu}^b := \begin{bmatrix} u \\ v \\ w \end{bmatrix} \in \mathbb{R}^3, \quad \boldsymbol{\omega}^b := \begin{bmatrix} p \\ q \\ r \end{bmatrix} \in \mathbb{R}^3 \tag{2.3}$$

Linear velocity in {b} is often named surge (u), sway (v) and heave (w).

Furthermore, the conversion from {b} to {n} can be represented by the (zyx convention) rotation matrix $R_b^n(\boldsymbol{\Theta}) \in SO(3)$ (Fossen 2011):

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{2.4}$$

where c and s is shorthand forms for cosine and sine, respectively. As a result, the linear velocity represented in the inertial {n} frame, $\boldsymbol{\nu}^n := [\dot{x}, \dot{y}, \dot{z}]^\mathsf{T} \in \mathbb{R}^3$ is then given by the transformation:

$$\boldsymbol{\nu}^n = \boldsymbol{R}_b^n(\boldsymbol{\Theta})\, \boldsymbol{\nu}^b \tag{2.5}$$

2.7.2 Kinetics

The multirotor is essentially a very simple machine, consisting of several individual rotors attached to a rigid airframe in an evenly distributed circular pattern, all pointing upwards. Control of the multirotor is thus achieved by differential control of the thrust generated by each rotor: Pitch and roll is achieved by adjusting the relative speed of opposing rotors, while heave is a function of the total thrust of all rotors. By alternating the rotation of every rotor, yaw control is obtained by adjusting the average speed of those rotating Clockwise (CW) relative to that of the ones rotating Counterclockwise (CCW). However, no thrust is achievable in surge or sway, making the system *underactuated*.

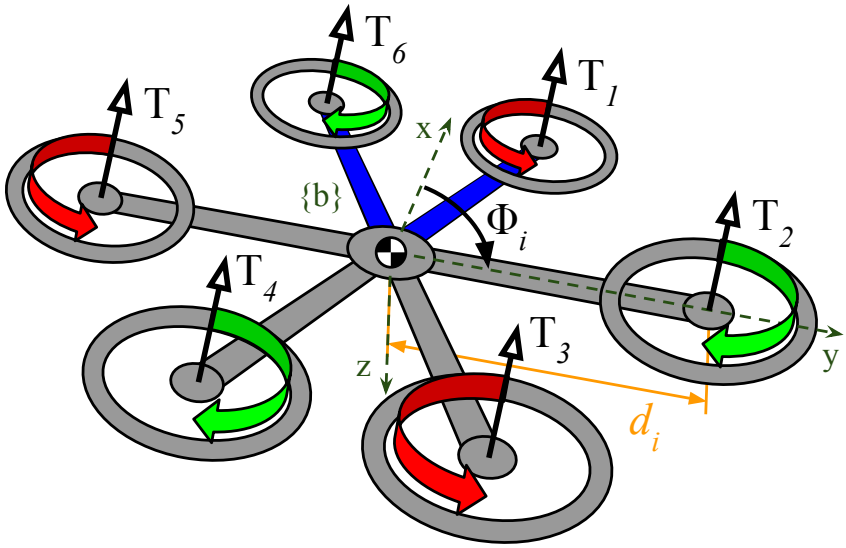


Figure 2.21: Notation for the multirotor equations of motion, here illustrated for the hexacopter used in this thesis. CW and CCW rotating rotors are represented as green and red arrows, respectively. Note that for this airframe, $\Phi_i = i\frac{\pi}{3} - \frac{\pi}{6}$.

Let the N rotors of the multirotor be labeled $i \in \{1,\dots,N\}$ in a clockwise direction as shown in Figure 2.21. Each rotor has associated an angle Φ_i between its airframe support arm and the x-axis in {b}, as well as a distance d_i from the central axis of the vehicle.

To find the mapping from forces and moments to rotor inputs, the thrust T_i generated by each individual rotor i can be modelled by the quadratic model:

$$T_i = c_T \omega_i^2 \tag{2.6}$$

where ω_i is the angular speed. $c_T > 0$ is a coefficient dependant on the geometry, profile, radius and area of the rotor, as well as the air density. In practise it can be modeled as a constant that can be easily determined from static thrust tests (Mahony, Kumar, and Corke 2012). Likewise, the reaction torque generated from each rotor can be modelled as:

$$Q_i = c_Q \omega_i^2 \tag{2.7}$$

where the coefficient c_Q has similar dependencies as c_T and can be determined in the same manner.

The total thrust force applied to the vehicle is the sum of the the thrusts from each individual rotor:

$$T_\Sigma := \sum_{i=1}^{N} |T_i| = c_T \sum_{i=1}^{N} \varpi^2 \tag{2.8}$$

Further, the net moments produced about the axes in {b} can be expressed as:

$$K = - c_T \sum_{i=1}^{N} d_i sin(\Phi_i) \varpi^2 \tag{2.9a}$$

$$M = c_T \sum_{i=1}^{N} d_i cos(\Phi_i) \varpi^2 \tag{2.9b}$$

$$N = c_Q \sum_{i=1}^{N} \sigma_i \varpi^2 \tag{2.9c}$$

where $\sigma_i \in \{-1, +1\}$ denotes the direction of rotation of the ith rotor; -1 corresponding to CW and $+1$ to CCW. This can be written in matrix form:

$$\begin{bmatrix} T_\Sigma \\ K \\ M \\ N \end{bmatrix} = \underbrace{\begin{bmatrix} c_T & \cdots & c_T \\ -c_T d_1 sin(\Phi_1) & \cdots & -c_T d_N sin(\Phi_N) \\ c_T d_1 cos(\Phi_1) & \cdots & c_T d_N cos(\Phi_N) \\ \sigma_1 c_Q & \cdots & \sigma_N c_Q \end{bmatrix}}_{\mathbf{\Gamma}} \begin{bmatrix} \varpi_1^2 \\ \varpi_2^2 \\ \vdots \\ \varpi_N^2 \end{bmatrix} \tag{2.10}$$

where $\mathbf{\Gamma} \in \mathbb{R}^{4xN}$ is a constant matrix, dependant on the number and placement of rotors. Thus, given the desired total thrust and moments, we can solve for the required rotor speeds. For $N = 4$, i.e. a *quadcopter*, $\mathbf{\Gamma}$ is a $4x4$ matrix and it is simply a matter of taking the inverse $\mathbf{\Gamma}^{-1}$. However, in this thesis a hexacopter ($N = 6$) is used, and for this reason the pseudoinverse $\mathbf{\Gamma}^\dagger$ would have to be used instead.

By defining $\boldsymbol{f}^b \in \mathbb{R}^3$ and $\boldsymbol{m}^b \in \mathbb{R}^3$ as the nonconservative forces and moments, respectively, acting on the vehicle in {b} we have[1]:

$$\boldsymbol{f}^b := \begin{bmatrix} 0 \\ 0 \\ -T_\Sigma \end{bmatrix} + \mathbf{\Lambda}, \quad \boldsymbol{m}^b := \begin{bmatrix} K \\ M \\ N \end{bmatrix} \tag{2.11}$$

where $\mathbf{\Lambda} \in \mathbb{R}^3$ comprises secondary aerodynamic forces whose main factors are induced drag and rotor flapping (Mahony, Kumar, and Corke 2012). $\mathbf{\Lambda} = 0$ in the neutral hover position, and the terms are small for modest translational velocities. In addition they

[1] Note the negative sign in $\boldsymbol{f}^b$ as the z-axis is defined positive downwards.

act as nonlinear damping, often coined as "good" nonlinear terms (Khalil 2002, Section 13.4), which in fact help improve the stability of the system. Hence, $\boldsymbol{\Lambda}$ is disregarded when further developing the dynamics of the multirotor.

Finally, the 6 DOF equations of motion for the multirotor are (Mahony, Kumar, and Corke 2012):

$$\dot{\boldsymbol{p}}^n = \boldsymbol{\nu}^n \tag{2.12a}$$

$$m\dot{\boldsymbol{\nu}}^n = m\boldsymbol{g}^n + \boldsymbol{R}_b^n(\boldsymbol{\Theta})\boldsymbol{f}^b \tag{2.12b}$$

$$\dot{\boldsymbol{R}}_b^n(\boldsymbol{\Theta}) = \boldsymbol{R}_b^n(\boldsymbol{\Theta})\boldsymbol{S}(\boldsymbol{\omega}^b) \tag{2.12c}$$

$$\boldsymbol{I}\dot{\boldsymbol{\omega}}^b = -\boldsymbol{S}(\boldsymbol{\omega}^b)\boldsymbol{I}\boldsymbol{\omega}^b + \boldsymbol{m}^b \tag{2.12d}$$

where $m \in \mathbb{R}$ and $\boldsymbol{I} \in \mathbb{R}^3$ denotes the mass and constant inertia matrix of the vehicle, while $\boldsymbol{g}^n = [0, 0, g]$ is the acceleration of gravity. Note that the equations of motion is split into a *translational*[2] (2.12a)-(2.12b) and a *rotational* (2.12c)-(2.12d) part. This separation will be useful in the further discussions.

2.7.3 Translational control

If only translational control of the multirotor is of interest, we can develop purely translational equations of motion. To do this, we first assume that there exists low-level control capable of keeping a desired attitude $\boldsymbol{\Theta}^\star$ and a desired thrust in heave $T_\Sigma^\star$. A simple linearized PD control law is shown to guarantee stability for small deviations from the hover position in (Mahony, Kumar, and Corke 2012). Moreover, they present an alternative nonlinear controller that is guaranteed to be exponentially stable for almost any rotation (Lee, Leoky, and McClamroch 2010).

Secondly, for small roll and pitch angles $\Delta\phi, \Delta\theta$, and an arbitrary yaw angle ψ_0, we can linearize (2.4) about the neutral hover position, i.e. $\boldsymbol{\Theta}_0 = [0, 0, \psi_0]^\mathsf{T}$. The resulting linear rotation matrix, $\boldsymbol{R}_\Delta$ is then (Mahony, Kumar, and Corke 2012):

$$\boldsymbol{R}_\Delta(\Delta\boldsymbol{\Theta}) = \begin{bmatrix} c\psi & -s\psi & \Delta\theta c\psi + \Delta\phi s\psi \\ s\psi & c\psi & \Delta\theta s\psi - \Delta\phi c\psi \\ -\Delta\theta & \Delta\phi & 1 \end{bmatrix} \tag{2.13}$$

where $\psi = \psi_0 + \Delta\psi$. The nominal force from the rotors in the neutral hover position is $T_{\Sigma 0} = mg$ to counter the force gravity. Using (2.13) and linearizing (2.12a)-(2.12b) about

[2] Even though *translational* and *linear* velocity is often used interchangeably, the former is used solely for describing velocity represented in the inertial {n} frame in this thesis.

$\boldsymbol{\Theta}_0$, $T_{\Sigma 0}$ we get (Mahony, Kumar, and Corke 2012):

$$m\ddot{x} = -mg(\Delta\phi s\psi_0 + \Delta\theta c\psi_0) \tag{2.14a}$$
$$m\ddot{y} = mg(\Delta\phi c\psi_0 - \Delta\theta s\psi_0) \tag{2.14b}$$
$$m\ddot{z} = -T_\Sigma + mg \tag{2.14c}$$

or in matrix form:

$$m\ddot{\boldsymbol{p}}^n = \boldsymbol{\Upsilon}\boldsymbol{F} + m\boldsymbol{g}^n \tag{2.15}$$

where:

$$\boldsymbol{\Upsilon} := \begin{bmatrix} 0 & -mgs\psi_0 & -mgc\psi_0 \\ 0 & mgc\psi_0 & -mgs\psi_0 \\ -1 & 0 & 0 \end{bmatrix} \tag{2.16}$$

is an invertible matrix defining the mapping between the vector of total rotor thrust and roll/pitch, $\boldsymbol{F} = [T_\Sigma, \phi, \theta]$, and the resulting translational acceleration.

Consequently, a desired translational force $\boldsymbol{\tau}^{n,\star} \in \mathbb{R}^3$ can be achieved by inverting (2.16) and solving for the desired total rotor thrust and roll/pitch set points $\boldsymbol{F}^\star = [T_\Sigma^\star, \phi^\star, \theta^\star]$:

$$\boldsymbol{F}^\star = \boldsymbol{\Upsilon}^{-1}(\boldsymbol{\tau}^{n,\star} - m\boldsymbol{g}^n) \tag{2.17}$$

also compensating for the force of gravity. Finally, inserting (2.17) into (2.15), we are left with the second order expression for the translational dynamics of the multirotor:

$$m\ddot{\boldsymbol{p}}^n = \boldsymbol{\tau}^{n,\star} \tag{2.18}$$

Note that this expression implies that a certain translational force would give rise to infinite acceleration. In reality, the damping from $\boldsymbol{\Lambda}$ that was neglected in (2.11) will always ensure a terminal velocity.

Chapter 3

Real-Time Kinematics Navigation

While an IMU provides the necessary measurements for stabilization, navigation for UAVs is usually highly dependant on a Global Navigation Satellite System (GNSS). The normal precision of such systems is however in the range of a few meters, which is insufficient in a cooperative control setting if multiple UAVs are to work together in close proximity.

Real-Time Kinematics (RTK) satellite navigation is a technique used to enhance the precision of a GNSS (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007). Instead of just using the information content of the received signals, it also utilizes phase measurements of the carrier waves, resulting in centimetre-level accuracy. However, this accuracy is only gained in the relative navigation of two receivers, as the comparison of said phase measurements is necessary to remove errors.

This chapter starts with the theory behind RTK, before presenting *Piksi*, the RTK receiver used in this thesis. A set of experiments investigating the accuracy and robustness of the receiver is also given, before ending the chapter with a short summary.

3.1 Introduction to RTK

In this section, the general concept of a GNSS and its extension to RTK navigation is briefly explained. The theory is based on (Svendsen 2001) and (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007), and the Global Positioning System (GPS) is used as an example, as this is currently the only GNSS for which Piksi supports RTK functionality (see Section 3.3).

3.1.1 The general concept

There are several GNSSs today, out of which the American Navigation System using Timing And Ranging (NAVSTAR), or just GPS, is the best known. Others are the Russian GLONASS and the European Galileo. All these satellite-based systems gives the users the opportunity to determine their own position, velocity and time in a common reference frame on a continuous basis (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007). This task is accomplished based on the two following principles:

- The velocity of a signal is determined by the distance covered and the time used.
- An electromagnetic signal travels through the atmosphere close to the speed of light (roughly 300,000 km per second).

All the three major GNSSs mentioned above emit signals that can be described by the three-layer model illustrated in Figure 3.1, where the frequencies of the GPS L1 band has been used.[1] The first layer characterizes the *physical* properties of the transmitted signals, while the second *ranging code layer* describes the method of measuring the propagation time from satellite to receiver. Finally, the *data-link layer* contains the navigation message, which among other information contains the time of transmission and satellite ephemerides (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007).

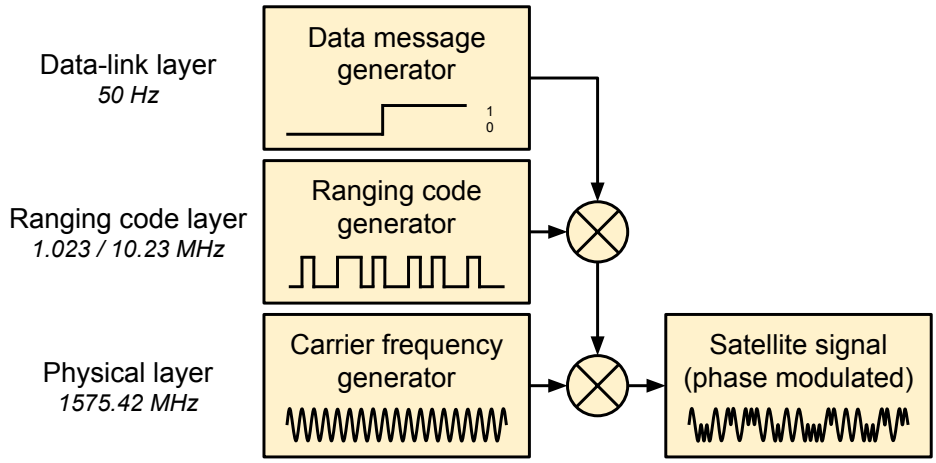


Figure 3.1: Composition of the general navigation satellite signal, with frequencies from GPS L1 given as an example. Each layer is successively multiplied onto the next, forming the complete phase modulated satellite signal.

[1] GPS does in fact transmit in several bands. However, as discussed in Section 3.3 Piksi currently only utilizes the L1 band, hence it is the only one mentioned here.

3.1.2 Code and phase measurements

For GPS, the ranging code is a continuous but periodic signal, strictly synchronized to the satellite clock. By generating and time shifting its own version of said signal, a receiver uses correlation techniques to precisely determine the propagation time of a measured signal from a given satellite. Further, by using satellite ephemerides in the data message to calculate the position of each satellite, the propagation times are effectively converted to the relative distance to each of these positions. These range measurements are actually called *pseudoranges*, as the difference between receiver and satellite clocks introduce potentially large biases. Each pseudorange R can hence be modeled as (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007):

$$R = c\triangle t + c\triangle\delta = \varrho + \triangle\varrho \tag{3.1}$$

where c is the speed of light, $\triangle t$ is the propagation time and $\triangle\delta$ is the difference between the satellite and receiver clock. ϱ is the actual distance between satellite and receiver, while $\triangle\varrho$ is the range correction resulting from the clock errors. This range correction is common for each satellite, assuming that their clocks are synchronized. As a result, a minimum of four pseudoranges (hence four satellites) is required to solve for the range correction $\triangle\varrho$, in addition to the 3D position of the receiver. Based on the ranging code, these measurements are appropriately called *code measurements*.

Because of the higher frequency, i.e. shorter wavelength, a more accurate alternative is to use the *phase measurements* Φ of the carrier signal (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007):

$$\lambda\Phi = \varrho + \triangle\varrho + \lambda N \tag{3.2}$$

where λ is the wavelength and N the number of integer carrier cycles between the satellite and the receiver, known as the *integer ambiguity*. This unknown ambiguity is the downside of pseudoranges obtained through phase measurements. The nontrivial task of solving for N is known as Integer Ambiguity Resolution (IAR), and is further discussed in Section 3.1.5.

Table 3.1: Frequencies, wavelengths and resulting theoretical measurement resolutions for the GPS ranging codes and L1 carrier signal.

Signal	Frequency [MHz]	Wavelength [m]	Resolution [mm]
C/A	1.023	293.26	2932.55
P	10.23	29.33	293.26
L1	1575.42	0.19	1.90

Using a measurement precision of 1 %, Table 3.1 compares the theoretical measurement resolution of the ranging codes[2] and carrier signal for GPS L1, where the latter obviously gives potential for much higher precision in navigation.

Velocity determination is in both cases achieved by using the Doppler principle of radio signals (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007). Because of the relative motion of the satellites with respect to a moving receiver, the frequency of a signal broadcast by the satellites is shifted when received at the vehicle. With the satellite velocity available through the ephemerides in the navigation message, the receiver velocity can be determined. This concept is not further explained here, but it can be mentioned that, like the pseudoranges, a minimum of four Doppler observables is required to solve for the velocity.

3.1.3 Sources of error

The pseudoranges from both code (3.1) and phase measurements (3.2) are affected by additional systematic errors or biases, as well as random noise. A short description of some of the typical sources of such errors is given below.

Satellite clock error
Poor synchronization between the clocks of different satellites can induce errors. However, each satellite has four atomic clocks, and a system that keeps the different satellites synced make sure that this error is only in the range of 1 nanosecond.

Ephemeris data
The precision of the calculated position of satellites naturally affects the precision of measurements.

Ionospheric delay
The ionosphere extends through various layers from about 50 km to 1000 km. Gas in this region is ionized by ultraviolet rays from the sun, causing the release of free electrons, which in turn affects the propagation of electromagnetic signals. The error is usually in the order of 1-10 m, but can be as large as 50 m. An approximation of the entire vertical ionospheric refraction known as the *Klobuchar model* is used to counter the induced delays. The coefficients of this model are broadcast by the GPS navigation message and it is thus implemented in most GPS receivers.

Tropospheric delay
Stretching from sea level to an altitude of about 15 km, the troposphere induces an error that can vary from about 2.4 m to about 25 m, depending on the angle between the satellite and the horizon. Temperature, pressure and humidity, which affects the delay, can be modeled and used to mitigate the imposed error.

Multipath
Multipath errors are caused by signals reflecting from surfaces near the receiver,

[2] The L1 signal uses two different ranging signals: The coarse/acquisition (C/A) code and the precision (P) code (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007).

causing an indirect (and hence longer) path from the satellite. Satellites at low elevations are more susceptible to multipath than those at high elevations. The use of a metal ground plane or choke rings can help remove multipath.

Dilution of precision
In addition to the accuracy of the individual pseudoranges, the geometry of the given satellite constellation affects the accuracy of the solution. This Geometric Dilution of Precision (GDOP) cannot be corrected by modelling, and hence makes it preferable to base a solution on widely separated satellites.

3.1.4 Differential GNSS and RTK

By combining measurements from two separate receivers, errors and noise common to observations from the same satellite can be removed in a process called *single differencing*. Likewise, receiver-specific errors can be eliminated by also comparing observations from several different satellites, similarly named *double differencing*. As a consequence, double-differenced pseudoranges are, to a high degree, free of systematic errors originating from the satellites and receivers (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007).

Differential positioning with GNSS, or DGNSS, thus requires a reference or base station receiver that transmits corrections to a second receiver. This means that it is only the relative navigation of the two receivers that benefit from the increased accuracy. However, if the correct global position of the base is known, the global accuracy of the other can be extrapolated with the same accuracy. DGNSS with phase measurements is used for the most precise kinematic applications, and this method is what is usually denoted as the Real-Time Kinematics (RTK) technique (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007) when performed in real-time. As mentioned earlier, this also involves the IAR, which is discussed next.

3.1.5 Integer ambiguity resolution

Integer Ambiguity Resolution (IAR) is a very active and advanced research area and only a brief explanation of the concept is mentioned here.

Since the magnitude of the systematic errors mentioned in Section 3.1.3 are much larger than the theoretical measurement resolution of the phase pseudoranges, removing them through double differencing is necessary to be able to isolate and solve the integer ambiguity in (3.2). Moreover, solving this ambiguity is not straightforward and has to utilize changes in satellite geometry: With a constant constellation there are more unknowns than equations, due to the fact that each new satellite adds a new ambiguity. However, as the constellation changes, new linearly independent equations can be added, and thus the observability of the ambiguities increase over time (Svendsen 2001).

Many different search algorithms have been developed to speed up the estimation of ambiguities and the interested reader is referred to (Svendsen 2001) and (Hofmann-Wellenhof, Lichtenegger, and Wasle 2007), where the most commonly used are presented in detail. Generally, an increased number of common satellites between two receivers will increase the speed of IAR algorithms, and utilizing known constraints, e.g. a known initial baseline between the two, can help solve the ambiguities in much shorter time.

Float and fix

During the IAR it is common to distinguish between a *float* and a *fixed* solution, where the former is a real valued estimate based on several *hypotheses*. When the algorithm is left with a single hypothesis, the ambiguity is said to be resolved or fixed, and we get the correspondingly named *fixed* solution.

3.2 Navigation System for Multirotors

With a price tag ranging from US$ 6,500 to US$ 25,000, RTK systems has traditionally been expensive and hence limited to applications like geodetic surveying, construction machine control or precision control of agriculture machinery. A growing Chinese market has put pressure on the high cost, with products often priced as low as 25 % of the equivalent brand-name products.[3]

The release of the open-source software package *RTKLIB* (Takasu and Yasuda 2009) has brought many low-cost applications of RTK. With RTKLIB and consumer-grade receivers able to output raw GPS data, users can construct and operate their own low-cost RTK GPS system. This has since been used in several applications for UAVs, e.g. (Skulstad and Syversen 2014; Bäumker, Przybilla, and Zurhorst 2013; Stempfhuber 2013). The downside is that RTKLIB has to run on a separate computer.

A complete RTK solution called *Piksi* was crowdfounded in 2013, with a lightweight and low-power design targeted at UAVs (Kickstarter 2014). Piksi is a GPS receiver with an on-board open-source mircocontroller that solves the RTK navigation (SwiftNAV 2013), directly outputting high-precision position and velocity solutions. Moreover, with the low price of US$ 995 for a complete kit with two receivers and a radio link to connect them (SwiftNAV 2015d), Piksi opens the door to lots of new applications. However, one drawback is that the open-source firmware for the microcontroller is still under development, with a prevalence of bugs that currently impedes robust and reliable applications. Still, with the continuous improvement of said problems, Piksi is a promising platform that this project has chosen for further examination.

[3] Prices are based on common market knowledge, and was confirmed by a sales representative from Leica Geosystems (`http://www.leica-geosystems.no/`)

3.3 Piksi

The Piksi RTK receiver measures the baseline, i.e. the relative distance, between itself and a second receiver. As mentioned in Section 3.1.4, this involves comparing the phase measurements from satellites, hereby referred to simply as *observations*, between the two receivers. Piksi needs observations from a minimum of 5 common satellites to produce RTK positioning (SwiftNAV 2013). It also produces highly accurate velocity measurements, which is not dependant on observations from another receiver.

Piksi's architecture consists of three main components:

- An RF **front-end** that downconverts and digitizes the RF signal from the antenna.
- An **FPGA** which performs basic filtering and correlation operations on the digitized signal stream.
- A **microcontroller** which controls the FPGA, programs the correlation operations, collects the results, and processes them all the way to PVT solutions.

While the current FPGA and microcontroller firmware is only capable of utilizing the L1 GPS code, the front-end is also able to receive Galileo and GLONASS signals, making future improvement to accuracy and IAR possible (Svendsen 2001). According to (SwiftNAV 2013), Piksi is theoretically capable of calculating PVT solutions at 50 Hz, but the developers have stated at the community forum (SwiftNAV 2015a) that a more realistic figure for the current firmware is around 20 Hz.

An overview of the peripherals of Piksi is shown in Figure 3.2.

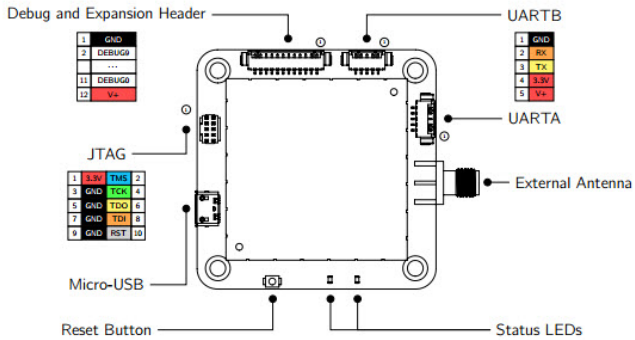


Figure 3.2: Connections for the Piksi RTK receiver. *Image courtesy of* swiftnav.com.

This section will first consider some of the challenges with the fact that the firmware of Piksi is still under development. Next, a discussion of the choice of antennas is given, before a set of experiments investigating the accuracy and robustness of Piksi is presented.

Base and Rover

For the remainder of this chapter, the distinction between *base* and *rover* is made when talking about multiple Piksi modules. A *base* or *base station* is in this context a stationary receiver, providing observations for other Piksi modules. A *rover* is then a mobile receiver, e.g. the one on a multirotor, that uses said observations from a base station to calculate the baseline between them.

3.3.1 Issues with stability

Piksi was crowdfunded on Kickstarter in September 2013 (Kickstarter 2014). While the hardware started shipping in the spring of 2014, the firmware and software is still in a beta stage, and under constant development. There has been a prevalence of stability issues, errors and other bugs, which is to be expected when software is still in its beta stage.

A long awaited version 1.0 firmware update, promising to fix most problems, has repeatedly been postponed since work towards this thesis started in August 2014 (at which time the firmware was at version 0.9). Instead, smaller partial updates have been released along the way, addressing some of the issues, the current version now being 0.16.

The issues with Piksi have been under heavy discussion on their community forum (SwiftNAV 2015a), and many were experienced during the work with this thesis. In the interest of accounting for the reliability of Piksi, the most prevalent issues are listed below, together with a small discussion of how they affected the experiments.

Temporary freeze

The stream of navigation data from the Piksi rover or base would sometimes freeze temporarily for $1-2$ minutes, while satellite tracking would in fact not freeze. When unfreezing, the navigational data from Piksi was often better than when the freeze started.

This problem was the most prevalent during early experiments, causing stagnation in the recording of navigation data. However, being aware of the issue meant that such a freeze didn't necessarily ruin tests completely, as the Piksi module would always unfreeze after a short while. The problem was fixed in the 0.13 firmware.

CPU overload

If the solution frequency was pushed above the stated 20 Hz by the developers, Piksi would indeed max out on its CPU usage and not manage to deliver solutions faster. Furthermore, the overload of its CPU would seem to prevent it from processing any

incoming messages, as any subsequent command was ignored, forcing a reset to default values to solve the issue.

The overload would sometimes even occur at frequencies at or right below 20 Hz. As a consequence, the default value of 10 Hz was mostly used throughout this thesis.

System crash

Cyclic redundancy check errors, stack overflow and other issues would sometimes cause Piksi to crash and only a reboot solved the problem.

Many of the experiments had to be repeated because of such errors, though their prevalence has changed considerably between firmware updates, as some were fixed and others appeared. Users still report them occurring sporadically in the newest firmware.

Undetected cycle slips

A cycle slip happens when the tracking of the carrier phase in (3.2) slips an integer number of cycles, often caused by a temporary signal loss. If undetected, it can give an offset in the baseline proportional to the signal wavelength (e.g. 19 cm for Piksi and GPS L1)(Takasu and Yasuda 2004). An incorrect IAR initialization can give similar offsets.

This problem would sometimes occur during tests, forcing a reset of the IAR and reacquiring a correct RTK fix. Version 0.14 and 0.15 introduced the capability to detect some cycle slips and incorrect initializations, but Piksi is still vulnerable to slips, especially when dropping and reacquiring satellites close to the limit of 5.

3.3.2 External antenna

The integrated antenna in Piksi is only a small patch antenna, prone to disturbances, and the use of an external antenna is highly recommended by the producer. Both a patch antenna from u-blox and an embedded antenna from Tallysman was used in an RTK solution in (Skulstad and Syversen 2014). They had problems with loss of satellites while maneuvering because of the antenna tilt during banking. Their solution was to attach the antenna with a gimbal to the UAV, to compensate for the banking angle. A gimbal adds complexity, not to mention extra weight and space, which makes it highly preferable to circumvent the need for it on the multirotors in this project.

Thus, when choosing which antennas to use on the multirotors, attention was given to the sensitivity to antenna tilting. Low weight and small size was also important factors. The antenna for the base station was not limited by the same constraints. A base station is on the other hand susceptible to multipath, as it is placed in fixed positions near the ground, possibly close to buildings and vegetation.

After investigating other users experiences in the Piksi community and consulting with experts in GNSSs at SINTEF, the following antennas were chosen:

GPS-701-GG from NovAtel
A high-performance pinwheel antenna, boasting excellent multipath rejection (NovAtel 2014). This was the main base station antenna, depicted in Figure 3.3a.

M1227HCT-A-SMA from Maxtena
Shown in Figure 3.3b, this is a special *helix* shaped antenna, chosen because of a low susceptibility to tilting. Its light weight of only 17 grams and not needing a ground plane also made it optimal for the system, saving both weight and space (Maxtena 2014). This antenna was the main rover (UAV) antenna.

(a) The GPS-701-GG pinwheel antenna from NovAtel. *Image courtesy of* novatel.com.

(b) M1227HCT-A-SMA, the helix antenna from Maxtena. *Image courtesy of* maxtena.com.

Figure 3.3: The base station (a) and rover (b) antennas for Piksi.

3.4 RTK Experiments

Several experiments were conduced to test the accuracy and robustness of the Piksi RTK navigation system and the chosen antennas. Mainly the helix antenna was tested, as this was the one chosen for the hexacopter and thus, will be exposed to high dynamics and disturbances.

Tests that measured the static accuracy is presented first, before an experiment that investigated the vulnerability of tilting the helix antenna given.

3.4.1 Static accuracy

Two Piksis with RTK fix (see Section 3.1.5) were kept a fixed distance from each other (constant baseline length, |BL|). This way, the static accuracy was a simple calculation of the standard deviation on the reported baseline. Given n samples the standard deviation σ for each coordinate is then calculated as (Walpole et al. 1993):

$$\sigma_j = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{j_i} - \bar{x}_j)^2} \tag{3.3}$$

where x_{j_i} is the value of sample $i \in \{1,\ldots,n\}$ for coordinate $j \in \{N,E,D\}$ and $\bar{x}_j$ is the mean value of coordinate j given by the equation:

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{j_i} \tag{3.4}$$

The static test setup is depicted in Figure 3.4, and all test were performed with the helix antenna, except for some early tests with the u-blox antenna used in (Skulstad and Syversen 2014). However, most of these early tests were corrupted by the freeze or crash errors mentioned in Section 3.3.1, and so only one was considered here. Two simple radios (identical to the ones used as the secondary telemetry link in the complete system) were used to transmit observations between the Piksis. The experiment was performed on the rooftop of the *Electro-block* B and D at NTNU, with an unobstructed view to the sky, and little or no clouds present.

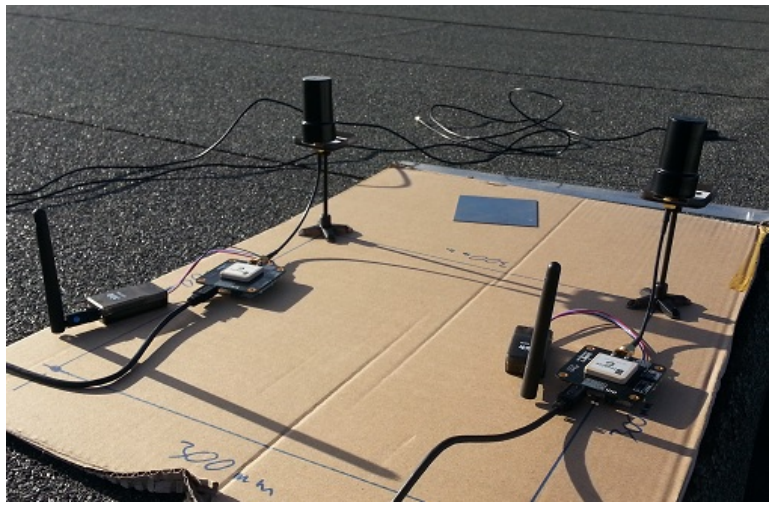


Figure 3.4: Piksi static accuracy test setup with helix antennas and radios from 3D Robotics for the communication of common observations.

Results and discussion

The results are listed in Table 3.2 with the resulting standard deviations for each separate coordinate and the total deviation of the NED position. Which firmware version in

use for each test is also listed, as Piksi has seen several improvements in performance with each update.

Table 3.2: Results from Piksi static accuracy experiment.

#	FW	Antenna	Samples	Avg. # of sats.	|BL| [m]	Standard deviation [m] N	E	D	NED
1	0.11	Helix	3177	8.89	0.3	7.13	5.66	9.42	13.10
2	0.11	Helix	769	6.71	0.3	5.24	4.34	9.27	11.50
3	0.11	u-blox	799	8.48	0.3	12.56	3.95	11.20	17.29
4	0.12	Helix	1120	10.00	0.4	3.06	2.11	5.16	6.36
5	0.12	Helix	725	10.00	0.4	2.35	2.24	6.90	7.63
6	0.12	Helix	756	10.99	2.6	2.63	1.95	5.38	6.30
AVG	-	-	*1224*	9.18	-	*5.49*	*3.38*	*7.89*	*10.36*

We see the best performance in tests 4-6, which was probably a combination of the firmware update itself and the fact that the firmware update brought improved acquisition of satellites. Like a normal GNSS (Svendsen 2001), we see that the horizontal accuracy (North and East) is noticeably better than the vertical accuracy (Down). However, an unknown discrepancy is seen in the standard deviation of the North position with the u-blox antenna, resulting in a distinctively worse total deviation than the other tests. The average total standard deviation of 10.4 mm is impressive, and disregarding the test with the u-blox antenna it would have been even better. Still, this is very close to the theoretical measurement resolution of 1.9 mm (see Table 3.1 in Section 3.1.2) and well within the centimetre-level accuracy promised by the producer (SwiftNAV 2013).

3.4.2 Antenna tilt

Since all maneuvering with a multirotor in the horizontal plane is produced by adjusting the pitch and roll, it is inevitable that a body mounted antenna will experience tilting. The main problem for an RTK system in this situation is that losing too many satellites can give a poor baseline solution, if not losing the fix altogether. Indeed, Piksi needs a minimum of 5 satellites in common with a base station to produce a baseline solution. For this reason a simple setup at the same location as the static accuracy experiment was used to test the system's susceptibility to antenna tilting:

- One Piksi as a base station, using the pinwheel antenna.
- One Piksi as a rover, using the helix antenna mounted on a servo.

Both Piksis ran the 0.12 firmware and the two antennas was kept a fixed distance of about 2.6 meters from each other. The radios from 3D Robotics was again used for the communication of observations. After acquiring good satellite tracking on both and the rover achieving RTK fix, the rover's antenna was tilted, to a certain angle orthogonal to the baseline, by the Arduino controlled servo depicted in Figure 3.5. The antenna was

kept at this angle for 30 seconds before resetting the tilt. It was then kept in an upright position for 30 seconds to allow the reacquiring of any lost satellites during tilting, after which the next tilt angle was applied. Note that Piksi had 11 satellites in the upright position.

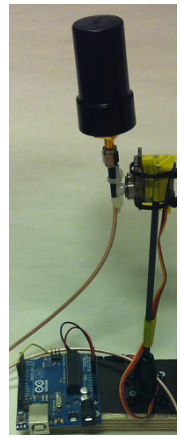


Figure 3.5: The Arduino controlled servo used in the tilt experiment.

Results and discussion

As seen in Table 3.3, the system only lost fix when the antenna was tilted -90 degrees, falling down to 4 satellites, which is below the minimum of 5 common satellites that Piksi needs. There was also a degradation of the baseline with the minimum of 5 satellites at -65 degrees, where the number of IAR hypotheses temporarily went up to 3.

Table 3.3: Results from Piksi antenna tilt experiment, with the minimum number of common satellites and the status of the RTK fix during each tilt cycle.

Tilt angle [deg]	Satellites	RTK fix
-90	4	lost
-75	6	ok
-65	5	poor
-55	8	ok
-45	9	ok
45	11	ok
55	8	ok
65	10	ok
75	8	ok
90	9	ok

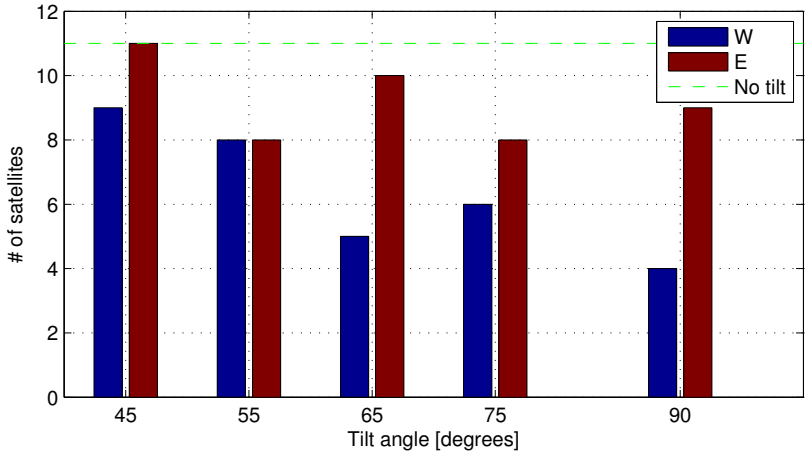


Figure 3.6: Bar graph of the Piksi antenna tilt experiment. The blue and red bars are from tilting towards the west and east, respectively, as the base station was placed north of the rover.

Interestingly, the results were much better when tilting towards the east, as is clear from the bar plot in Figure 3.6. This was probably a result of the satellite constellation at the time of the testing. The experiment shows an overall impressive resilience to tilting the helix antenna. In the normal operation of a multirotor, the induced tilt during maneuvering should stay well below the problematic angles in this experiment.

The fact that the helix antenna accepts satellites even during extensive tilting is a result of a low elevation mask. This would normally be undesirable, as the signals from satellites at low elevations must travel much further through the atmosphere, and will thus suffer more from the errors discussed in Section 3.1.1. By contrast, a "good" satellite at zenith can appear to be at a low elevation when on a tilting multirotor. However, since only the satellites a rover has in common with the base can be used in the RTK solution, the low horizon satellites are still filtered out by the base antenna with a higher elevation mask.

3.5 Interference from Telemetry

During the first test of the complete system, i.e. after the experiments presented in this chapter was performed, a new issue was encountered: The Piksi modules on the multirotors experienced a significant drop in satellite signal strengths when the main telemetry unit, PicoStation M2 (see Section 2.2.7), was turned on. Screenshots from the Piksi Console in Figure 3.7 show the dramatic drop and rise in signal strength when PicoStation was turned on and off.

(a) PicoStation turned on. **(b)** PicoStation turned off.

Figure 3.7: GPS interference from PicoStation with the original placement of the antenna masts on the *carrier plate*. The plots are GPS signal strengths from the Piksi Console, where the green and red arrows indicate when PicoStation was powered on and off, respectively.

After thorough experimentation it was found that the interference was originating from PicoStation itself and not from RF signals from its antenna. Attempts were made to shield the PicoStation in a Faraday cage made of copper (see Figure 3.8), unfortunately without any noticeable reduction of the interference.

The most effective remedy turned out to be simply moving the antenna further away from PicoStation. Since the mount for the PicoStation was already placed at the very end of the carrier plate, this meant designing a new placement for the antenna mast. Thus, the third payload level was designed and placed on top of the previous design (see Figure 2.18). The effect of powering on PicoStation with the new antenna placement is shown in Figure 3.9. As is clearly visible, the new antenna placement didn't entirely solve the problem, as the signal strength was still affected. However, it reduced the interference to an acceptable level without complicating the design too much.

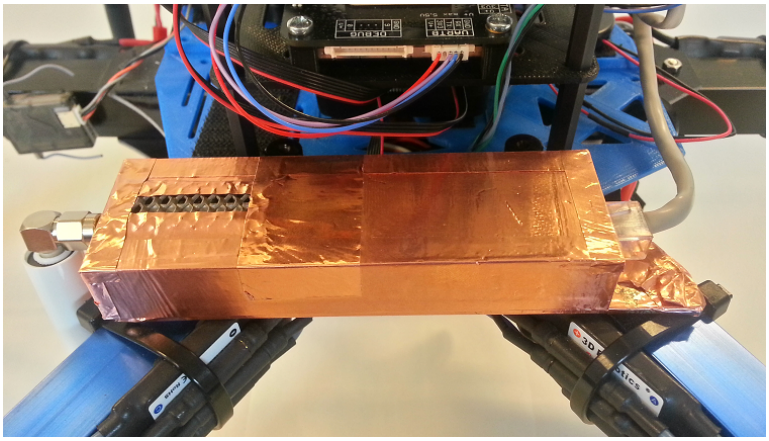


Figure 3.8: The shielded housing for the PicoStation, using copper tape.

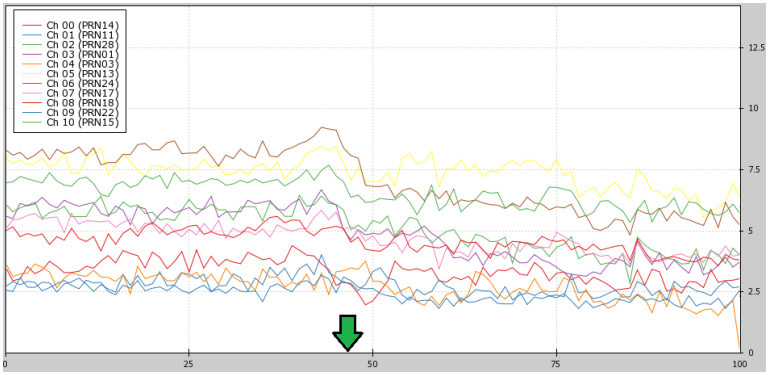


Figure 3.9: GPS interference from PicoStation with the improved placement of the antenna masts on the separate *GPS plate*. The green arrow indicates when PicoStation was powered on.

3.6 Summary

Together with the small size and light weight, the experiments in this chapter show great potential for Piksi and the chosen antennas as an RTK solution for UAVs. However, the issues encountered with stability currently limits the feasibility of tight integration in an autonomous system. As a consequence, Piksi was only partially integrated in the cooperative implementation presented in Chapter 5.

Chapter 4

Cooperative Control

This thesis has chosen to investigate the distributed approach to cooperative control mentioned in the introduction, where the agents ought to operate based on local information only, obtained through sensing or intercommunication. Consequently, the communication topology becomes essential to the design of a decentralized controller. Thus, a cooperative system can be divided into four basic elements:

- Global objective
- Agents
- Information topology
- Control algorithm for the motion of agents

Depending on the application, the global objective may differ. The main focus of this thesis is the problem of *formation control*, where the global objective is to stabilize the relative positions of agents to desired values. In (Bai, Arcak, and Wen 2011), the authors present a unifying passivity-based framework for cooperative problems. The advantages are:

- admissibility of complex and heterogeneous agent dynamics,
- design flexibility, robustness and adaptivity,
- modularity and scalability.

This chapter starts by defining the coordination problem, before a design procedure from (Bai, Arcak, and Wen 2011) is presented and then applied to the platform from Chapter 2 to achieve formation stabilization. The resulting distributed coordination law is then verified by simulation, before a short summary ends the chapter. The following sections assume that the reader is familiar with basic graph theory.[1]

[1] Note also that the labeling of reference frames has been dropped in this chapter, as all spatial vectors are represented in a common inertial frame.

4.1 Problem Statement

Consider a group of N agents where the variables to be coordinated is represented by the vector $\boldsymbol{x}_i \in \mathbb{R}^p$ and $i \in \{1,\dots,N\}$ denotes each agent. This variable can for example be the position of each agent (in which case $p = 3$). Assuming symmetric information flow, let G be the undirected graph modelling the information topology between the agents. Furthermore, assume that G is connected and has ℓ undirected links. Because of symmetric information flow, the analysis can be simplified without changing the results by considering one of the agents at each link to be at the positive end. G and the orientations of each link k can then represented by the *incidence matrix*:

$$\boldsymbol{D} = \begin{bmatrix} d_{11} & \cdots & d_{1\ell} \\ \vdots & \ddots & \vdots \\ d_{N1} & \cdots & d_{N\ell} \end{bmatrix} \in \mathbb{R}^{N\times\ell} \tag{4.1}$$

where each element is defined as (Bai, Arcak, and Wen 2011):

$$d_{ik} := \begin{cases} +1 & \text{if } k \in \mathscr{L}_i^+ \\ -1 & \text{if } k \in \mathscr{L}_i^- \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where the set $\mathscr{L}_i^+$ ($\mathscr{L}_i^-$) denotes the links for which agent i is at the positive (negative) end.

The overall goal is to implement distributed coordination laws, using only local information about neighboring agents, that guarantee the following two group behaviours:

A1) Each agent achieves a common velocity vector $\boldsymbol{v}(t) \in \mathbb{R}^p$ prescribed for the group; that is:

$$\lim_{t\to\infty} |\dot{\boldsymbol{x}}_i - \boldsymbol{v}(t)| = 0, \quad \forall i \in \{1,\dots,N\} \tag{4.3}$$

A2) If agents i and j are connected by link k, then the difference variable $\boldsymbol{z}_k$ defined as:

$$\boldsymbol{z}_k := \sum_{l=1}^{N} d_{lk}\boldsymbol{x}_l = \begin{cases} \boldsymbol{x}_i - \boldsymbol{x}_j & \text{if } k \in \mathscr{L}_i^+ \\ \boldsymbol{x}_j - \boldsymbol{x}_i & \text{if } k \in \mathscr{L}_i^- \end{cases} \tag{4.4}$$

converges to a prescribed compact set $\mathscr{A}_k \subset \mathbb{R}^p, \quad \forall k \in \{1,\dots,\ell\}$.

Note that $\boldsymbol{v}(t)$ can be seen as the common mission velocity to be achieved by the group. Furthermore, note that the target set $\mathscr{A}_k$ may change form depending on the application. For the purpose of this thesis, it can be designed such that the vehicles maintain a prescribed distance, hence keeping a formation. However, as discussed further in

Section 4.4, it is important that the target sets are feasible (Bai, Arcak, and Wen 2011). Introducing the concatenated vectors:

$$\boldsymbol{x} := \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} \in \mathbb{R}^{pN}, \quad \boldsymbol{z} := \begin{bmatrix} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_\ell \end{bmatrix} \in \mathbb{R}^{p\ell} \tag{4.5}$$

partitioning $\boldsymbol{D}$ in terms of column vectors:

$$\boldsymbol{D} = [\boldsymbol{D}_1 | \cdots | \boldsymbol{D}_\ell] \tag{4.6}$$

and using (4.4), $\boldsymbol{z}$ can be rewritten as:

$$\boldsymbol{z} = (\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)\boldsymbol{x} \tag{4.7}$$

Hence, $\boldsymbol{z}$ is restricted to $\mathscr{R}(\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)$ and thus, the target sets $\mathscr{A}_k$ are only feasible if (Bai, Arcak, and Wen 2011):

$$\{\mathscr{A}_1 \times \cdots \times \mathscr{A}_\ell\} \cap \mathscr{R}(\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p) \neq \emptyset \tag{4.8}$$

4.2 The Passivity-based Design Procedure

(Bai, Arcak, and Wen 2011) suggests a two-step design procedure with the objective to render the target sets $\mathscr{A}_k$ in (4.4) invariant and asymptotically stable, using passivity properties. This section first presents each step, before applying them to the system used in this thesis.

4.2.1 Step 1: Internal feedback

*For each agent $i \in \{1, \ldots, N\}$, design an internal feedback loop that renders its dynamics passive from an external feedback signal $\boldsymbol{u}_i$ (to be designed in **Step 2**) to the velocity error:*

$$\boldsymbol{y}_i := \dot{\boldsymbol{x}}_i - \boldsymbol{v}(t). \tag{4.9}$$

Step 1 assumes that the input-output dynamics of each agent are given by:

$$\boldsymbol{x}_i = \mathscr{H}_i^0\{\boldsymbol{\tau}_i\} \tag{4.10}$$

where $\mathscr{H}_i^0\{\boldsymbol{\tau}_i\}$ denotes the input-output dynamics of the system in agent i. Thus, we seek a feedback controller $\boldsymbol{\tau}_i$ for each agent, such that the agent dynamics may be expressed in the form:

$$\dot{\boldsymbol{x}}_i = \mathscr{H}_i\{\boldsymbol{u}_i\} + \boldsymbol{v}(t) \tag{4.11}$$

where $\mathscr{H}_i$ is a strictly passive system that is either *dynamic* or *static*. This transformation is shown in Figure 4.1

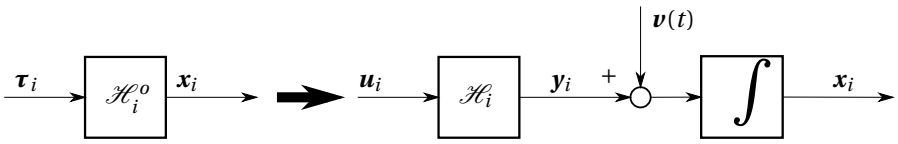


Figure 4.1: Transformation of the dynamics of agent i in Step 1, from (4.10) to (4.11).

If $\mathscr{H}_i$ is dynamic, it is assumed to have the form:

$$\mathscr{H}_i : \begin{cases} \dot{\boldsymbol{\xi}}_i = f_i(\boldsymbol{\xi}_i, \boldsymbol{u}_i) \\ \boldsymbol{y}_i = h_i(\boldsymbol{\xi}_i, \boldsymbol{u}_i) \end{cases} \tag{4.12}$$

where $\boldsymbol{y}_i$ and $\boldsymbol{\xi}_i \in \mathbb{R}^{n_i}$ is the velocity error and state variable of subsystem $\mathscr{H}_i$, respectively. Both $f_i(\cdot,\cdot)$ and $h_i(\cdot,\cdot)$ are assumed to be C^2 functions such that:

$$f_i(0, \boldsymbol{u}_i) = 0 \Rightarrow \boldsymbol{u}_i = 0 \tag{4.13}$$

and:

$$h_i(0,0) = 0 \tag{4.14}$$

As stated, $\mathscr{H}_i$ is designed to be strictly passive. By (Khalil 2002, Definition 6.3) this restricts (4.12) to have a C^1, positive definite, radially unbounded storage function, thus making the origin globally asymptotically stable.

If $\mathscr{H}_i$ is a static block, it is assumed to be of the form:

$$\boldsymbol{y}_i = h_i(\boldsymbol{u}_i) \tag{4.15}$$

where $h_i : \mathbb{R}^p \to \mathbb{R}^p$ is a locally Lipschitz function satisfying:

$$\boldsymbol{u}_i^\mathsf{T} \boldsymbol{y}_i = \boldsymbol{u}_i^\mathsf{T} h_i(\boldsymbol{u}_i) > 0, \quad \forall \boldsymbol{u}_i \neq 0 \tag{4.16}$$

thus making the system strictly passive by (Khalil 2002, Definition 6.1).

4.2.2 Step 2: External feedback

Design the external feedback signal $\boldsymbol{u}_i$ in the form:

$$\boldsymbol{u}_i = -\sum_{k=1}^{\ell} d_{ik}\Psi_k(\boldsymbol{z}_k) \tag{4.17}$$

where $\boldsymbol{z}_k$ is the difference variables defined in (4.4) and $\Psi_k(\boldsymbol{z}_k)$ is a multivariable nonlinearity designed such that the target set $\mathscr{A}_k$ is invariant and asymptotically stable.

Step 2 is basically applying a feedback-term consisting of an input $\boldsymbol{u}_i$ that is the sum of some function Ψ_k of the difference variable $\boldsymbol{z}_k$ for each link k. Note that since $d_{ik} \neq 0$ only if agent i is communicating on link k, this feedback is decentralized and implementable with local information.

By again introducing concatenated vectors:

$$\boldsymbol{y} := \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix} \in \mathbb{R}^{pN}, \quad \boldsymbol{u} := \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_N \end{bmatrix} \in \mathbb{R}^{pN}, \quad \boldsymbol{\Psi} := \begin{bmatrix} \Psi_i \\ \vdots \\ \Psi_\ell \end{bmatrix} \in \mathbb{R}^{p\ell} \tag{4.18}$$

and noting that $\boldsymbol{u}_i$ can be expressed as:

$$\boldsymbol{u}_i = -[d_{i1}\boldsymbol{I}_p|\cdots|d_{i\ell}\boldsymbol{I}_p]\boldsymbol{\Psi} \tag{4.19}$$

the external feedback $\boldsymbol{u}_i$ for all agents can be written as:

$$\boldsymbol{u} = -(D \otimes \boldsymbol{I}_p)\boldsymbol{\Psi}(\boldsymbol{z}) \tag{4.20}$$

Hence, with the applied external feedback and some rearranging, the closed-loop structure for all agents can be represented as shown in Figure 4.2. As mentioned, $\mathscr{H}_i$ is designed to be strictly passive, and pre- and post-multiplying with $\boldsymbol{D}^\intercal$ and $\boldsymbol{D}$ preserves this passivity. Therefore, the restrictions for the feedback in **Step 2** boils down to designing Ψ_k such that the passivity is preserved from $\dot{\boldsymbol{z}}$ to $\boldsymbol{\Psi}$, rendering the whole interconnected system in Figure 4.2 passive.

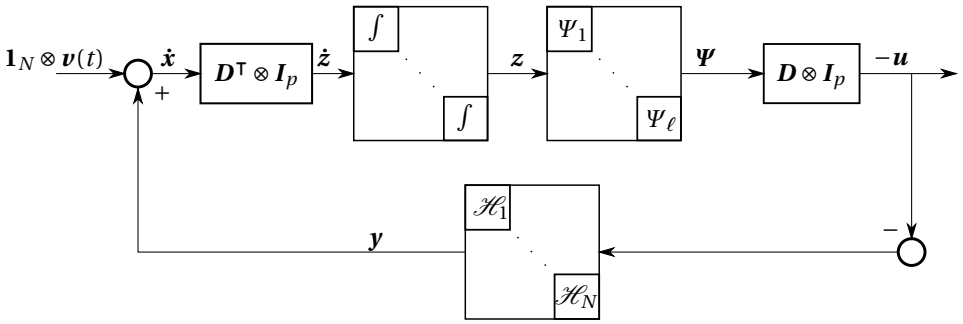


Figure 4.2: The closed-loop structure of the cooperative control of all agents after applying Step 2.

4.2.3 Internal feedback for the multirotor

As discussed in Section 2.7, the mathematical model for each multirotor can be described as double integrators:

$$m_i \ddot{\boldsymbol{x}}_i = \boldsymbol{\tau}_i \tag{4.21}$$

where $m_i \in \mathbb{R}$ and $\boldsymbol{x}_i \in \mathbb{R}^3$ is the mass and position of agent i, respectively, while $\boldsymbol{\tau}_i \in \mathbb{R}^3$ is its control input; the desired translational force.

Step 1 can then be accomplished by choosing the internal feedback as:

$$\boldsymbol{\tau}_i = -k_i(\dot{\boldsymbol{x}}_i - \boldsymbol{\nu}(t) - \boldsymbol{u}_i) + m_i \dot{\boldsymbol{\nu}}(t), \quad k_i > 0 \tag{4.22}$$

which, together with (4.9) and the change of variables:

$$\boldsymbol{\xi}_i = \dot{\boldsymbol{x}}_i - \boldsymbol{\nu}(t) \tag{4.23}$$

gives the resulting transformed system:

$$\mathscr{H}_i : \begin{cases} \frac{m_i}{k_i} \dot{\boldsymbol{\xi}}_i = -\boldsymbol{\xi}_i + \boldsymbol{u}_i \\ \boldsymbol{y}_i = \boldsymbol{\xi}_i \end{cases} \tag{4.24}$$

The system in (4.24) can be represented by its transfer function $H_i(s)$:

$$H_i(s) = \frac{k_i}{m_i s + k_i} \boldsymbol{I}_3 \tag{4.25}$$

This transfer function is strictly positive real by (Khalil 2002, Lemma 6.1) since $H_i(s-\varepsilon)$ is positive real for any $\varepsilon = k_i/m_i > 0$.[2] Thus, (4.24) is strictly passive by (Khalil 2002, Lemma 6.4).

4.2.4 External feedback for formation control

To make sure the restrictions on Ψ_k are met, (Bai, Arcak, and Wen 2011) propose designing the nonlinearities to be of the form:

$$\Psi_k(\boldsymbol{z}_k) =: \nabla P_k(\boldsymbol{z}_k) \tag{4.26}$$

where P_k is a non-negative C^2 function. There are several additional restrictions listed in (Bai, Arcak, and Wen 2011) when designing P_k for the general problem. However, for the formation problem discussed in this thesis, P_k can be chosen to have the property:

$$\boldsymbol{z}_k^\mathsf{T} \nabla P_k(\boldsymbol{z}_k) = \boldsymbol{z}_k^\mathsf{T} \Psi_k(\boldsymbol{z}_k) > 0, \quad \forall \boldsymbol{z}_k \neq 0 \tag{4.27}$$

which ensures that the necessary requirements are met and, furthermore, making the system from $\dot{\boldsymbol{z}}$ to $\boldsymbol{\Psi}$ strictly passive by (Khalil 2002, Definition 6.1).

Formation control by shifted agreement

By defining the desired difference variables[3] $\boldsymbol{z}_k^d$ and using $\Psi_k(\boldsymbol{z}_k - \boldsymbol{z}_k^d)$ instead of $\Psi_k(\boldsymbol{z}_k)$, the formation problem can be viewed as a shifted agreement problem (Bai, Arcak, and Wen 2011, Corollary 2.2).

Moreover, by choosing a quadratic potential function:

$$P_k = \delta_k 2|\boldsymbol{z}_k - \boldsymbol{z}_k^d|^2, \quad \delta_k \in \mathbb{R} > 0 \tag{4.28}$$

the resulting nonlinearity for each agent:

$$\Psi_k(\boldsymbol{z}_k) = \delta_k(\boldsymbol{z}_k - \boldsymbol{z}_k^d) \tag{4.29}$$

satisfies (4.27), thus making the complete interconnected system passive (Bai, Arcak, and Wen 2011). The constant δ_k is then a feedback gain, regulating the relative emphasis of link k.

[2] Remember that $k_i > 0$ and mass can only be positive.

[3] Note that this essentially means the value for link k, i.e. the relative position between the two agents connected by said link.

The resulting feedback controller

Finally, inserting (4.29) into (4.17) gives the resulting feedback term $\boldsymbol{u}_i$ for each agent:

$$\boldsymbol{u}_i = -\sum_{k=1}^{\ell} d_{ik}\delta_k(\boldsymbol{z}_k - \boldsymbol{z}_k^d) \tag{4.30}$$

Collision avoidance and other objectives

The above controller ensures convergence to a desired formation, however it is worth mentioning that other objectives can be achieved by incorporating additional feedback terms in (4.30). As an example, (Bai, Arcak, and Wen 2011) explains how the artificial potential field approach in robotics can be used to avoid collisions. (Bai, Arcak, and Wen 2011) goes on to show how a few simple requirements for the potential function can be used to prove stability of the resulting augmented system, although no further analysis is done as part of this thesis.

4.3 Stability and Equilibria

So far the strict passivity by design of the interconnected system of all agents in the formation problem has been shown. The passivity and thus asymptotic stability is formally proved in (Bai, Arcak, and Wen 2011). To assess whether the stability of the system implies that the objectives (4.3), (4.4) are met, the equilibrium points of the interconnected system of all agents in Figure 4.2 are investigated.

As stated in (Khalil 2002, Chapter 6.5), the closed loop state model will consist of the dynamic states from both subsystems, i.e. $(\boldsymbol{z}, \boldsymbol{\xi})$, where $\boldsymbol{\xi}$ is the concatenated vector of all state variables in (4.12) for all agents. The property of strict passivity, combined with (4.12) and (4.13), ensures that the equilibrium point $\boldsymbol{\xi} = 0$ only occurs when $\boldsymbol{u} = 0$, which from (4.20) also means that $\boldsymbol{\Psi}(\boldsymbol{z})$ is in $\mathcal{N}(\boldsymbol{D} \otimes \boldsymbol{I}_p)$. Further, from (4.14) we have that $\boldsymbol{\xi} = 0 \Rightarrow \boldsymbol{y} = 0$, which from (4.9) implies that $\dot{\boldsymbol{x}}_i = \boldsymbol{v}(t), \forall\, i \in \{1, \ldots, N\}$.

Thus, the set of equilibria is given by:

$$\mathcal{E} = \left\{(\boldsymbol{z}, \boldsymbol{\xi}) \mid \boldsymbol{\xi} = 0, (\boldsymbol{D} \otimes \boldsymbol{I}_p)\boldsymbol{\Psi}(\boldsymbol{z}) = 0 \text{ and } \boldsymbol{z} \in \mathcal{R}(\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)\right\} \tag{4.31}$$

Next, we must check that no equilibria arise outside the target sets $\mathcal{A}_k$, i.e. $\forall (\boldsymbol{z}, 0) \in \mathcal{E}, \boldsymbol{z}$ satisfies $\boldsymbol{z} \in \{\mathcal{A}_1 \times \cdots \times \mathcal{A}_\ell\}$. From (4.20) this means that:

$$\boldsymbol{u} = 0 \Longleftrightarrow \boldsymbol{z} = (\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)\boldsymbol{x} \in \{\mathcal{A}_1 \times \cdots \times \mathcal{A}_\ell\} \tag{4.32}$$

For the agreement problem ($\mathscr{A}_k = 0$), it turns out that this holds when Ψ_k is designed according to (4.27). This is because $\boldsymbol{z} \in \mathscr{R}(\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)$ and $\boldsymbol{\Psi}(\boldsymbol{z}) \in \mathscr{N}(\boldsymbol{D} \otimes \boldsymbol{I}_p)$ imply that $\boldsymbol{z}$ and $\boldsymbol{\Psi}$ are orthogonal to each other, which, in view of (4.27), is only possible if $\boldsymbol{z} = 0$. Consequently, since $\boldsymbol{\xi} = 0 \Rightarrow \boldsymbol{y} = 0$, we have shown that both (4.3) and (4.4) are met using decentralized coordination laws.

4.4 Designing Feasible Target Sets

In the formation problem, the difference variables $\boldsymbol{z}_k$ are the link vectors between the agents, as defined by the incidence matrix $\boldsymbol{D}$. Hence, the target sets $\mathscr{A}_k$ of desired difference variables $\boldsymbol{z}_k^\star$ represent the desired formation structure. As mentioned in Section 4.1, these target sets are only valid if (4.8) holds, i.e. if $\boldsymbol{z}_k^\star \in \mathscr{R}(\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)$.

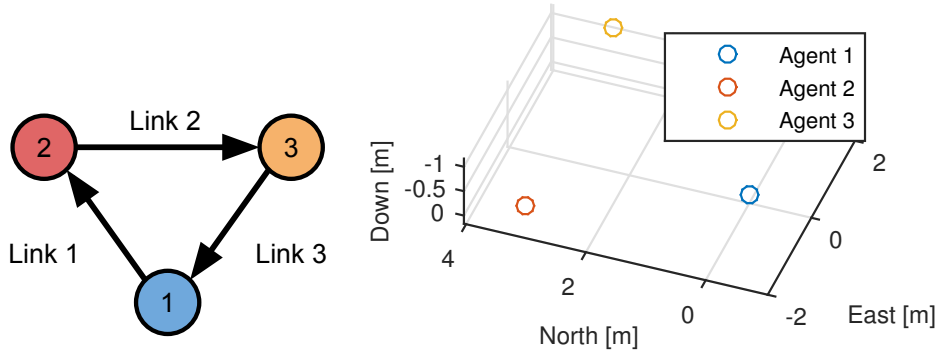
Consequently, given a desired formation $\boldsymbol{x}^F$, the target sets $\mathscr{A}_k$ can be trivially calculated as:

$$\boldsymbol{z}_k^\star := (\boldsymbol{D}^\mathsf{T} \otimes \boldsymbol{I}_p)\boldsymbol{x}^F \tag{4.33}$$

where the same concatenated structure as in (4.5) is used:

$$\boldsymbol{x}^F := \begin{bmatrix} \boldsymbol{x}_1^F \\ \vdots \\ \boldsymbol{x}_N^F \end{bmatrix} \in \mathbb{R}^{pN}, \quad \boldsymbol{z}^\star := \begin{bmatrix} \boldsymbol{z}_1^\star \\ \vdots \\ \boldsymbol{z}_\ell^\star \end{bmatrix} \in \mathbb{R}^{p\ell} \tag{4.34}$$

and $\boldsymbol{x}_i^F \in \mathbb{R}^p$ represents the desired position of agent i relative to that of the others.



(a) Communication graph. **(b)** Desired formation.

Figure 4.3: Example of communication graph and desired formation.

As an example, the desired formation of three agents in Figure 4.3b is defined as:

$$\boldsymbol{x}_1^F = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_2^F = \begin{bmatrix} 3 \\ -2 \\ -0.5 \end{bmatrix}, \quad \boldsymbol{x}_3^F = \begin{bmatrix} 3 \\ 2 \\ -1 \end{bmatrix} \tag{4.35}$$

and the communication structure in Figure 4.3a would give the incidence matrix:

$$\boldsymbol{D} = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \tag{4.36}$$

Using (4.33), the resulting target sets are thus:

$$\begin{aligned} \mathscr{A}_1 &= \left\{ \boldsymbol{z}_1 \mid \boldsymbol{z}_1 = \boldsymbol{z}_1^\star = [3, -2, -0.5]^\mathsf{T} \right\} \\ \mathscr{A}_2 &= \left\{ \boldsymbol{z}_2 \mid \boldsymbol{z}_2 = \boldsymbol{z}_2^\star = [0, 4, -0.5]^\mathsf{T} \right\} \\ \mathscr{A}_3 &= \left\{ \boldsymbol{z}_3 \mid \boldsymbol{z}_3 = \boldsymbol{z}_3^\star = [-3, -2, 1]^\mathsf{T} \right\} \end{aligned} \tag{4.37}$$

For the remainder of the thesis, the desired difference variables $\boldsymbol{z}^\star$ and the resulting target sets $\mathscr{A}_k$ are always constructed from $\boldsymbol{x}^F$ as in (4.33).

4.5 Cooperative Simulation in MATLAB

To investigate the viability of the passivity based method for formation control, the simplified translational model discussed in Section 2.7.3, together with the the resulting feedback laws from Section 4.2.3 and 4.2.4, was implemented and simulated in MATLAB (MATLAB 2015).

4.5.1 Setup

Three agents with a mass of $m_i = m = 3$ kilograms were simulated, using the same communication structure and triangular desired formation as in the example shown in Figure 4.3, thus resulting in the same incidence matrix (4.36) and target sets (4.37).

Initially, the agents were positioned along the north axis, 1 meter apart form each other:

$$\boldsymbol{x}_1^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_2^0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_3^0 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \tag{4.38}$$

The mission velocity in (4.22) was simply chosen to be a constant 0.2 m/s southward and westward, i.e. $\boldsymbol{v}(t) = \boldsymbol{v} = [-0.2, -0.2, 0]^\mathsf{T}$, while the velocity error gain and link gain was set to $k_i = 2m$ and $\delta_k = \frac{1}{2}m$ for all three agents and links. The simulation ran for 6.0 seconds.

4.5.2 Results

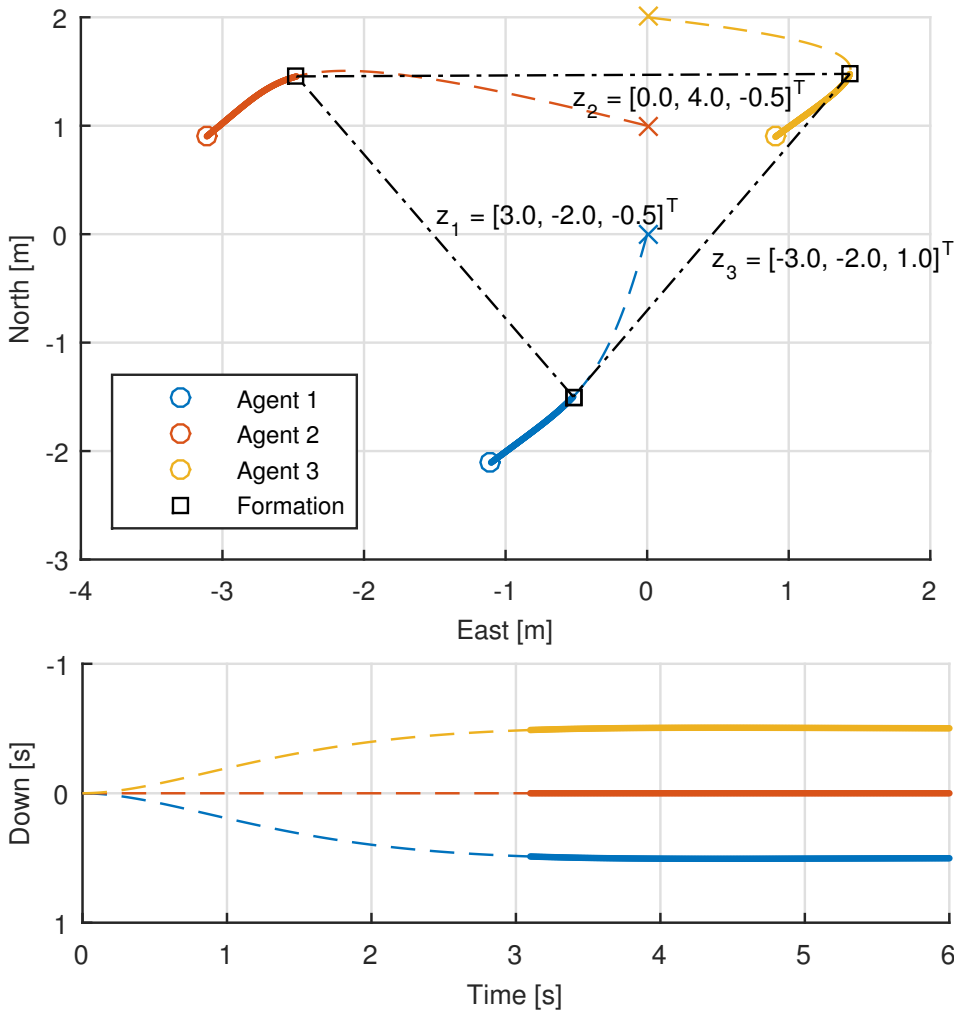


Figure 4.4: Results of the MATLAB simulation. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each agent, turning solid after reaching the desired formation drawn by the black lines. Note that the Down axis is reversed, since a negative value equals positive altitude.

The results of the simulation is shown in Figure 4.4, with the 2D position and altitude of each agent. As can be seen, the trajectories of each agent converge to a formation with the desired link lengths, i.e. the difference variables z_k, which are drawn and denoted with their respective values upon reaching said formation. A 3D plot is given in Figure 4.5, where is is easy to recognize that the agents converge to the desired formation in Figure 4.3b. Further, both figures show how the agents continue to move in formation in a south-western direction.

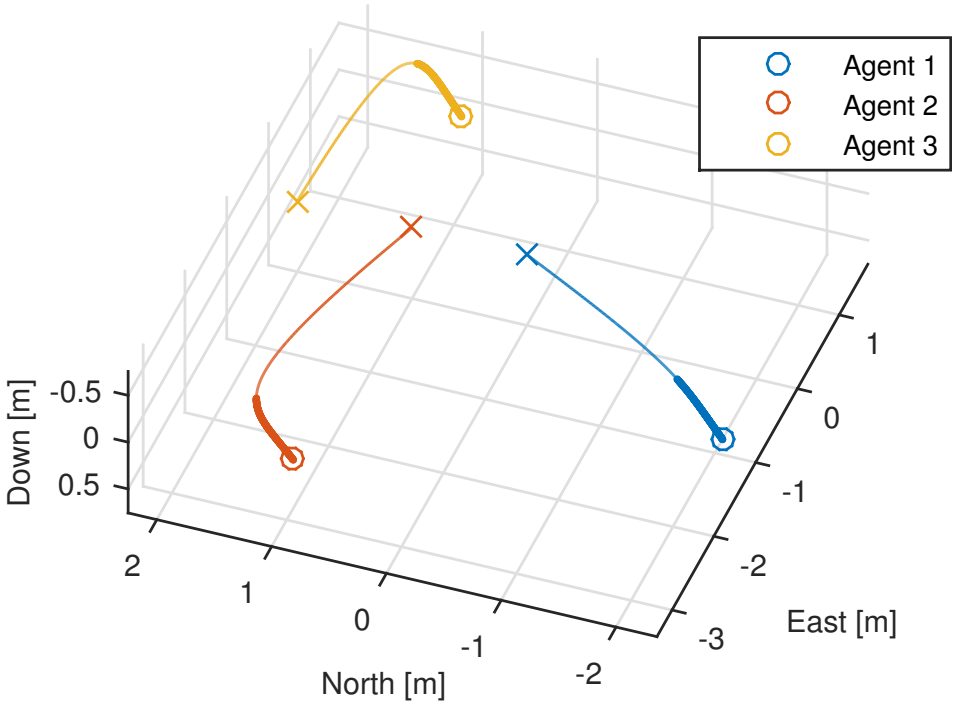


Figure 4.5: 3D plot of the MATLAB simulation. Initial positions are marked with a cross, while the colored lines show the trajectory of each agent, turning bold after reaching the desired formation.

4.5.3 Discussion

The implementation of formation control via the passivity based method was successful, as the three agents reached and kept their formation, while continuing to move in the southwestern direction as specified by the mission velocity $\boldsymbol{v}$. However, the simulation was greatly simplified as there were perfect and instant measurements for the controller, as well as instant control input and no external disturbances. The proper realization of the system, presented in the next chapter, should give noticeable delays for both measurements and control input.

4.6 Summary

The simulation in Section 4.5 proved that the passivity based method for cooperative control is a viable solution to the formation problem, as both objectives (4.3) and (4.4) were achieved. Further, by replacing the constant mission velocity, a large group of different objectives can be solved by the formation, making the method a powerful tool for cooperative control.

In the next chapter, a realization of the control laws developed here is presented, before Software-in-the-Loop (SIL) simulations and field experiments are performed in Chapter 6 and Chapter 7, respectively.

Chapter 5

Implementation

In this chapter, a realization of distributed formation control for the multirotor platform is presented, using the feedback laws developed in Chapter 4. The necessary interface for Piksi is discussed, as well as the low-level controller used in APM:Copter, before presenting the main implementation in DUNE. The chapter ends with the results from testing the delay between the autopilot and the payload computer.

Rovers and agents

For the remainder of the thesis, the term *rover* from Section 3.3 is extended to denote the different multirotors acting in the system, thereby separating them from the theoretical term *agent* used in Chapter 4.

5.1 Interface with Piksi

By adding the libswiftnav library to DUNE, the provided C bindings for SBP (see Section 2.6.1) was used to implement a DUNE interface for Piksi, i.e. a DUNE *task*. The interface had to accomplish three objectives:

- Forward observations between base and rovers through the network, eliminating the need for a dedicated radio link for Piksi.
- Extract navigational data from rovers.
- Send configuration commands to Piksi, enabling resetting and initialization of the IAR.

As mentioned in Section 3.6, a choice to avoid tight integration of Piksi with the autopilot was made based upon the issues with stability, as well as the fragility of the

signals caused by the interference from the telemetry. An interesting alternative (or supplement) to send the navigation data from Piksi through DUNE is the direct integration with the APM:Copter autopilot (indicated by the pale red connection in Figure 2.3).

Direct integration of Piksi in APM:Copter was started in January of 2014, but no new information on the progress was reported on the community forums along the main timeline of this project. APM:Copter has been able to use Piksi as a normal GPS receiver, but the support to handle the RTK data has been in a buggy beta stage. Just recently however, users have reported successful use with full integration. Unfortunately, this was too late to be pursued in this project. Consequently, the high-precision navigation from Piksi was only integrated with control in DUNE, while the lower-level APM:Copter autopilot only has data available from the Pixhawk itself.

5.2 Controller in APM:Copter

APM:Copter on the Pixhawk is a complete autopilot by it self, and supports many features mentioned in Section 2.5. However, for this thesis it was used to realize the lower-level control discussed in Section 2.7.3 to achieve translational control, and further the internal feedback law from Section 4.2.3. The latter is (4.22) implemented as a velocity setpoint controller, where $\boldsymbol{v}_i^\star := \boldsymbol{v}(t) + \boldsymbol{u}_i$ is the desired velocity and the feed-forward acceleration term $m\dot{\boldsymbol{v}}(t)$ is estimated by an internal filter.

Henceforth, the cooperative control culminates in the continuous calculation of the external feedback $\boldsymbol{u}_i$ for each rover and the mission velocity $\boldsymbol{v}(t)$, which together comprise the desired velocity vector $\boldsymbol{v}_i^\star$ to be sent to the autopilot. This higher-level control is performed in DUNE and presented in Section 5.3.

5.3 Cooperative Control in DUNE

As mentioned in Section 2.4.2, Neptus is used to send and execute mission plans to vehicles in the LSTS toolchain. Hence, a cooperative mission is started by issuing a formation plan to any of the rovers in the group. Said rover then assumes the role as *leader*, but only in the sense that it is responsible for generating and distributing the mission velocity. Further, the remaining rovers are notified and a formation flight has started.

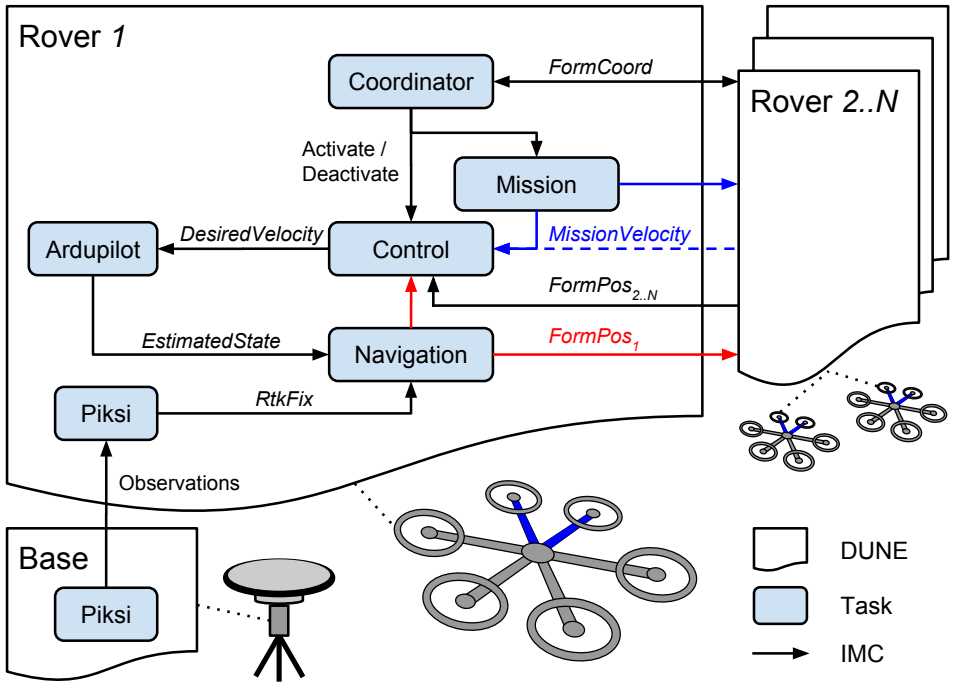


Figure 5.1: Overview of the implementation in DUNE for Rover *1* and its communication with other rovers and the base station. The individual tasks are explained in Section 5.3.1. Arrowheads indicate the direction of information flow. The *MissionVelocity* and *FormPos* messages are both consumed locally and externally, and are therefore given their own color of blue and red for clarity, respectively. Note that *MissionVelocity* comes from whichever rover is chosen as mission leader, be it the local rover (solid blue) or another one (dashed blue).

An overview of the implementation in DUNE is presented in Figure 5.1. Each rounded square is a separate task running in parallel with the others, communicating through IMC as indicated by the arrows. The functionality of each task is explained in the next section.

5.3.1 The main DUNE tasks

DUNE runs many basic tasks, e.g. tasks that handle logging, supervision, and communication with the GCS. However, only the implementation specific tasks are illustrated in Figure 5.1, and described further here.

Piksi

This task is the interface discussed in Section 5.1. *RtkFix* is an IMC message with the high precision position and velocity from the local Piksi on each rover.

Ardupilot
The Ardupilot task interfaces the on-board Pixhawk on each rover through MAVLink (see Section 2.5.2). It extracts attitude, position and velocity data which is forwarded through the *EstimatedState* IMC message. The task also sends commands to Pixhawk, such as the velocity setpoint given by *DesiredVelocity*. Alternatively, the Ardupilot task can be configured to interface the Ardupilot simulator, as is done in Chapter 6.

Navigation
This task is responsible for generating the navigational IMC message *FormPos*, containing the relative position and velocity of each rover in a common {n} frame. The message is generated from the *EstimatedState* or *RtkFix*, depending on which mode is chosen in the initialization file: *ESTATE* or *RTK*, respectively. *FormPos* is further passed on locally, as well as to all other rovers in the system.

Mission
If the local rover is the leader, this task is responsible for generating the velocity $\boldsymbol{v}(t)$ in (4.3), i.e. the common mission velocity for the formation of rovers. The generation of this velocity is dependant on whichever objective is pursued by the group, and is further discussed in Section 5.3.2.

Control
The external feedback (4.30) is implemented in this task. The incidence matrix $\boldsymbol{D}$ and link gains are loaded from a common initialization file for all rovers. Together with the mission velocity from the leader, the desired velocity for the rover is calculated and sent to the autopilot via the Ardupilot task. The Control task is also responsible for issuing an abort if it detects missing navigational data for any rover in the formation.

Coordinator
On whichever rover a formation plan is issued by the GCS, the Control and Mission tasks are activated and the rover assumes the role as a leader. The Coordinator task on said rover is then responsible for notifying the task on other participating rovers that a plan has started, so that the Control task is activated on all of them. The similar stopping of a plan, as a result of either completion, abortion or a manual stop from the GCS, is likewise forwarded to all other rovers in the formation. All this coordination is communicated through the *FormCoord* IMC message.

5.3.2 Mission velocity

As discussed in Chapter 4, the mission velocity can be tailored to solve different group objectives. For this thesis, the Mission task in Figure 5.1 responsible for generating said velocity, was implemented as a simple waypoint tracker based on the current position of the vehicle. Since this task is only activated on the rover leader, the whole system

will in effect implement a formation waypoint tracking based on the position of the leader.

To avoid jolts in the mission velocity when switching from one waypoint to the next, a reference model was implemented as part of the tracker, using the desired position $\boldsymbol{x}^\star$, i.e. the waypoint, as input. The reference model is a 2nd order velocity reference model with relative damping ratios $\boldsymbol{Z} = \text{diag}\{\zeta, \zeta, \zeta\} \in \mathbb{R}^{3\times3}$ and natural frequencies $\boldsymbol{\Omega} = \text{diag}\{\omega_n, \omega_n, \omega_n\} \in \mathbb{R}^{3\times3}$, with the resulting continuous state space representation (Fossen 2011):

$$\boldsymbol{A} = \begin{bmatrix} 0 & \boldsymbol{I}_3 \\ -\boldsymbol{\Omega}^2 & -2\boldsymbol{Z}\boldsymbol{\Omega} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ \boldsymbol{\Omega}^2 \end{bmatrix} \tag{5.1}$$

$$\dot{\boldsymbol{X}}_{ref} = \boldsymbol{A}\boldsymbol{X}_{ref} + \boldsymbol{B}\boldsymbol{x}^\star \tag{5.2}$$

where $\boldsymbol{X}_{ref} := [\boldsymbol{x}_{ref}^\mathsf{T}, \boldsymbol{v}_{ref}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^6$ is the state vector.

The velocity generated by the tracker is the sum of the velocity of the reference model and a proportional gain multiplied with the error between the current position of the vehicle and the reference model:

$$\boldsymbol{v}(t) := \boldsymbol{v}_{ref} - K_p(\boldsymbol{x} - \boldsymbol{x}_{ref}) \tag{5.3}$$

5.4 Execution Frequencies

Table 5.1 lists the execution frequencies used for the main DUNE tasks in the simulation and experiments to follow. Note that the rate listed for Piksi and Ardupilot is not actually their execution frequency, but the rate at which they receive their navigation data. The tasks themselves, as well as the tasks in Figure 5.1 that are not listed on Table 5.1, are awoken every time they receive messages that need processing, possibly inducing some operation. Mission and Control are on the other hand periodic tasks, running their main loops to produce the desired setpoints at the given frequencies.

Table 5.1: Task execution frequencies and data rates.

Task	Rate
Piksi (data)	10 Hz
Ardupilot (data)	25 Hz
Mission	30 Hz
Control	30 Hz

A few comments should be made about the rates listed in Table 5.1. The low rate for Piksi is only half of what it is able to deliver. However, as mentioned in Section 3.3.1,

pushing too close to the current limit around 20 Hz could cause instability, and thus it was kept at the default setting of 10 Hz. The somewhat low telemetry rate of 25 Hz from Pixhawk to the Ardupilot task is a result of the current limitation of outgoing messages from the Pixhawk. This is a currently hard coded limit of 50 Hz, but a bug in the execution timer caused it to deliver a rate of about 25 Hz.[1] Note however that the internal update rate in APM:Copter does in fact run at 400 Hz on the Pixhawk (DIY Drones 2015d).

5.5 Delays

To investigate the delay between APM:Copter on the Pixhawk and DUNE on the BBB, two delay tests were performed:

- **Command setpoint**: The time from a DUNE-controller dispatching a setpoint IMC message, to the MAVLink message being processed on Pixhawk.
- **Attitude feedback**: The time from a change in attitude on Pixhawk, to this change was available for controllers in DUNE.

All tests were run with a baud-rate of 921600, 50 Hz telemetry update rate, and were measured with an oscilloscope connected to GPIO-pins on both units. The results are listed in Table 5.2.

Table 5.2: Results from the delay tests between Pixhawk and BBB.

Test	Delay [ms]	
	Min	Max
Command setpoint	20	30
Attitude feedback	10	20

The total maximum delay of 50 milliseconds in the loop is small compared to the current low rates used in the system (see Table 5.1. A similar delay of about $10-50$ milliseconds was observed when sending IMC messages between rovers across the PicoStation through UDP. However, as mentioned in Section 1.2, communication delays below a certain threshold doesn't affect the consensusability of the system (Cao et al. 2013). Still, these and other delays, as well as the somewhat low telemetry and controller rates, impose restrictions on the system dynamics. For this reason, saturation in the controller input for the multirotors was set relatively low in the setup for the simulation and experiments in Section 6.2 and 7.2.

[1] This bug was fixed towards the end of the project, but the rate of 25 Hz is what was present during the simulation and experiments in Chapter 6 and Chapter 7, respectively

Chapter 6

Multirotor Simulations

A powerful verification tool during development, Software-in-the-Loop (SIL) simulations were used extensively throughout the implementation of the various features in DUNE. This chapter presents the simulator that was used, before the finished application from the previous chapter is verified in a simulated mission with three multirotors.

6.1 ArduPilot Software-in-the-Loop

The ArduPilot Software-in-the-Loop simulator, or AP-SIL, allows running APM:Copter without the hardware. The multirotor and environment is replaced by a physics simulator while the Pixhawk is virtualized, and APM:Copter is run as a native executable on the computer (DIY Drones 2015c). Written in Python, the physics simulator uses similar dynamics as those presented in Section 2.7.[1] A lightweight, terminal based GCS called MAVProxy (Tridgell 2015) is used to interface APM settings and perform the simulated arming of the multirotor.

Hence, the implementation presented in the previous chapter could be tested by simply configuring the Ardupilot task to connect with APM:Copter running on the simulator, rather than on the actual Pixhawk. Unfortunately, there was no simulator for Piksi, meaning that only navigational data from the virtual Pixhawk was available during simulations, i.e. the Navigation task in Figure 5.1 had to be placed in *ESTATE* mode. Figure 6.1 illustrates how DUNE communicates with APM:Copter when the latter runs as part of AP-SIL versus running on hardware (Pixhawk).

[1] Likewise, the physics simulator also lacks the incorporation of environmental forces (wind), which will be a challenge faced in the later experiments

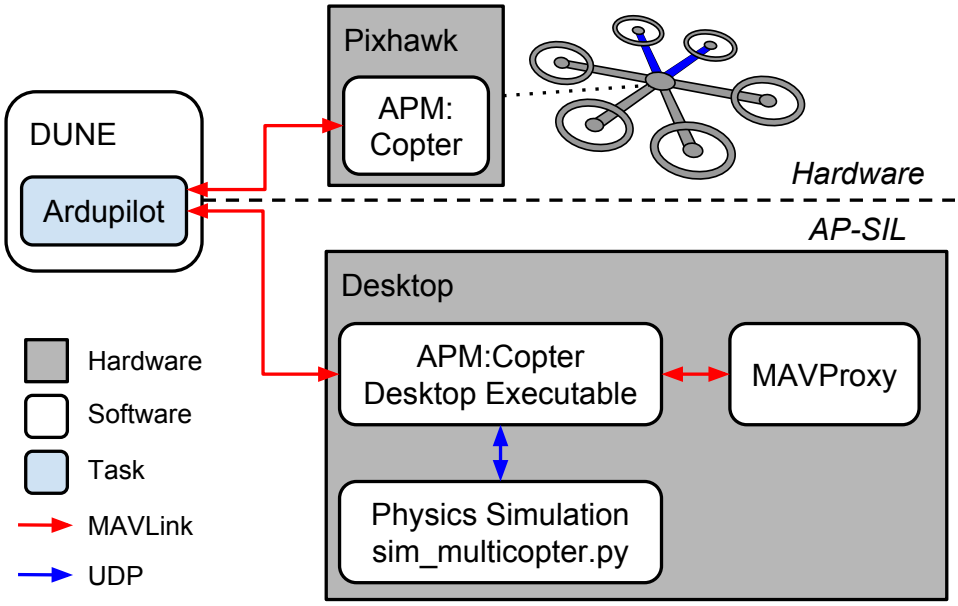


Figure 6.1: The AP-SIL architecture vs. the normal hardware setup. Note that during simulations, DUNE can either run on hardware (GLUED on BBB) or on the desktop together with AP-SIL.

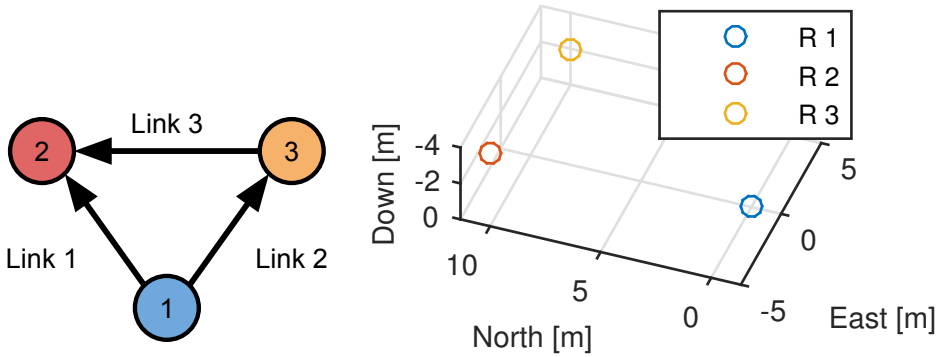
6.2 Setup

Like the MATLAB simulation in Section 4.5, three agents, in this case multirotors or *rovers*, were simulated. Table 6.1 lists the different parameters used for the reference model and controllers.[2] Note the choice of relatively low velocity saturations because of the delays and low controller rates discussed in Section 5.3.1 and 5.5.

Table 6.1: Parameters used during the AP-SIL simulation.

Parameter	Value	Unit
ζ	1	-
ω_n	0.1	Hz
K_p	0.05	-
$v_{ref,sat}$	1.5	m/s
v_{sat}	2	m/s
$v^\star_{sat}$	3	m/s
δ	0.05	-

[2] Note that δ without a subscript here indicates an equal link gain for all links, i.e. $\delta_k = \delta, \quad \forall k \in \{1, \ldots, L\}$.



(a) Communication graph. **(b)** Desired formation.

Figure 6.2: Communication graph and desired formation in the AP-SIL simulation.

The desired formation for the rovers was set to:

$$\boldsymbol{x}_1^F = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_2^F = \begin{bmatrix} 10 \\ -5 \\ -4 \end{bmatrix}, \quad \boldsymbol{x}_3^F = \begin{bmatrix} 10 \\ 5 \\ -2 \end{bmatrix} \tag{6.1}$$

which gives the tilted triangular shape illustrated in Figure 6.2b. Further, the communication structure in Figure 6.2a was used, which gives the resulting incidence matrix:

$$\boldsymbol{D} = \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \tag{6.2}$$

Thus, from (4.33), the desired link lengths are:

$$\boldsymbol{z}_1^\star = \begin{bmatrix} 10 \\ -5 \\ -4 \end{bmatrix}, \quad \boldsymbol{z}_2^\star = \begin{bmatrix} 10 \\ 5 \\ -2 \end{bmatrix}, \quad \boldsymbol{z}_3^\star = \begin{bmatrix} 0 \\ -10 \\ -2 \end{bmatrix} \tag{6.3}$$

A formation plan with two waypoints was sent to Rover 1, thus making it the leader that the group tracks said waypoints relative to. Coordinates for the waypoints are not given explicitly, as they should be apparent in the plots below. Likewise, the initial positions for the rovers are not listed, but a remark can be made that they were considered so as to avoid any concern with collisions when converging towards their formation.

6.3 Results

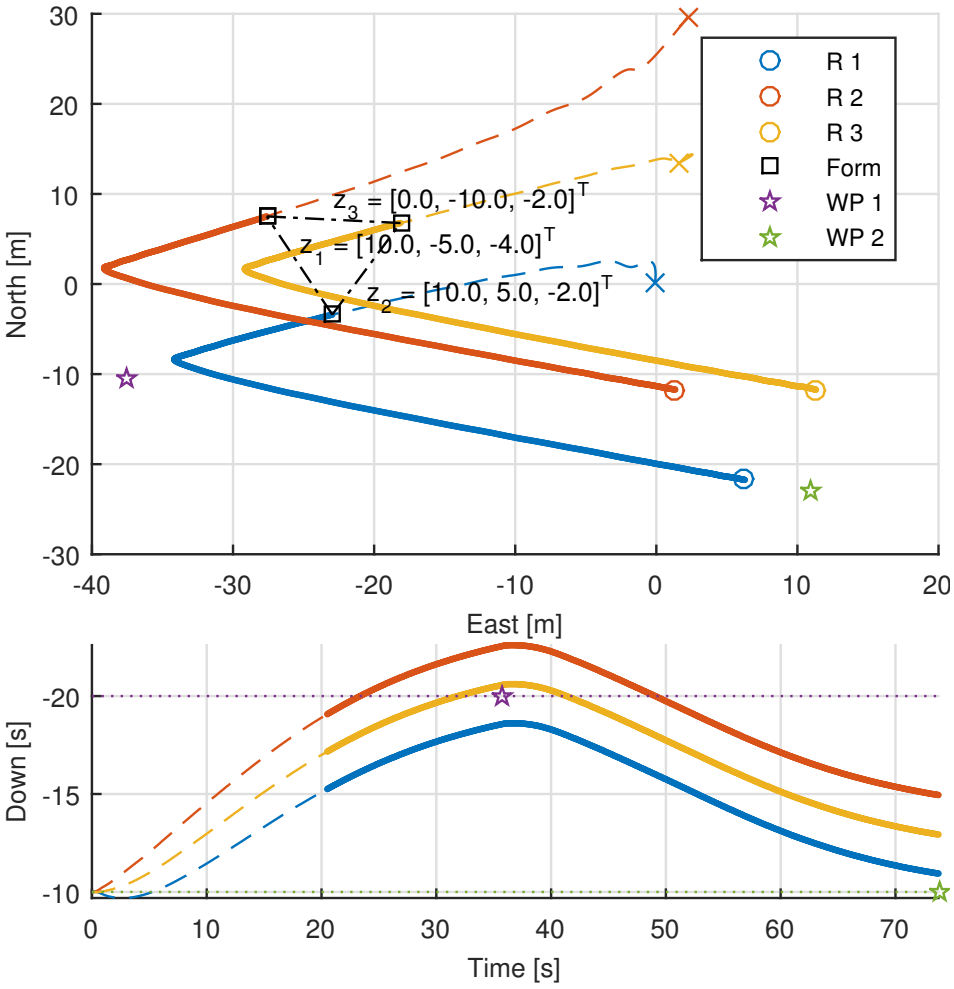


Figure 6.3: Results of the AP-SIL simulation. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each rover (R $1-3$), turning solid after reaching the desired formation drawn by the black lines. Waypoints are marked by colored stars and their height by the dotted lines of the same color. Note that the Down axis is reversed, since a negative value equals positive altitude.

The results of the simulation are given in Figure 6.3. Clearly, the rovers reach the desired link lengths in (6.3) as they fly towards the first waypoint. Further, upon arrival they smoothly turn towards the next waypoint, still adhering to their formation. The 3D plot in Figure 6.4 portray the same story, where again the desired formation from Figure 6.2b

is palpable. Figure 6.5 shows how the absolute error for each link, i.e. $|z_k - z_k^\star|$, converges to zero. In addition, the individual rover velocities can be found in Appendix A.

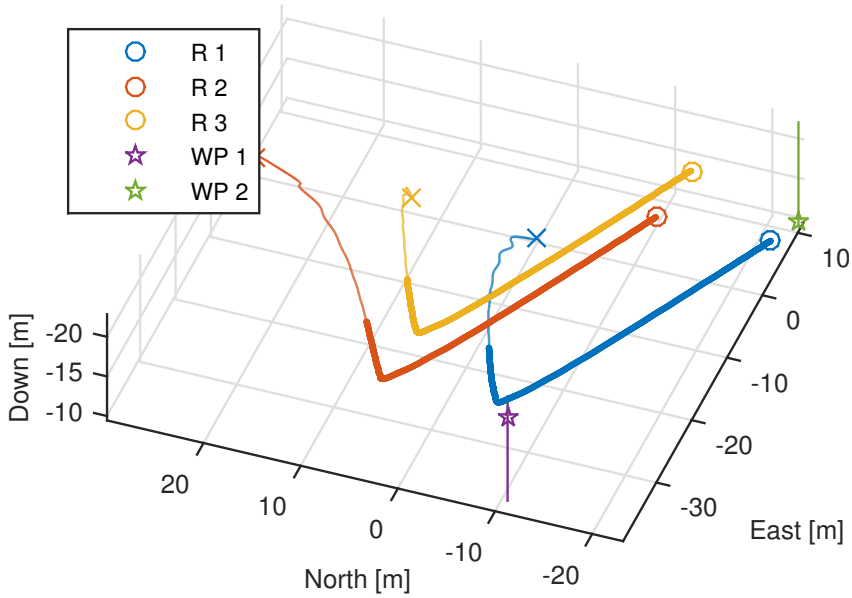


Figure 6.4: 3D plot of the AP-SIL simulation. Initial positions are marked with a cross, while the colored lines show the trajectory of each rover, turning bold after reaching the desired formation. Waypoints are marked by colored stars, pinned by vertical bars for visibility.

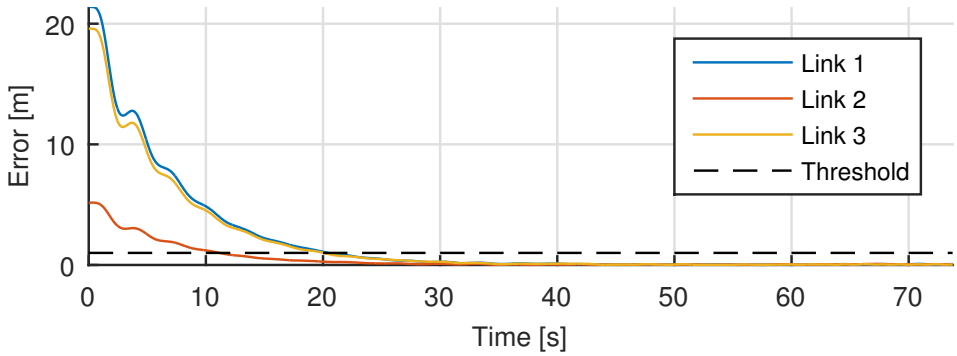


Figure 6.5: Link errors in the AP-SIL simulation. An arbitrary threshold, here set at 1 meter, decides when the rovers are in formation, and is only used during plotting.

6.4 Discussion

The simulation demonstrates that the realization of formation control from Chapter 5 works, as the rovers successfully reached their desired formation, as well as tracking the given waypoints with respect to the leader. However, like the simulation in Section 4.5, there were still no environmental forces other than the force of gravity. By contrast, delays and imperfect measurements were present, even though these are probably somewhat smaller and better than those faced in the actual experiments in the next chapter.

Chapter 7

Multirotor Experiments

The importance of field experiments is one of the cornerstones in this thesis, proving the viability of the cooperative multirotor control system. This chapter starts with a short description of the locale, followed by an overview of the different experiments categorized on two different sets. Thereafter, each different set of experiments is presented individually: The specific setup is outlined, followed by a presentation of the results.[1] Finally, a collective discussion of the results from both sets is given, where the outcome is compared to the simulations in the previous chapter.

7.1 Agdenes Airfield

Figure 7.1: Aerial photo of Agdenes Airfield. *Image courtesy of* norskeflyplasser.no.

[1] Similar to the simulations in Chapter 6, the individual rover velocities can be found in Appendix A.

The UAV-Lab of AMOS operates their UAVs from Agdenes Airfield (openAIP 2015) shown in Figure 7.1. The airstrip is located at Breivika, a 90 km drive northwest of Trondheim, and is where all field experiments were conducted.[2]

7.2 Overview

A large number of field experiments were performed, out of which five are presented below. In all experiments, two rovers were used, as only two experienced pilots were available for the project.[3] Take-off and landing was always performed manually by the pilots, who also were responsible for reclaiming manual control if anything went wrong during the execution of the autonomous missions. Further, one person was always operating the GCS, in addition to a fourth member responsible for logging and visual observation of both rovers. A photo of the setup can be seen in Figure 7.2.

Figure 7.2: The experimental setup at Agdenes Airfield. Co-supervisor Kristian Klausen can be seen performing a magnetometer calibration on one of the multirotors, while the author inspects the base station antenna. Collaborative MSc student Recep Cetin is seen manning the GCS next to the base station.

Table 7.1 lists the different parameters used for the reference model and controllers in all the experiments. Note that compared to the simulation in the previous chapter, the maximum desired speed was lowered by 1 m/s, and the maximum mission and reference speed by 0.5 m/s, while the link gain was doubled. The velocities were lowered

[2] In fact five expeditions to Agdenes Airfield were carried out as part of this project and (Cetin 2015), spanning a total of six days.

[3] A single pilot could be responsible for more than one rover, but that would pose a great risk if anything went wrong with more than one multirotor at once.

to account for the probability of larger system delays in field experiments than in the SIL simulations, while the link gain was increased because there is now only one link active, as opposed to three in the simulation.[4]

Table 7.1: Parameters used during field experiments.

Parameter	Value	Unit
ζ	1	-
ω_n	0.1	Hz
K_p	0.05	-
$v_{ref,sat}$	1	m/s
v_{sat}	1.5	m/s
$v^{\star}_{sat}$	2	m/s
δ	0.1	-

Even with the extra GPS plate, the interference discussed in Section 3.5 made acquiring an RTK fix on Piksi very fragile. For this reason, many of the experiments were run with navigation from the Pixhawk instead, i.e. the Navigation task in Figure 5.1 was put in *ESTATE* mode, like the simulation in the previous chapter. The experiments are thus divided in two separate sets, with the differing details summarized in Table 7.2.

Table 7.2: Overview of the different field experiments.

Experiment	Navigation	Control	Waypoints
1-1	ESTATE	3D	0
1-2	ESTATE	3D	1
1-3	ESTATE	3D	2
2-1	RTK	2D	0
2-2	RTK	2D	1

In each set, an experiment without waypoints is first presented to show convergence of the desired formation. 2D control means that velocity setpoints are only given to the autopilot in the horizontal plane, while the pilot remains in control of the vehicle height. It is also worth mentioning that waypoint tracking was only performed in the horizontal plane for all experiments, i.e. the z-component of the mission velocity was always set to zero,[5] hence it is only the formation control in Experiment 1-1 to 1-3 that span 3D. Again, the formation tracks waypoints with respect to Rover 1's position, as the formation plan was always sent to said Rover, making it the leader.

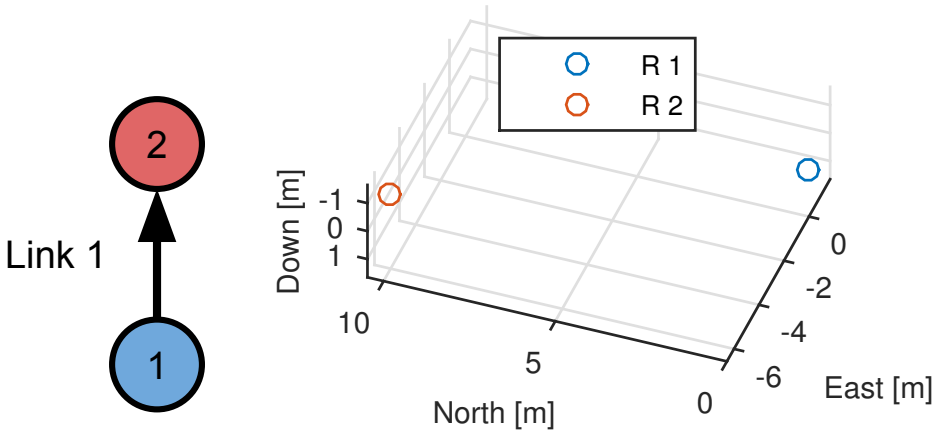
[4] Remember that links errors are added cumulatively in the external feedback (4.30). Hence, more links add up to more feedback, and their gains should be adjusted accordingly.

[5] This simplification of the mission velocity was a result of an ambiguity problem with the path controller class inherited by the Mission task in Figure 5.1, where the reported altitude of waypoints would change or grow.

7.3 Experiment Set 1 - Pixhawk Navigation

Because of the fragile RTK fix on Piksi, most experiments were performed using only navigation from the Pixhawk. Three of these are presented next. It should be mentioned that there was considerable amounts of wind during this set of experiments.

7.3.1 Setup



(a) Communication graph. **(b)** Desired formation.

Figure 7.3: The communication graph and desired formation during Experiment Set 1.

Figure 7.3 shows the communication structure and desired formation used during the first set of experiments, where the straight line formation in Figure 7.3b is defined as:

$$\boldsymbol{x}_1^F = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_2^F = \begin{bmatrix} 10.47 \\ -4.87 \\ 0 \end{bmatrix} \tag{7.1}$$

Moreover, the structure in Figure 6.2a results in the simple incidence matrix:

$$\boldsymbol{D} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{7.2}$$

which, according to (4.33), leads to the trivial desired link length:

$$\boldsymbol{z}_1^\star = \begin{bmatrix} 10.47 \\ -4.87 \\ 0 \end{bmatrix} \tag{7.3}$$

7.3.2 Results

The results in Figure 7.5, 7.8 and 7.11 show that the rovers reach their formation in all the experiments in the set, both horizontally and in altitude. Indeed, the desired north-west line formation in Figure 7.3b is evident from the 3D plots in Figure 7.6, 7.9 and 7.12. The link error in Experiment 1-2 and 1-3 (Figure 7.10 and 7.13) converge somewhat better towards zero than in Experiment 1-1 (Figure 7.7).

Figure 7.4: Photo of Rover 1 and 2 at the start of Experiment 1-3, marked by their respective colors of blue and red. One of the pilots is also present, as well as the base station and the GCS with operator.

With no waypoint in Experiment 1-1, we see in Figure 7.6 that the achieved formation starts slightly drifting. There is noticeable oscillation in altitude in all three experiments, most evident in Figure 7.8 of Experiment 1-2. Figure 7.4 shows a picture of the rovers during the execution if this second experiment in the set.

Experiment 1-1: Formation only

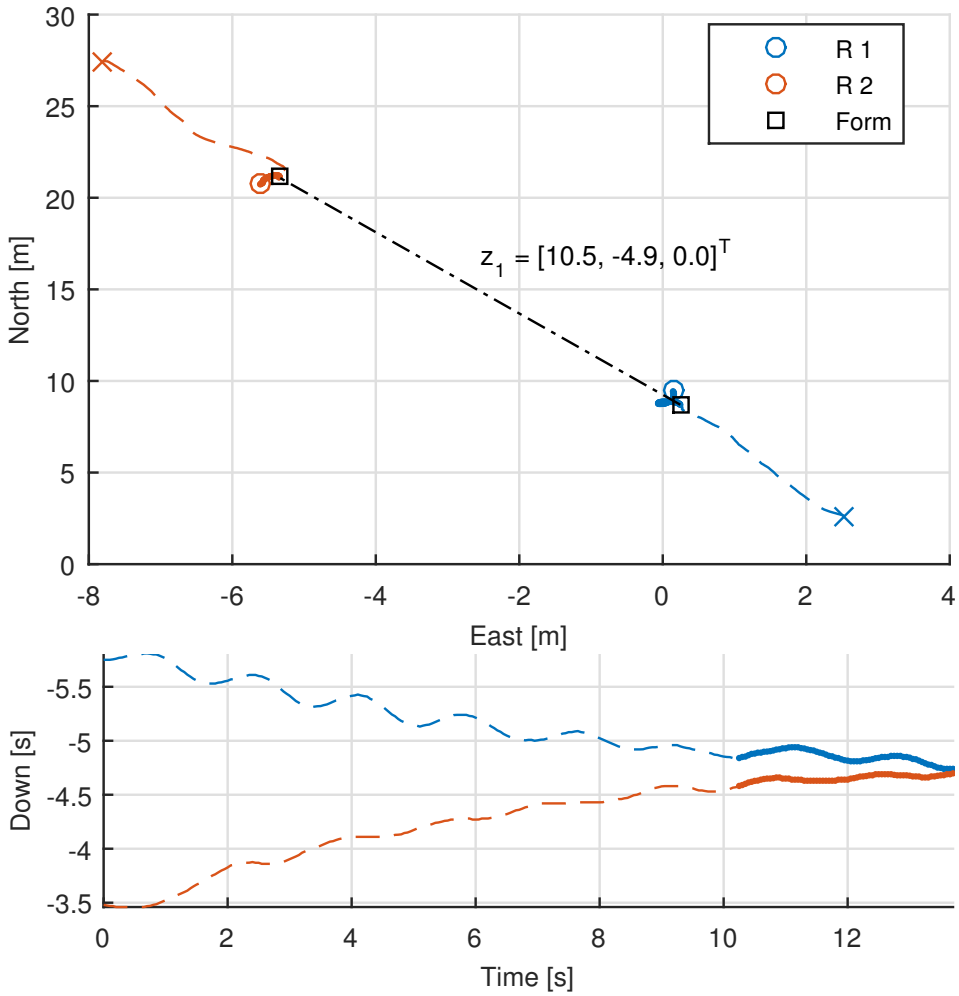


Figure 7.5: Results from Experiment 1-1. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each rover, turning solid after reaching the desired formation drawn by the black lines. Note that the Down axis is reversed, since a negative value equals positive altitude.

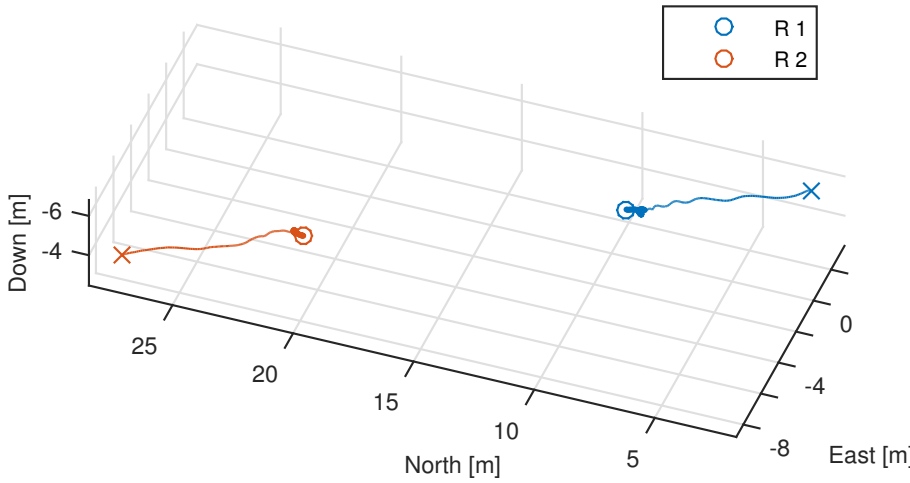


Figure 7.6: 3D plot of Experiment 1-1. Initial positions are marked with a cross, while the colored lines show the trajectory of each rover, turning bold after reaching the desired formation.

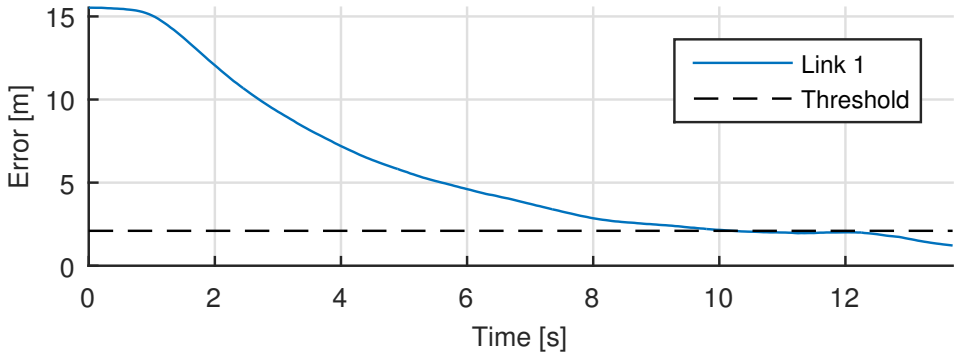


Figure 7.7: Link error during Experiment 1-1. An arbitrary threshold, here set at 2 meters, decides when the rovers are in formation, and is only used during plotting.

Experiment 1-2: 1 waypoint

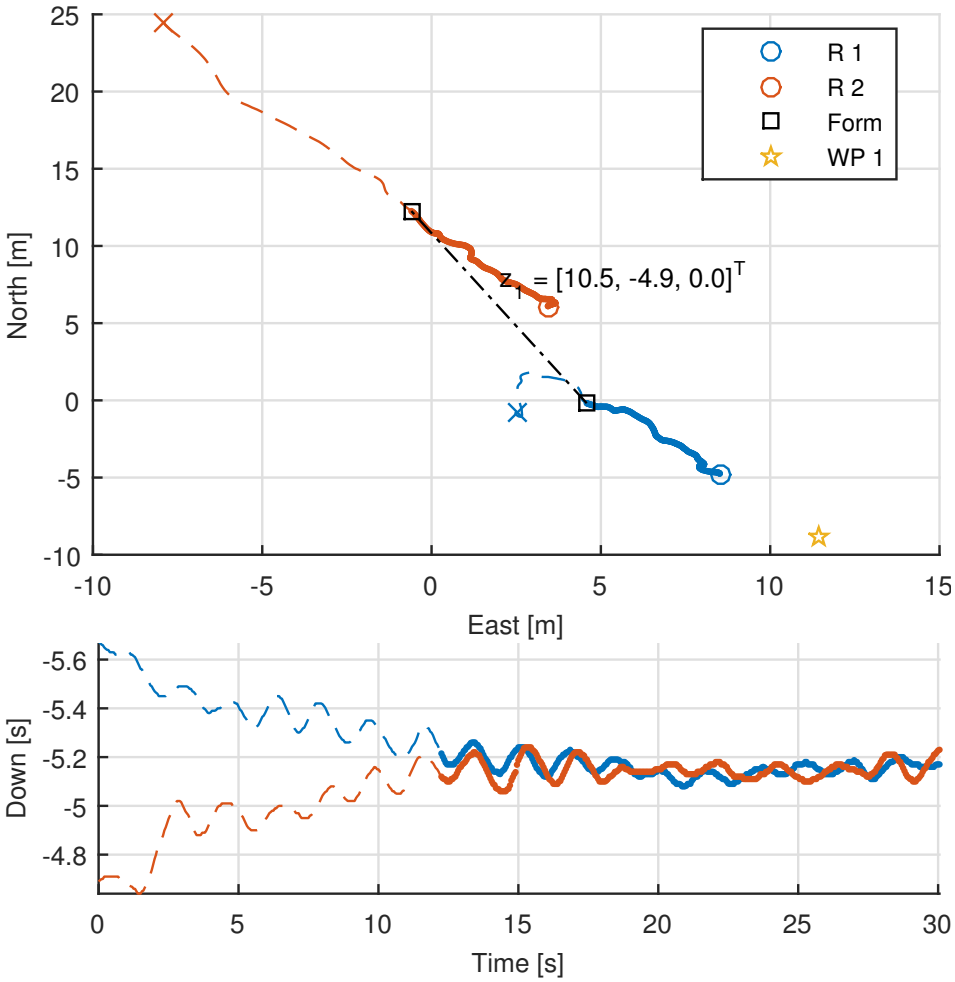


Figure 7.8: Results from Experiment 1-2. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each rover, turning solid after reaching the desired formation drawn by the black line. Note that the Down axis is reversed, since a negative value equals positive altitude.

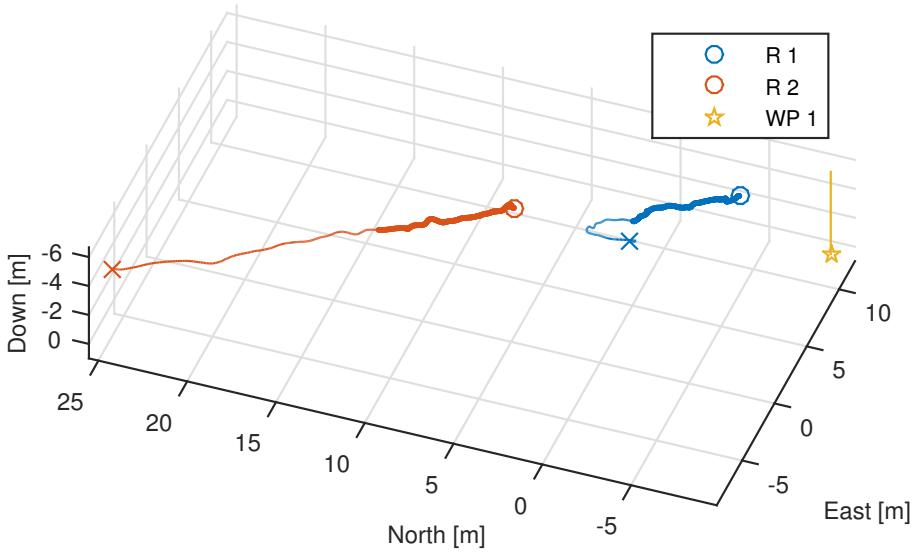


Figure 7.9: 3D plot of Experiment 1-2. Initial positions are marked with a cross, while the colored lines show the trajectory of each rover, turning bold after reaching the desired formation. The waypoint is marked by a colored star, pinned by a vertical bar for visibility.

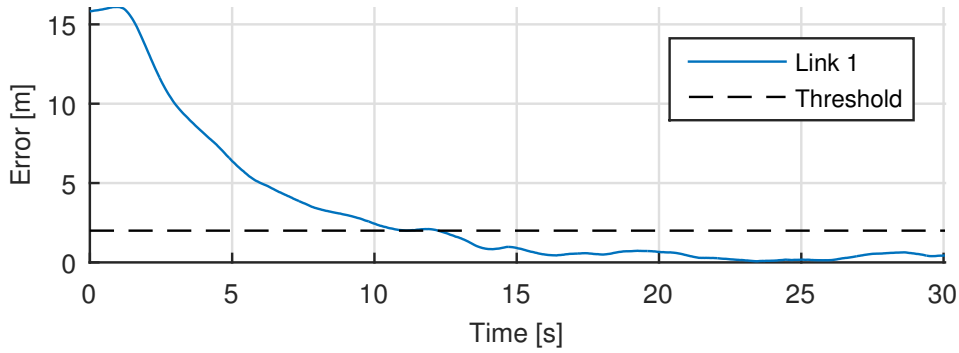


Figure 7.10: Link error during Experiment 1-2. An arbitrary threshold, here set at 2 meters, decides when the rovers are in formation, and is only used during plotting.

Experiment 1-3: 2 waypoints

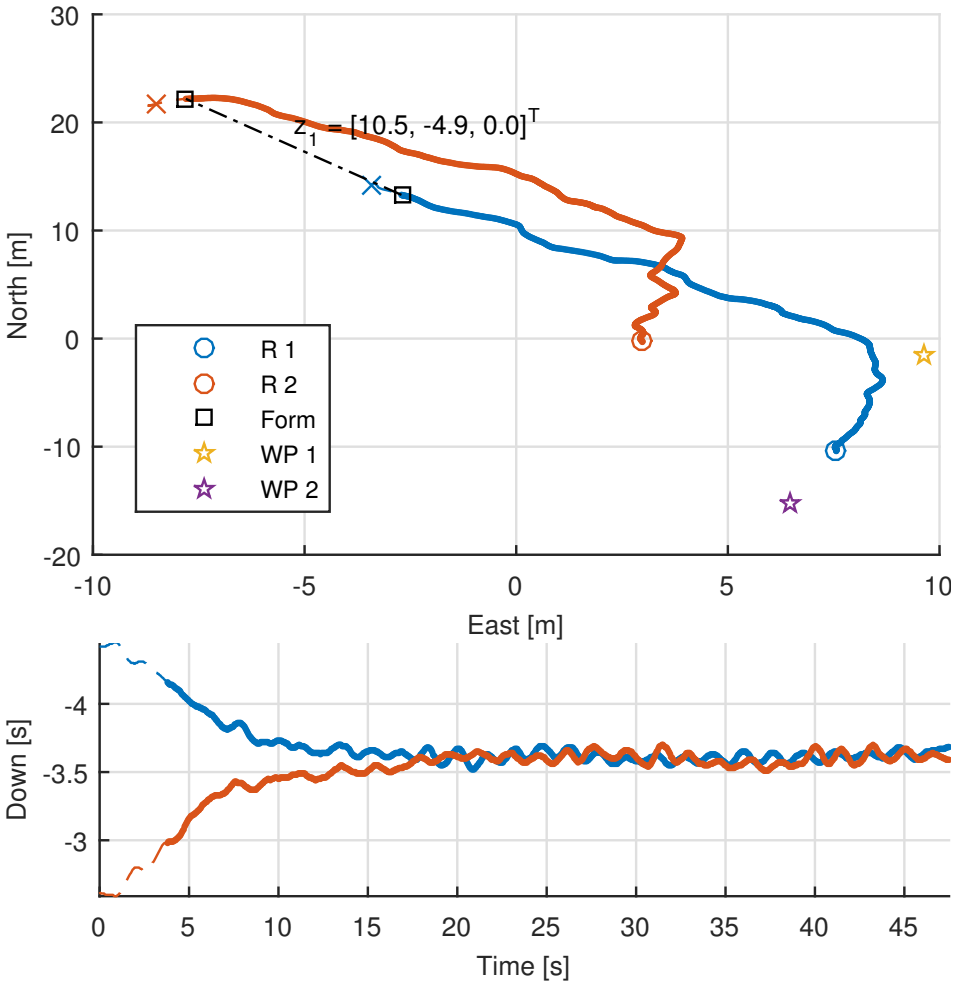


Figure 7.11: Results from Experiment 1-3. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each rover, turning solid after reaching the desired formation drawn by the black line. Note that the Down axis is reversed, since a negative value equals positive altitude.

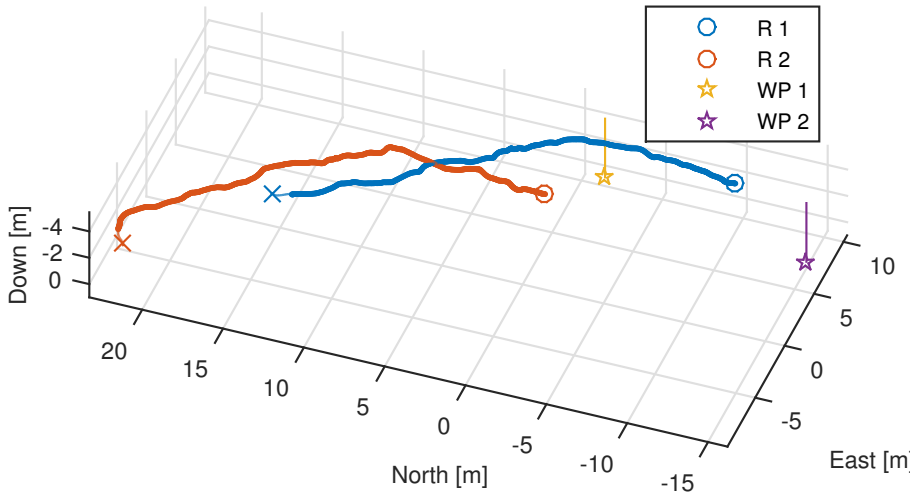


Figure 7.12: 3D plot of Experiment 1-3. Initial positions are marked with a cross, while the colored lines show the trajectory of each rover, turning bold after reaching the desired formation. Waypoints are marked by colored stars, pinned by vertical bars for visibility.

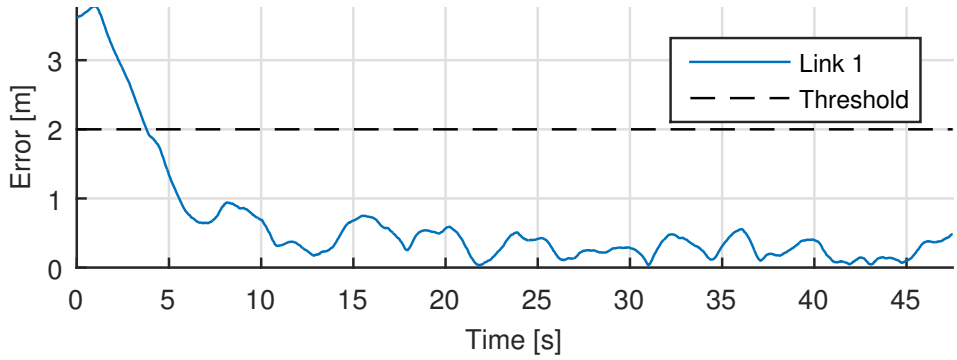
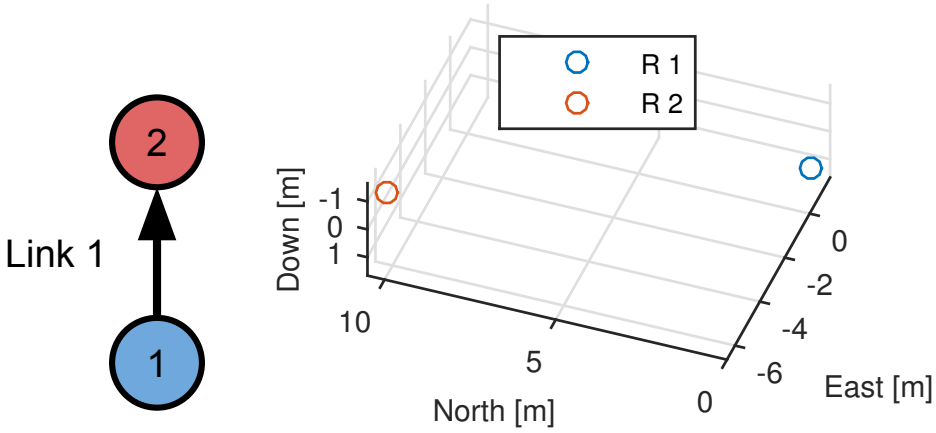


Figure 7.13: Link error during Experiment 1-3. An arbitrary threshold, here set at 2 meters, decides when the rovers are in formation, and is only used during plotting.

7.4 Experiment Set 2 - Piksi Navigation

Only a few experiments with RTK navigation from Piksi was successful, as a temporary degradation of the fix would cause a mission abort with the current implementation. Nonetheless, after a few attempts, a consistent fix was kept throughout some missions, out of which two are presented next.

7.4.1 Setup



(a) Communication graph. **(b)** Desired formation.

Figure 7.14: The communication graph and desired formation during Experiment Set 2.

As shown in Figure 7.14a, the second set of experiments used the same communication structure as in the first one, but the desired formation was changed, ever so slightly:

$$\boldsymbol{x}_1^F = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_2^F = \begin{bmatrix} 10.5 \\ -5 \\ 0 \end{bmatrix} \tag{7.4}$$

resulting in the near identical illustration in Figure 7.14b. Consequently, the incidence matrix is identical to (7.2):

$$\boldsymbol{D} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{7.5}$$

while the resulting desired link length is also changed slightly:

$$z_1^\star = \begin{bmatrix} 10.5 \\ -5 \\ 0 \end{bmatrix} \tag{7.6}$$

7.4.2 Results

Again, the results in Figure 7.16 and 7.19 show that the rovers reach their objective. However, the desired formation in Figure 7.14b is not that discernible in the 3D plots of Figure 7.17 and 7.20, as autonomous control was only performed in the horizontal plane (see Table 7.2). Indeed, the difference in altitude of the rovers does not converge to zero as in the previous set of experiments, although the link error in Figure 7.18 and 7.21 does (since it was accordingly defined for the horizontal plane only in these experiments).

Figure 7.15: Photo of Rover 1 and 2 during Experiment 2-2, marked by their respective colors of blue and red. One of the pilots, as well as the GCS and operators, are visible in the lower left corner.

The formation of rovers in Experiment 2-1 starts drifting together substantially in Figure 7.16, as only their relative formation is controlled in the absence of any waypoints. Note also that the plan in Experiment 2-2 was aborted prematurely because of a low-voltage alarm, which is why the final position of Rover 1 in Figure 7.19 is rather far away from the waypoint. A photo of the two rovers during this experiment can be seen in Figure 7.15.

For comparison, the positioning data from Pixhawk has also been plotted as an extra, green trajectory for each rover in Figure 7.16 and 7.19, and a deviation of about a meter can be seen along both the Down and North axis for Rover 1 in Figure 7.19. Likewise, the velocities from Pixhawk has also been added to the velocity plots in Appendix A.

Experiment 2-1: Formation only

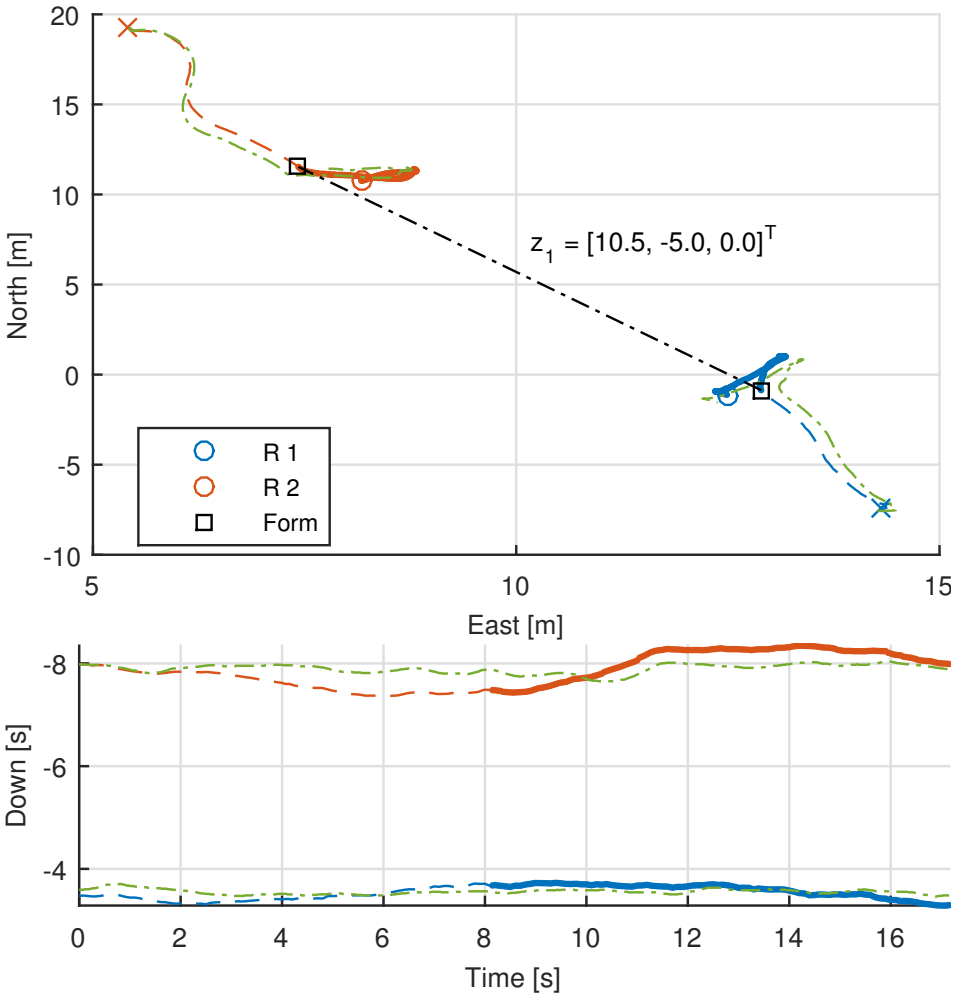


Figure 7.16: Results from Experiment 2-1. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each rover, turning solid after reaching the desired formation drawn by the black line. The green lines shadowing the rover trajectories are their respective positions from the Pixhawk. Note that the Down axis is reversed, since a negative value equals positive altitude.

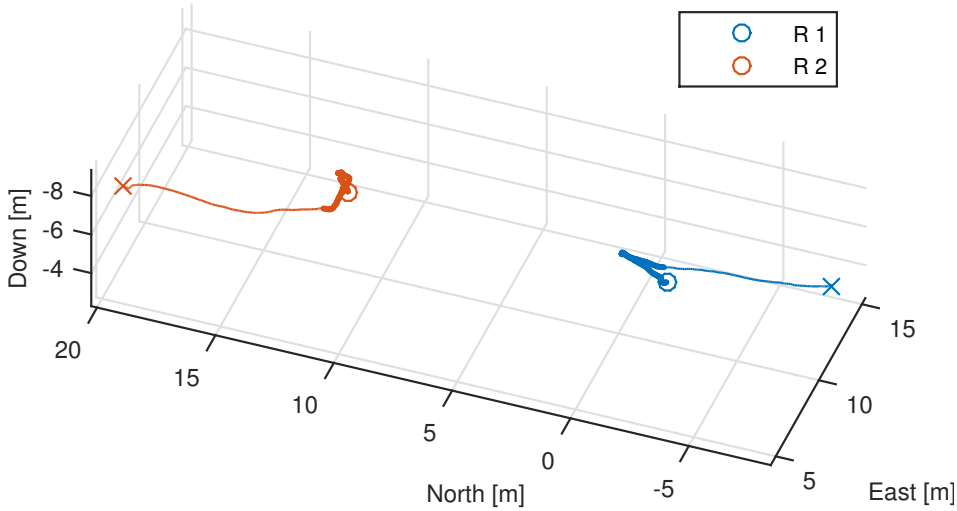


Figure 7.17: 3D plot of Experiment 2-1. Initial positions are marked with a cross, while the colored lines show the trajectory of each rover, turning bold after reaching the desired formation.

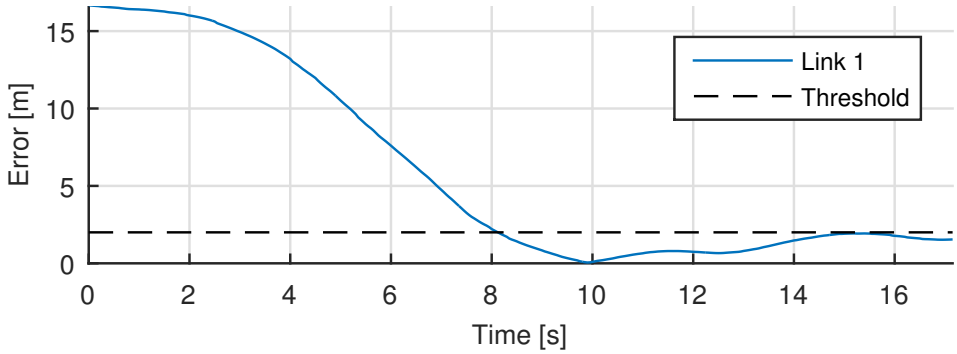


Figure 7.18: Link error during Experiment 2-1. An arbitrary threshold, here set at 2 meters, decides when the rovers are in formation, and is only used during plotting.

Experiment 2-2: 1 waypoint

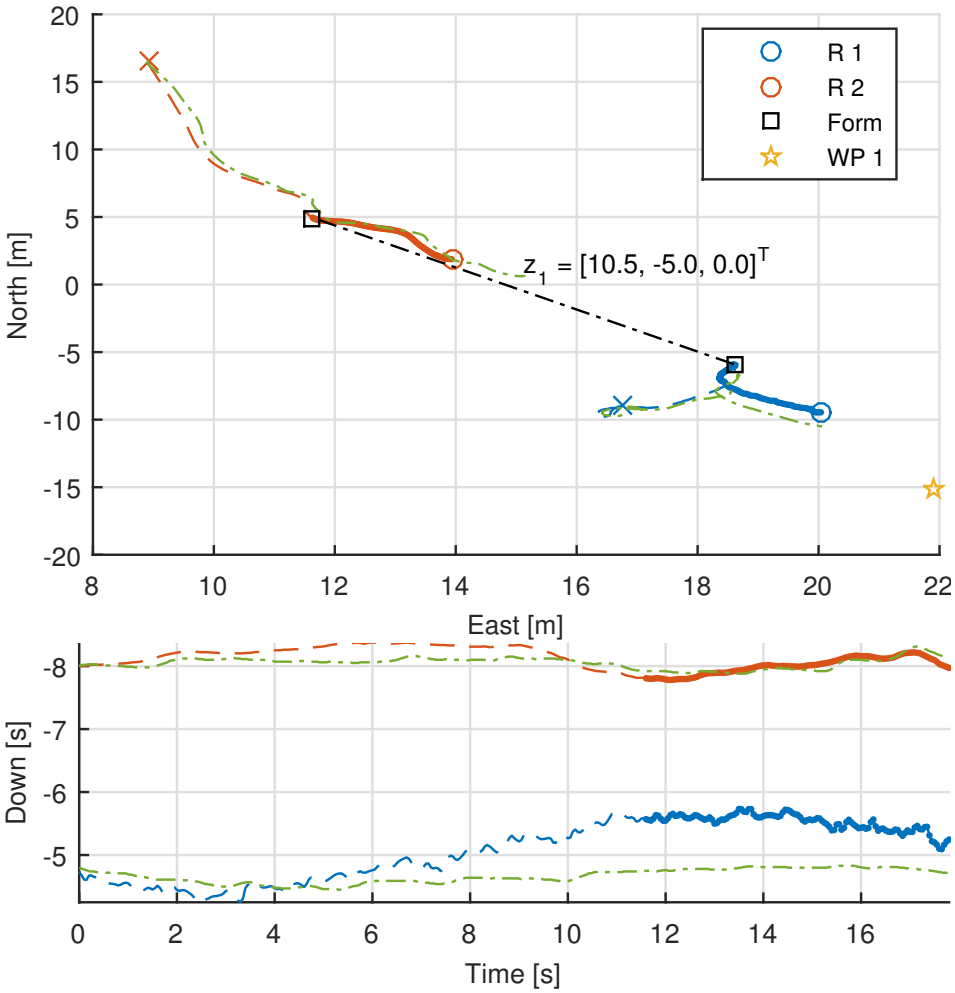


Figure 7.19: Results from Experiment 2-2. Initial positions are marked with a cross, while the dashed colored lines show the trajectory of each rover, turning solid after reaching the desired formation drawn by the black line. The green lines shadowing the rover trajectories are their respective positions from the Pixhawk. Note that the Down axis is reversed, since a negative value equals positive altitude.

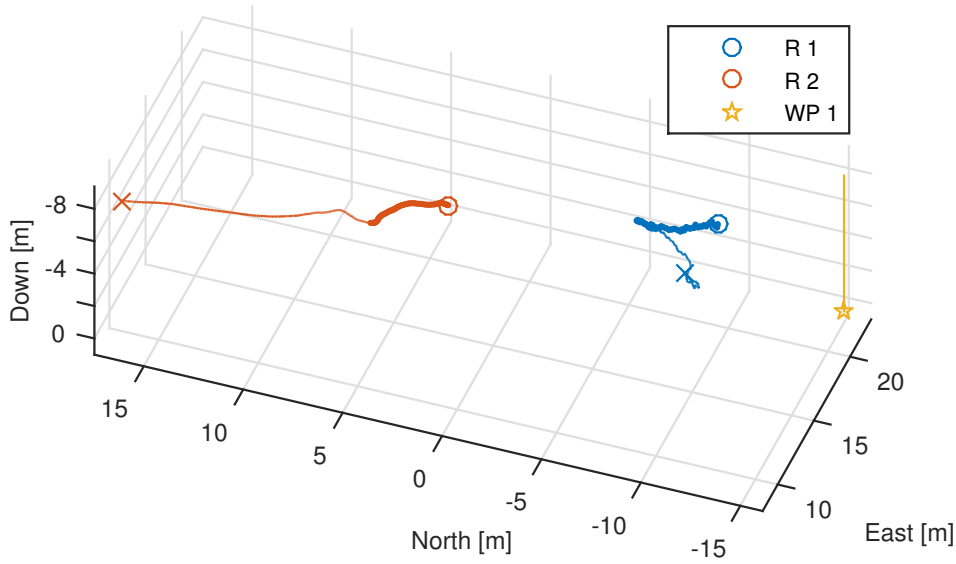


Figure 7.20: 3D plot of Experiment 2-2. Initial positions are marked with a cross, while the colored lines show the trajectory of each rover, turning bold after reaching the desired formation. The waypoint is marked by a colored star, pinned by a vertical bar for visibility.

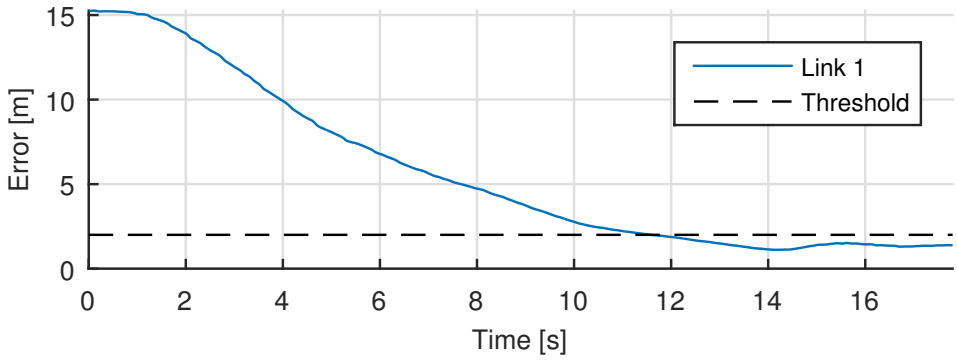


Figure 7.21: Link error during Experiment 2-2. An arbitrary threshold, here set at 2 meters, decides when the rovers are in formation, and is only used during plotting.

7.5 Discussion

When comparing the rover velocities from the SIL simulation in Figure A.1 and the experiments in Figure A.2 to A.6, we see a striking difference in the correlation between desired and actual speeds. Except for some oscillations when the desired velocities change rapidly, the rovers manage to track them reasonably well during the simulation. By contrast, the rovers in the experiments tend to lag somewhat behind the desired velocity, especially noticeable in the north- and eastward components in Figure A.2.[6] Oscillations are also much more prominent in the experiments, especially in the downward component of Figure A.2, A.3 and A.4.

The large oscillations in height from the first set of experiments was probably caused by poor accuracy in the altitude measurements from Pixhawk, as these are based on barometer pressure readings, which downwash from the rotors can interfere with. Adding to that the undesirable wind conditions, the ability to deactivate autonomous control of the altitude was implemented as a safety option and used in the second set of experiments presented. It would have been interesting to see if the altitude control fared any better with high-precision measurements from Piksi, but unfortunately the only successful missions with RTK fix was performed with this 2D control. However, since RTK from Piksi in the second set of experiments was only available for the DUNE controller producing the setpoints, the slow response of the velocity controller in APM:Copter is still apparent in Figure A.5 and A.6. Proper integration of RTK navigation with the autopilot would surely improve the results significantly. In addition, part of the velocity controller's poor performance can probably be attributed to suboptimal tuning.

Nonetheless, the experiments show that the implementation of cooperative formation control was successful, even with a poorly tuned velocity controller, noticeable delays and strong wind. Substituting the mission velocity with more complex 3D guidance would be a trivial task, and make the formation of rovers able to solve more intricate group objectives.

[6] Note that the delay might seem less in Figure A.3 and A.4 because of the larger timescale of those experiments.

Chapter 8

Conclusion and Closing Discussions

The goal of this thesis to develop a system of multirotors capable of performing tasks cooperatively was achieved. The final system was verified through simulations and demonstrated with experiments of successful formation flying and waypoint tracking.

Firstly, a complete multirotor platform with a custom payload module was presented, along with an introduction to the necessary software tools. Further, a mathematical model for the multirotor was developed, and it was shown, with certain assumptions to existing low-level control, how the dynamics could be regarded by translational equations only.

Next, an RTK GPS receiver called Piksi was investigated to provide high-precision navigation. Experiments showed an impressive accuracy in optimal, static conditions, with an average standard deviation in the absolute position of only 1 centimetre. Moreover, the combination of a special helix antenna for the multirotors together with a multipath resistant pinwheel antenna for the base station, proved a good resilience to tilting while still filtering out undesired satellites. However, several firmware stability issues were encountered, which together with an interference issue with the multirotor telemetry, limited the usefulness of Piksi in this thesis.

A passivity-based method to cooperative control was investigated, and used to solve formation control by developing decentralized feedback laws, capable of making a group of agents converge to desired relative positions. Further, the solution allows great flexibility for the formation of agents to pursue common objectives, through the tailoring of a shared mission velocity.

Moreover, a framework known as the LSTS Software Toolchain was used to realize cooperative control for the multirotors. It was shown how the developed feedback laws were implemented as a combination of high-level guidance of setpoints in DUNE, using a

velocity controller present in APM:Copter on the Pixhawk autopilot to achieve control. Piksi was integrated with DUNE to provide accurate, relative guidance for the multirotors, although full integration with the autopilot was not performed because of the issues with stability. Further, the mission velocity was implemented as a simple waypoint tracker, before the complete realization was verified through SIL simulation.

Finally, the validity of the whole system was demonstrated in numerous field experiments with formation flight of two multirotors. Although successful, the experiments uncovered several problems. Most noticeable was the slow response of the velocity controller in APM:Copter, which severely limited the performance in challenging environmental conditions. Furthermore, the instability and interference experienced with Piksi impeded the full benefit of RTK navigation for the multirotors. In conclusion, this leaves room for much improvement in future work. Although not exhaustive, some of the main areas that should be considered are addressed in the next section.

8.1 Future Work

Improving the performance of the velocity controller in APM:Copter is fundamental to achieving better performance of the overall system. As mentioned in Section 7.5, better tuning as well as tight integration with RTK GPS should go a long way, although even better performance would probably be gained by implementing the lower-level control in DUNE. This would eliminate the delay in Table 5.2, and with an alternative IMU and compass it could replace Pixhawk altogether. This would also circumvent the telemetry rate limitation of 50 Hz from the Pixhawk.

If Piksi is still to be used for RTK navigation in the system, the interference discussed in Section 3.5 must be properly mitigated. This could involve proper analysis of the interference with a spectrometer, followed by applying the ensuing remedies. Another alternative would be to substitute the telemetry with one that doesn't cause the same interference. The latter is advisable anyway, as one would surely see improvements to communication delays and throughput by choosing a telemetry that operates in a less busy band than the crowded 2.4 GHz of PicoStation.

Lastly, the fact that ArduCopter Hexa B has gone out of production, as well as the low effective flight time of 5-10 minutes, assert the need to replace the multirotor base platform. Indeed, the UAV-Lab is already in the process of acquiring new carbon fiber octocopter frames, replacing the old aluminum-based hexacopters. Adapting the system to these new frames will give an increased flight time, and significantly improve lifting capacity, which further will extend the possible applications.

Appendix A

Supplementary Figures

A.1 Rover Velocities from Simulations and Experiments

A.1.1 AP-SIL simulation

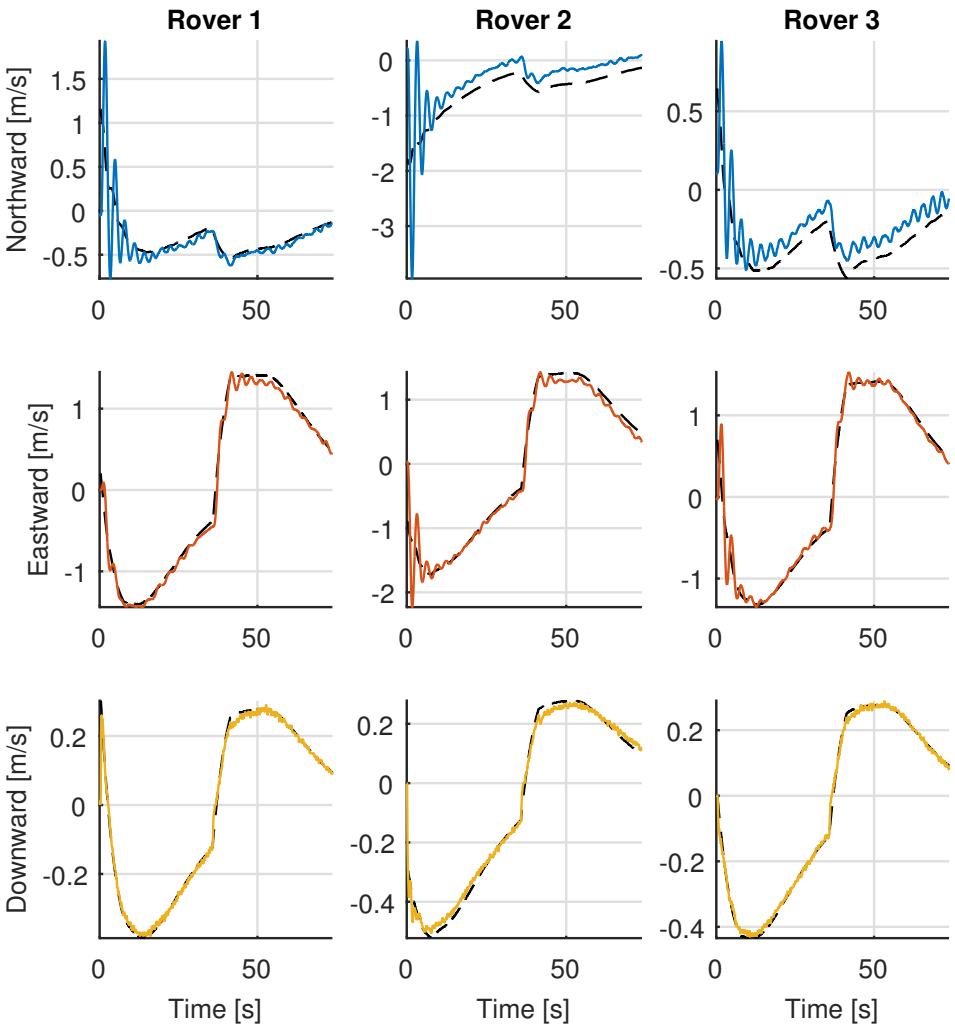


Figure A.1: Rover velocities from the AP-SIL simulation. The black dashed lines indicate the desired velocity $\boldsymbol{v}^{\star}$.

A.1.2 Experiment Set 1 - Pixhawk navigation

Experiment 1-1: Formation only

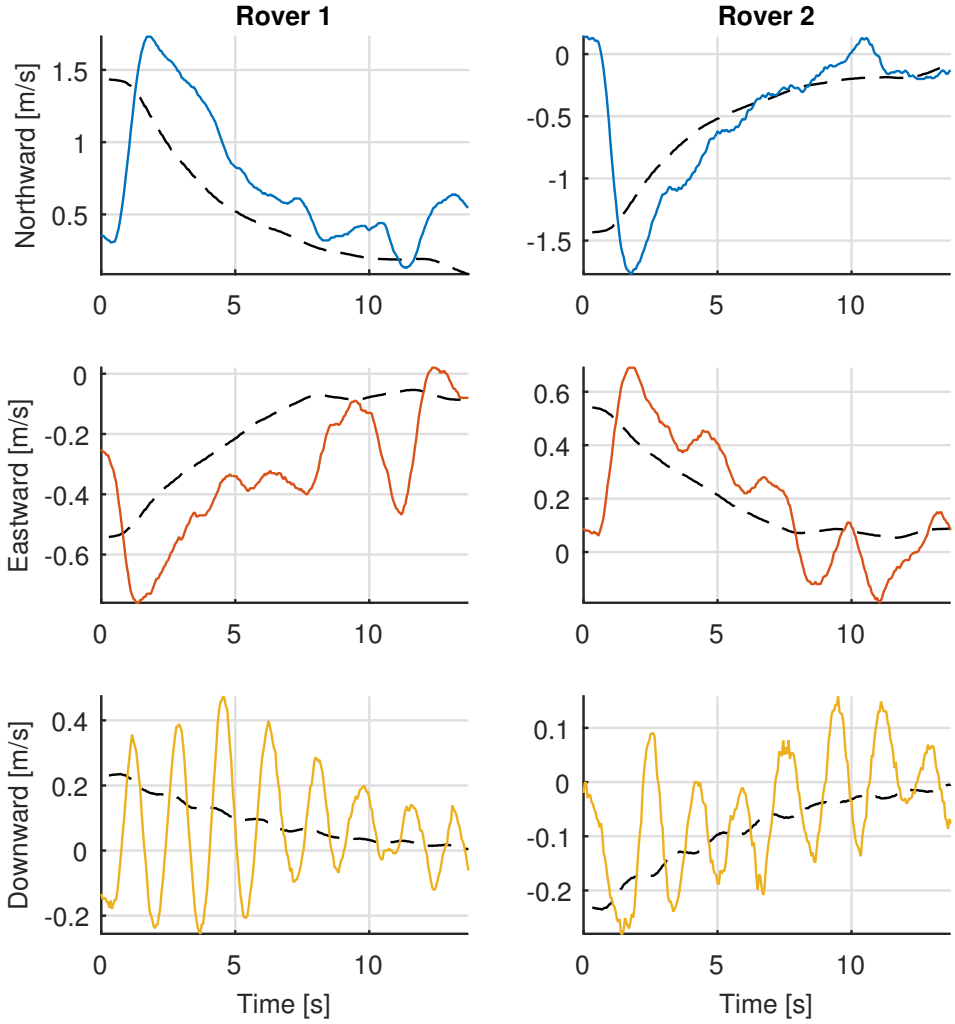


Figure A.2: Rover velocities from Experiment 1-1. The black dashed lines indicate the desired velocity $\boldsymbol{v}^{\star}$.

Experiment 1-2: 1 waypoint

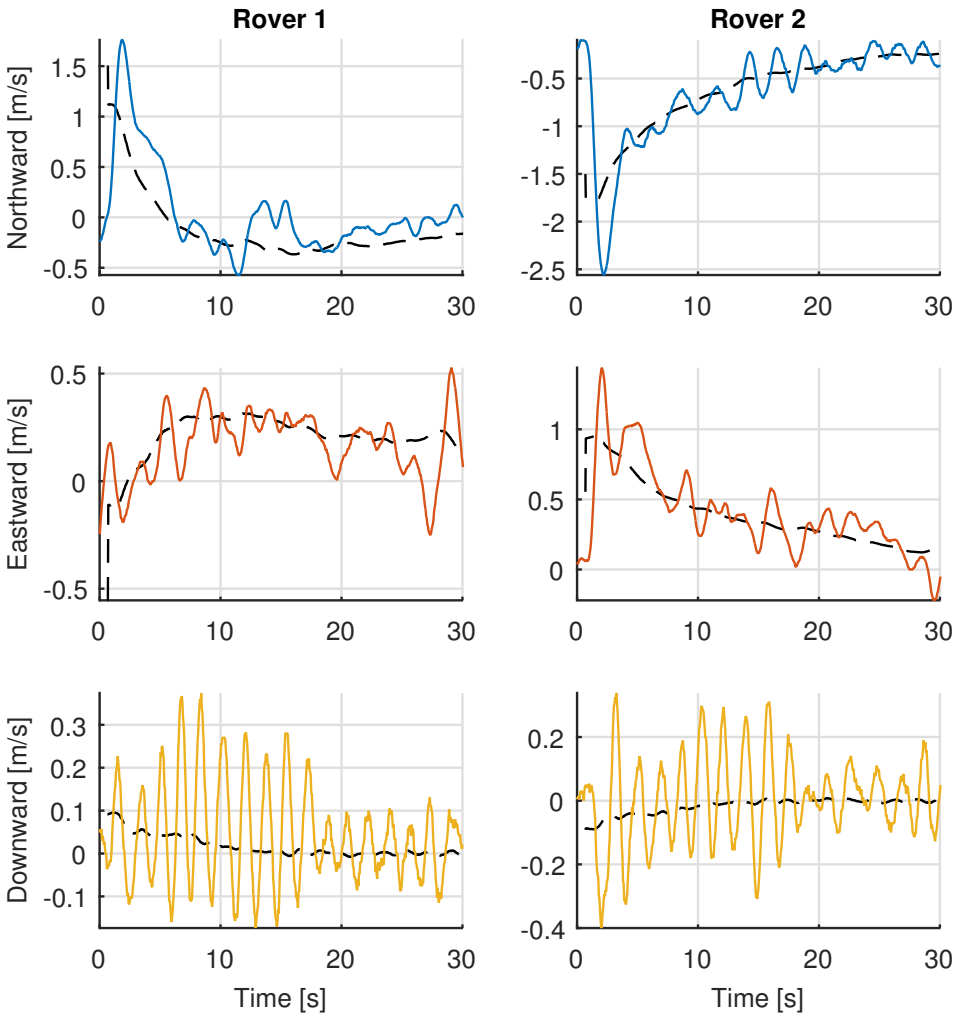


Figure A.3: Rover velocities from Experiment 1-2. The black dashed lines indicate the desired velocity $\boldsymbol{v}^{\star}$.

Experiment 1-3: 2 waypoints

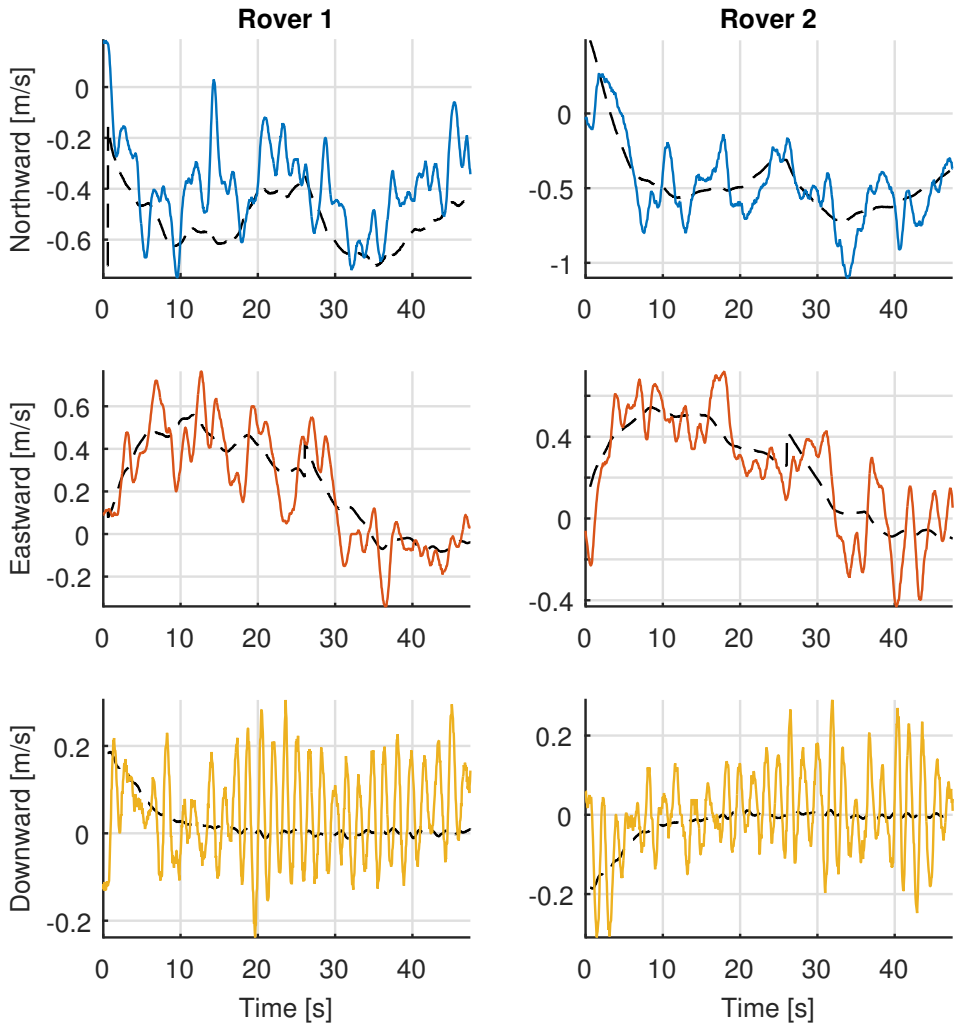


Figure A.4: Rover velocities from Experiment 1-3. The black dashed lines indicate the desired velocity $\boldsymbol{v}^{\star}$.

A.1.3 Experiment Set 2 - Piksi navigation

Experiment 2-1: Formation only

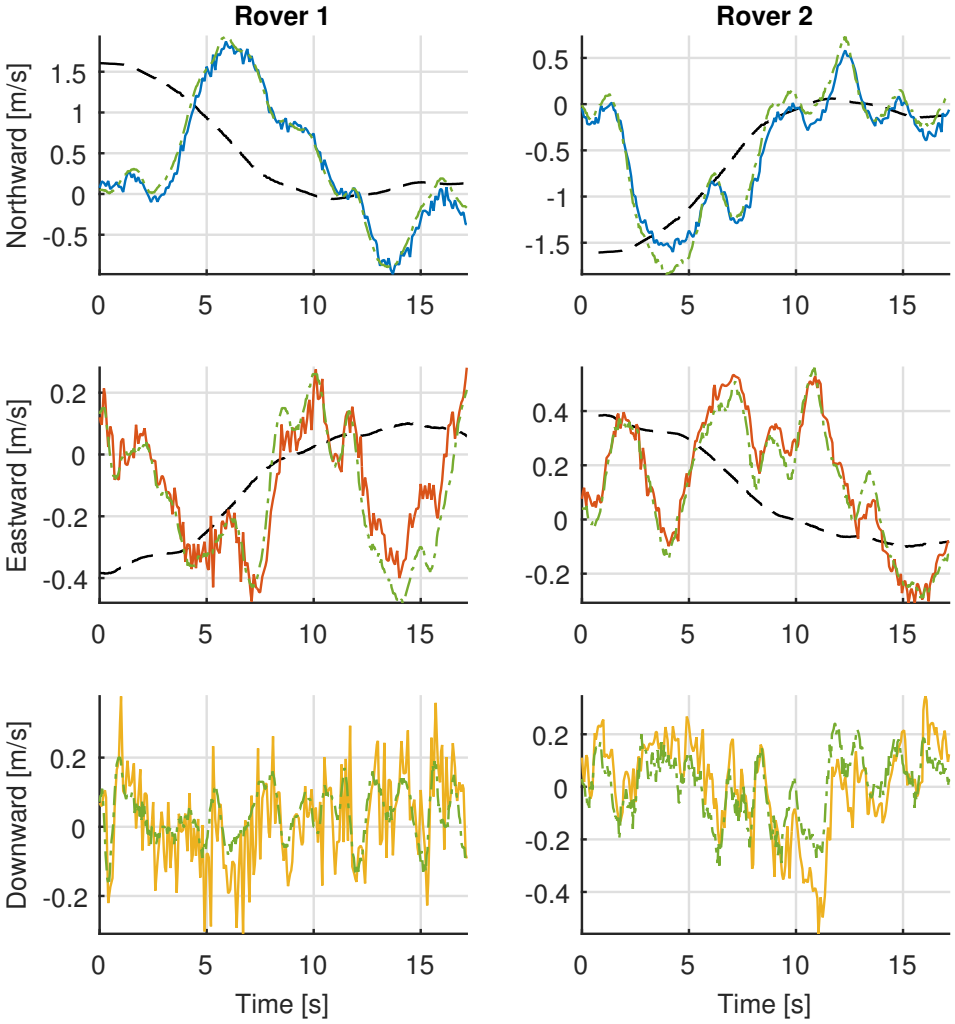


Figure A.5: Rover velocities from Experiment 2-1. The black dashed lines indicate the desired velocity $\boldsymbol{v}^{\star}$, while the green lines are are velocities from the Pixhawk. Note that there is no desired downward speed as only 2D control was in effect.

Experiment 2-2: 1 waypoint

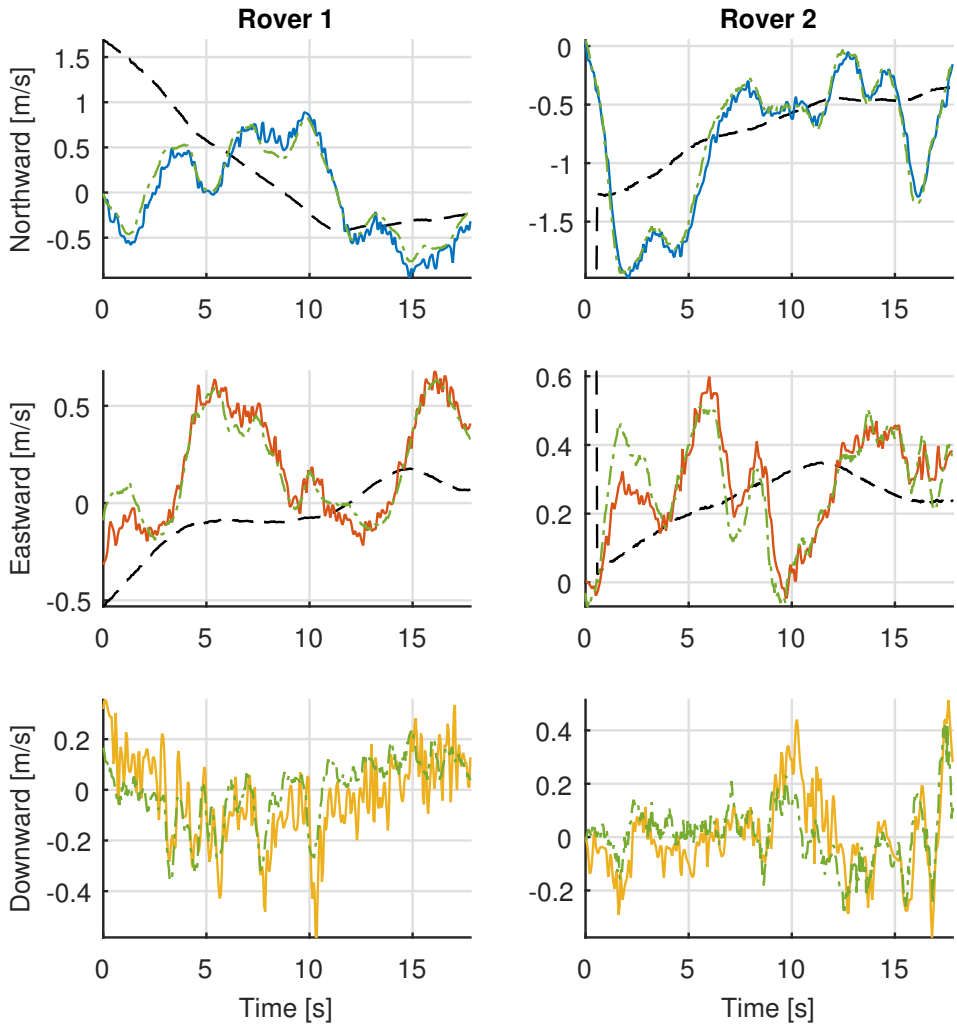


Figure A.6: Rover velocities from Experiment 2-2. The black dashed lines indicate the desired velocity $\boldsymbol{v}^{\star}$, while the green lines are are velocities from the Pixhawk. Note that there is no desired downward speed as only 2D control was in effect.

Bibliography

3D Robotics (2014a). *Arducopter 3DR Hexa B*. URL: http://store.3drobotics.com/products/arducopter-3dr-hexa-b-1 (visited on 10/28/2014).

— (2014b). *Mission Planner*. URL: http://planner.ardupilot.com/ (visited on 12/15/2014).

— (2014c). *Pixhawk Autopilot System*. URL: http://3drobotics.com/pixhawk-autopilot-system/ (visited on 10/28/2014).

— (2015). *3DR-radio*. URL: https://store.3drobotics.com/products/3dr-radio-433_mhz (visited on 05/21/2015).

AMOS (2015). *The Unmanned Aerial Vehicle Laboratory*. URL: http://uavlab.itk.ntnu.no/ (visited on 06/10/2015).

Andersen, Håvard Lægreid (2014). "Path Planning for Search and Rescue Mission Using Multicopters". MA thesis. Norwegian University of Science and Technology.

Arcak, Murat (2007). "Passivity as a Design Tool for Group Coordination". In: *Automatic Control, IEEE Transactions on* 52.8, pp. 1380–1390.

Bai, He, Murat Arcak, and John Wen (2011). *Cooperative Control Design - A systematic Passivity-Based Approach*. Vol. 89. Springer. ISBN: 9781461400134.

Bäumker, M, HJ Przybilla, and A Zurhorst (2013). "Enhancements in UAV Flight Control and Sensor Orientation". In: *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 1.2, pp. 33–38.

BeagleBoard (2014). *BeagleBone Black*. URL: http://beagleboard.org/black (visited on 12/16/2014).

Belleter, Dennis Johannes Wouter and Kristin Ytterstad Pettersen (2014). "Path Following for Formations of Underactuated Marine Vessels Under Influence of Constant Ocean Currents". In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, pp. 4521–4528.

Børhaug, Even et al. (2011). "Straight Line Path Following for Formations of Underactuated Marine Surface Vessels". In: *Control Systems Technology, IEEE Transactions on* 19.3, pp. 493–506.

Broek, Thijs HA Van den, Nathan Van de Wouw, and Henk Nijmeijer (2009). "Formation Control of Unicycle Mobile Robots: A Virtual Structure Approach". In: *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*. IEEE, pp. 8328–8333.

Cao, Yongcan et al. (2013). "An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination". In: *Industrial Informatics, IEEE Transactions on* 9.1, pp. 427–438.

Cetin, Recep (2015). "Indoor Navigation System and Suspended Load Control for Multirotors". MA thesis. Norwegian University of Science and Technology.

Cheung, Yushing, Jae H Chung, and Norman P Coleman (2009). "Semi-autonomous Formation Control of a Single-master Multi-slave Teleoperation System". In: *Computational Intelligence in Control and Automation, 2009. CICA 2009. IEEE Symposium on*. IEEE, pp. 117–124.

Chung, Soon-Jo and J-JE Slotine (2009). "Cooperative Robot Control and Concurrent Synchronization of Lagrangian Systems". In: *Robotics, IEEE Transactions on* 25.3, pp. 686–700.

DIY Drones (2015a). *APM:Copter*. URL: http://copter.ardupilot.com/ (visited on 05/21/2015).

— (2015b). *APM:Copter, GitHub*. URL: https://github.com/diydrones/ardupilot/tree/master/ArduCopter (visited on 05/21/2015).

— (2015c). *ArduPilot Software-in-the-Loop*. URL: http://dev.ardupilot.com/wiki/simulation-2/sitl-simulator-software-in-the-loop/ (visited on 05/25/2015).

— (2015d). *Attitude Control (Copter Code Overview)*. URL: http://dev.ardupilot.com/wiki/apmcopter-code-overview/apmcopter-programming-attitude-control-2/ (visited on 06/12/2015).

— (2015e). *MAVLink Micro Air Vehicle Communication Protocol*. URL: http://qgroundcontrol.org/mavlink/start (visited on 05/21/2015).

Dunbar, William B and Richard M Murray (2006). "Distributed Receding Horizon Control for Multi-Vehicle Formation Stabilization". In: *Automatica* 42.4, pp. 549–558.

Fossen, Thor I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd.

Ghabcheloo, Reza et al. (2009). "Coordinated Path-following in the Presence of Communication Losses and Time Delays". In: *SIAM Journal on Control and Optimization* 48.1, pp. 234–265.

Hofmann-Wellenhof, Bernhard, Herbert Lichtenegger, and Elmar Wasle (2007). *Global Navigation Satellite Systems: GPS, GLONASS, Galileo & more*. Springer Science & Business Media.

Ihle, Ivar-André F, Murat Arcak, and Thor I Fossen (2007). "Passivity-based Designs for Synchronized Path-Following". In: *Automatica* 43.9, pp. 1508–1518.

Kaminer, Isaac et al. (2007). "Coordinated Path Following for Time-Critical Missions of Multiple UAVs via L1 Adaptive Output Feedback Controllers". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6409.

Khalil, Hassan K. (2002). *Nonlinear Systems*. Vol. 3. Prentice-Hall. ISBN: 0131227408.

Kickstarter (2014). *Piksi the RTK receiver*. URL: https://www.kickstarter.com/projects/swiftnav/piksi-the-rtk-gps-receiver (visited on 12/14/2014).

Klausen, Kristian (2013). "Cooperative Behavioural Control for Omni-Wheeled Robots". MA thesis. Norwegian University of Science and Technology.

Klausen, Kristian, Thor Fossen, Tor Arne Johansen, et al. (2014). "Suspended Load Motion Control Using Multicopters". In: *Control and Automation (MED), 2014 22nd Mediterranean Conference of*. IEEE, pp. 1371–1376.

Lee, Taeyoung, Melvin Leoky, and N Harris McClamroch (2010). "Geometric Tracking Control of a Quadrotor UAV on SE (3)". In: *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, pp. 5420–5425.

Leonard, Naomi Ehrich and Edward Fiorelli (2001). "Virtual Leaders, Artificial Potentials and Coordinated Control of Groups". In: *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*. Vol. 3. IEEE, pp. 2968–2973.

LSTS (2015a). *DUNE: Unified Navigation Environment*. URL: http://lsts.fe.up.pt/toolchain/dune (visited on 05/09/2015).

— (2015b). *GLUED/Linux Uniform Environment Distribution*. URL: http://lsts.fe.up.pt/toolchain/glued (visited on 05/09/2015).

— (2015c). *Inter-Module Communication protocol*. URL: http://lsts.fe.up.pt/toolchain/imc (visited on 05/09/2015).

— (2015d). *LSTS, GitHub*. URL: https://github.com/LSTS (visited on 05/10/2015).

— (2015e). *Neptus Command and Control Software*. URL: http://lsts.fe.up.pt/toolchain/neptus (visited on 05/09/2015).

— (2015f). *Underwater Systems and Technology Laboratory*. URL: http://lsts.fe.up.pt/ (visited on 05/09/2015).

Mahony, Robert, Vijay Kumar, and Peter Corke (2012). "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor". In: *IEEE Robotics & amp amp Automation Magazine* 19, pp. 20–32.

MATLAB (2015). *version 8.5.0.197613 (R2015a)*. Natick, Massachusetts, USA: The MathWorks Inc.

Maxtena (2014). *M1227HCT-A-SMA Datasheet*. URL: http://www.maxtena.com/uploads/6/6/6/5/6665461/m1227hct-a-sma.pdf (visited on 10/31/2014).

Moreau, Luc (2005). "Stability of Multiagent Systems with Time-Dependent Communication Links". In: *Automatic Control, IEEE Transactions on* 50.2, pp. 169–182.

Murray, Richard M (2007). "Recent Research in Cooperative Control of Multivehicle Systems". In: *Journal of Dynamic Systems, Measurement, and Control* 129.5, pp. 571–583.

NovAtel (2014). *GPS-701-GG Datasheet*. URL: http://www.novatel.com/assets/Documents/Papers/GPS701_702GG.pdf (visited on 10/31/2014).

Olfati-Saber, Reza, J Alex Fax, and Richard M Murray (2007). "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE* 95.1, pp. 215–233.

Olfati-Saber, Reza and Richard M Murray (2002). "Distributed Cooperative Control of Multiple Vehicle Formations Using Structural Potential Functions". In: *IFAC World Congress*, pp. 346–352.

openAIP (2015). *Agdenes Airfield, Breivika*. URL: http://www.openaip.net/node/157338 (visited on 06/02/2015).

Pettersen, Kristin Y, Jan Tommy Gravdahl, and Henk Nijmeijer (2006). "Group Coordination and Cooperative Control". In:

Qu, Zhihua, Jing Wang, and Richard A Hull (2008). "Cooperative Control of Dynamical Systems with Application to Autonomous Vehicles". In: *Automatic Control, IEEE Transactions on* 53.4, pp. 894–911.

Ren, Wei, Randal W Beard, and Ella M Atkins (2007). "Information Consensus in Multivehicle Cooperative Control". In: *IEEE Control systems magazine* 2.27, pp. 71–82.

Ren, Wei, Randal W Beard, et al. (2005). "Consensus Seeking in Multiagent Systems Under Dynamically Changing Interaction Topologies". In: *IEEE Transactions on automatic control* 50.5, pp. 655–661.

Skulstad, Robert and Christoffer Lie Syversen (2014). "Low-cost Instrumentation System for Recovery of Fixed-Wing UAV in a Net". MA thesis. Norwegian University of Science and Technology.

Smith, Randall W (1987). *Department of Defense World Geodetic System 1984: Its Definition and Relationships with Local Geodetic Systems*. Defense Mapping Agency.

SolidWorks (2014). *version 22.3.0.56 (2014 SP03)*. Waltham, Massachusetts, USA: Dassault Systèmes SolidWorks Corp.

Steen, Thor Audun (2014). "Search and Rescue Mission Using Multicopters". MA thesis. Norwegian University of Science and Technology.

Stempfhuber, W (2013). "3D-RTK Capability of Single GNSS Receivers". In: *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 1.2, pp. 379–384.

Svendsen, Jon Glenn Gjevestad (2001). "Real-Time Phase Ambiguity Resolution in Global Navigation Satellite Systems". PhD thesis.

SwiftNAV (2013). *Piksi Datasheet v2.3.1*. URL: http://docs.swift-nav.com/pdfs/piksi_datasheet_v2.3.1.pdf (visited on 05/20/2015).

— (2015a). *General discussion community*. URL: https://groups.google.com/forum/#!forum/swiftnav-discuss (visited on 05/20/2015).

— (2015b). *libswiftnav documentation*. URL: http://docs.swift-nav.com/libswiftnav/ (visited on 05/20/2015).

— (2015c). *SBP client libraries*. URL: https://github.com/swift-nav/libsbp (visited on 05/20/2015).

— (2015d). *Swift Navigation store*. URL: http://store.swiftnav.com/ (visited on 05/19/2015).

— (2015e). *The Swift Navigation Binary Protocol*. URL: http://docs.swiftnav.com/wiki/SwiftNav_Binary_Protocol (visited on 05/20/2015).

Takasu, Tomoji and Akio Yasuda (2004). "Cycle Slip Detection and Fixing by MEMS - IMU / GPS Integration for Mobile Environment RTK-GPS". In:

— (2009). "Development of the Low-cost RTK-GPS Receiver with an Open Source Program Package RTKLIB". In: *International Symposium on GPS/GNSS*, pp. 4–6.

Tridgell, Andrew (2015). *MAVProxy: A UAV ground station software package for MAVLink based systems*. URL: http://tridge.github.io/MAVProxy/ (visited on 05/25/2015).

Ubiquiti Networks (2014). *PICOM2HP Datasheet*. URL: http://www.ubnt.com/downloads/datasheets/picostationm/picom2hp_DS.pdf (visited on 12/12/2014).

Walpole, Ronald E et al. (1993). *Probability and Statistics for Engineers and Scientists*. Vol. 5. Macmillan New York.

Wu, Chai Wah and Leon O Chua (1995). “Application of Graph Theory to the Synchronization in an Array of Coupled Nonlinear Oscillators”. In: *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on* 42.8, pp. 494–497.

Xargay, E et al. (2013). “Time-Critical Cooperative Path Following of Multiple Unmanned Aerial Vehicles over Time-Varying Networks”. In: *Journal of Guidance, Control, and Dynamics* 36.2, pp. 499–516.