NTNU
Norwegian University of
Science and Technology

Autonomous landing of Fixed-Wing UAV in net suspended by Multirotor UAVs

A Multirotor recovery system

Jostein Borgen Moe

Master of Science in Cybernetics and Robotics	
Submission date:	June 2016
Supervisor:	Tor Arne Johansen, ITK
Co-supervisor:	Thor Inge Fossen, ITK
	Kristian Klausen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU	**Faculty of Information Technology,**
Norwegian University of	**Mathematics and Electrical Engineering**
Science and Technology	**Department of Engineering Cybernetics**

MSC THESIS DESCRIPTION SHEET

Name:	Jostein Borgen Moe
Department:	Engineering Cybernetics
Thesis title:	Autonomous landing of Fixed-Wing UAV in net suspended by Multirotor UAVs - A Multirotor recovery system

Thesis Description:

The purpose of this thesis is to study the operation of landing a fixed-wing Unmanned Aerial Vehicle (UAV) in a net suspended by two multirotor type UAVs. In addition to the scientific contribution, the work will provide an important demo for the capabilities of the UAV lab at ITK.

The following items should be considered:

1. Investigate methods to coordinate along-track speed of the multirotor formation with the fixed-wing surge speed.
2. Literature overview on modeling the constrained dynamics of multiple multirotors in 6DOF with a suspended net.
3. Modeling of impact effects when the fixed-wing hits the suspended net.
4. Investigate the usability of adaptive control for the multirotors.
5. Guidance system for the multirotor formation.
6. Develop simulators to enable *software-in-the-loop* simulations of the interconnected system (fixed-wing + nx multirotors).
7. Discussions on the practical usage of a net-catch maneuver (Including operational aspects.)
8. Test and implement the synchronized controller for net-catching in a framework used for real UAV operations (DUNE) with emphasis on the multirotor controllers.
9. Prove the usability of the proposed solution in an experimental setup.
10. Conclude findings in a report. Include a user-guide for the C-code as digital appendices.

Start date:	2016-01-11
Due date:	2016-06-20

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Tor Arne Johansen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor:	Kristian Klausen, Dept. of Eng. Cybernetics, NTNU

Abstract

This thesis considers the recovery of a fixed-wing Unmanned Aerial Vehicle (UAV) in a net suspended by multirotor UAVs. Motivated by marine fixed-wing UAV operations from ships, where limited landing space and harsh weather conditions often are present, a dynamically controlled net is presented. Traditionally a net is located on the ship-deck, introducing challenges such as heave motion and turbulence near the net. The concept presented in this thesis gives the possibility to move the net away from the ship and optimize the landing trajectory of the fixed-wing UAV according to the current weather conditions. Moreover, the multirotors can carry the net at a certain speed along the desired landing trajectory, hence, reducing the relative velocity and the impact forces of the impact.

A mathematical model of the interconnected multi-body system is presented, emphasizing the impact dynamics. Furthermore, a two-folded preliminary control system architecture is presented, subdivided into the transport and the recovery phase. The transport phase consist of the transportation of a suspended net with multirotors along a desired path, and in the actual recovery phase the net is controlled according to the fixed-wing UAV landing trajectory. The transportation is performed using a Line-Of-Sight (LOS) guidance law in combination with a reference generator. While the recovery phase utilize a modified pure-pursuit guidance law for cross-track control in combination with a desired along-track velocity trajectory, instructed by the approaching fixed-wing UAV. Further different along-track trajectories are proposed with the purpose of controlling the recovery location. The cooperative multirotor control problem are solved using the passivity-based approach given in [Bai et al., 2011] and implemented for cooperative multirotor control in [Røli, 2015]. Furthermore, the controllers are implemented in the open-source *DUNE: Uniform Navigational Environment* framework on an embedded system in combination with the open-source APM:Copter autopilot. The feasibility of the concept are verified by simulations and field experiments.

Samandrag

(Norwegian translation of the abstract)

Denne avhandlinga har som mål å syne ein landingsstrategi for eit ubemanna fly i eit nett spent opp av eit sett med ubemanna multirotorar. Motivert av marine operasjoner med ubemanna fly fra skip, der det er avgrensa landingsplass og krevjande vêrforhold, presenterar denne avhandlinga ei løysing der posisjonen til nettet kan regulerast. Tradisjonelt spennast nettet opp på skipets dekk, men skipets vertikalrørsle og turbulens nær skipet gjev utfordringar for det ubemanna flyet når det nærmar seg nettet. Her presenterast eit konsept som gjev moglegheit til å flytte nettet vekk frå skipet og optimalisere landingsbanen til det ubemanna flyet gitt dei gjeldande vêrforholda. Vidare kan dei ubemanna multirotorane bere nettet i ein bestemd hastighet langs den ynskte landingsbanen, på den måten vert den relative hastigheita og kreftene i kollisjonen mellom nettet og flyet redusert.

Ein matematisk modell av fleir-legeme systemet er presentert med vekt på kollisjonsdynamikken. Vidare er eit to-delt reguleringssystem presentert med det føremål å høvevis transportere det oppstrekte nettet og gjennomføre den faktiske landingsmanøveren. Føremålet med transportfasen er å flytte eit nett strekt opp av eit sett med ubemanna multirotorar langs ein ynskja bane. På den andre sida skal nettet regulerast i samsvar med det ubemanna flyet sin landingsbane under sjølve landingsfasen. Transporten av nettet er utført ved hjelp av en *Line-of-Sight* (LOS) reguleringslov i kombinasjon med ein referanse generator. Under landingsfasen vert det nytta ein modifisert *Pure-Pursuit* reguleringslov, der posisjonen i planet ortogonalt på landingsbanen vert regulert kombinert med ein ynskja hastighet langsmed landingsbanen, instruert av det ubemmane flyet. Utfordringa knytt til det samarbeidande reguleringssystemet for dei ubemanna multirotorane er løyst ved hjelp av ein metode basert på passivitet. Denne metoden er introdusert i [Bai et al., 2011] og implementert for eit sett med ubemanne multirotorar i [Røli, 2015]. Vidare er reguleringssystemet implementert i eit åpen-kjelde rammeverk *DUNE: Uniform Navigational Environment* på eit innvevd datasystem i kombinasjon med APM:Copter, ein åpen-kjeldekode autopilot. Til slutt er konseptet verifisera ved hjelp av simuleringar og feltforsøk.

Preface

This Master's thesis on the subject of autonomous UAV operations at the Norwegian University of Science and Technology (NTNU) was carried out during the spring semester of 2016 in the Engineering Cybernetics study program.

The work continuous the pre-project carried out during the autumn semester of 2015 as a part of the course *TTK4550 - Specialization Project* in Engineering Cybernetics. All equipment used in the work is from the Unmanned Aerial Vehicles Laboratory (UAV-Lab), a test facility for NTNU's Research on UAV, and a part of the Centre for Autonomous Marine Operations and Systems (AMOS). Furthermore, this work continuous and extends the cooperative controller as implemented in the Master's thesis of Jon-Håkon Bøe Røli [Røli, 2015] carried out in the same study program.

The concept presented in this thesis relies on the work done by with the fellow MSc. students Sigurd Olav Nevstad and Kjetil Hope Sørbø. Here, Nevstad [Nevstad, 2016] presents a fixed-wing UAV landing system cooperating with the multirotor UAVs. A high-precision navigation system is presented by Sørbø [Sørbø, 2016] which was utilized for the multirotor control system.

I would like to thank my supervisors, Tor Arne Johansen and Thor Inge Fossen for the opportunity to work with an innovative project in a research field of growing interest, as well as advices and help during the process. Furthermore, the NTNU UAV operators Lars Semb and Pål Kvaløy receives my gratitude for their patience and feedback during the field experiments. Finally, I would also thank my co-supervisor the PhD Candidate Kristian Klausen for all his help and advices during the project.

Trondheim, June 15, 2016

Jostein Borgen Moe

Table of Contents

Thesis Description **i**

Abstract **iii**

Samandrag **v**

Preface **vii**

Table of Contents **xii**

List of Tables **xiii**

List of Figures **xviii**

Abbreviations **xix**

1 Introduction **1**
1.1 Background and Motivation 1
1.2 Objectives and methods 1
1.3 Previous work 2
1.4 Contributions 3
1.5 Organization of the thesis 4

2 Notation and background theory **7**
2.1 Definitions 7
2.2 Classical mechanics 10
2.3 Aerodynamic drag 11
2.4 Rigid-body dynamics 12

3 System **15**
3.1 Overview 15

3.2 Platforms 15
3.3 Software 17
3.3.1 LSTS Software Toolchain 17
3.3.2 ArduPilot software 18
3.4 Hardware 19
3.4.1 Autopilot 19
3.4.2 Navigation 20
3.4.3 Computer 20
3.4.4 Communication 20
3.4.5 Suspended payload sensors 21

4 The multi-body dynamics 23
4.1 Unconstrained body dynamics 23
4.1.1 Multi-rotor dynamics 23
4.1.2 Net dynamics 25
4.2 Impact dynamics 26
4.3 Multi-body constrained dynamics for simulation 27
4.4 Dynamics for control analysis 29
4.5 Collision simulation 30

5 Cooperative control 33
5.1 Centroid - *the virtual vehicle* 33
5.2 Centroid control 36
5.2.1 Heading and formation control 36
5.2.2 Link gain scheduling 37
5.2.3 Velocity control 37
5.2.4 Adaptive control 38
5.3 Simulations 41
5.3.1 Adaptive link gain 41

6 Payload transport 43
6.1 Centroid LOS steering law 43
6.1.1 Reference LOS velocity 44
6.1.2 Surge and heading 44
6.2 Marine craft simulator 45
6.2.1 Surge dynamics and control 46
6.2.2 Heading dynamics and control 46
6.3 Simulations 47
6.3.1 Reference simulator 47
6.3.2 MRAC 49

7 The recovery maneuver 53
7.1 Virtual-runway 53
7.2 Along-track trajectory 54
7.2.1 Polynomial 56
7.2.2 Piecewise 57

7.2.3 Reference model 58
7.2.4 Comparing 59
7.2.5 Fixed-Wing ETA 61
7.3 Cross-track control 61
7.4 Position hold 63
7.5 Supervisor 63
7.6 Simulations 66
7.6.1 Trajectory comparing 66
7.6.2 Recovery maneuver 67
7.6.3 MRAC 70

8 Experimental setup 75
8.1 Architecture 76
8.1.1 Configuration 76
8.1.2 Transport layers 78
8.1.3 Transport configuration 79
8.2 Controllers 79
8.2.1 Recovery Coordinator 80
8.2.2 Path Control 80
8.2.3 Coordinated Velocity Control 81
8.3 Software in The Loop (SITL) 81
8.4 SITL results 84
8.4.1 Payload transport 84
8.4.2 Recovery maneuver 87

9 Operational aspects 91
9.1 Overall maneuver 91
9.2 Abort 92
9.3 Recovery considerations 94
9.4 Mission Control 96

10 Experiments 97
10.1 Test facility - Agdenes airfield 97
10.2 Setup 98
10.3 Case 1: Gain scheduling 99
10.3.1 Setup 99
10.3.2 Results 99
10.4 Case 2: Recovery 102
10.4.1 Setup 102
10.4.2 Case 2a: Minor compensation 103
10.4.3 Case 2b: Major compensation 106
10.5 Case 3: Payload transport 108
10.5.1 Setup 108
10.5.2 Case 3a: Compare reference surge 108
10.5.3 Case 3b: Cooperative 110
10.6 Discussion 112

11 Conclusion and Closing Discussions **113**
11.1 Dynamic models 113
11.2 Transport and recovery concept 114
11.3 Low-level controllers 114
11.4 Further work 115

A Additional Dynamical Modeling Theory **117**
A.1 Modeling multi-body constrained dynamics 117
A.1.1 Udwadia-Kalaba 118
A.2 Collision dynamics 119
A.3 Time dependent equation of motions 121

B Polynomial Trajectory Theory **123**
B.1 Polynomial trajectory generation 123
B.2 Polynomial generation matrices 125

C Adaptive Control Theory **127**
C.1 Model Reference Adaptive Control 127
C.2 Controllability 128

D Controllers – Tuning considerations **129**
D.1 Alternative velocity control 129
D.2 Closed-loop dynamics 130

E Submitted conference paper for ICUAS'16 **131**

Bibliography **146**

List of Tables

2.1 Notation for the different body frames 7

4.1 Masses and velocity set-points 30

5.1 Link gain-scheduling parameters 41

6.1 Key reference simulator parameters in simulation 47
6.2 Key MRAC path-control parameters in simulation 49

7.1 Trajectory parameters 59
7.2 Trajectory comparing simulation parameters 66
7.3 Recovery maneuver simulation parameters 67
7.4 MRAC recovery key parameters 70

8.1 SITL - Controller and feedback frequencies 84
8.2 Payload transport (SITL): Parameters 84
8.3 Recovery (SITL): Parameters 87

9.1 The different legs on the path $\Gamma(s)$ 91
9.2 The different time-parameters 93

10.1 Experiments - Controller and feedback frequencies 98
10.2 Case 1: Gain scheduling – Parameters 100
10.3 Case 2: Recovery – Parameters 103

List of Figures

2.1 Sketch of the platforms in the North-East plane with the approaching fixed-wing airplane $\{a\}$ and the two multirotors $\{c_1\}$ and $\{c_2\}$ with a net suspended between them. 7

3.2 Software architecture, the different software systems are illustrated as the boxes, and the application protocols are given in the *italic text* over the dotted lines. 18
3.3 Hardware architecture, the dotted lines represents wireless communication 19
3.4 Load cell 21
3.5 Angular encoder 21

4.1 Multirotor body frame $\{c_i\}$ definition 24
4.2 Net body frame $\{l\}$ definition 25
4.3 Average collision force $\bar{\mathbf{f}}_l$ during impact on the suspended net, assuming the fixed-wing UAV hits the CG of the net. 26
4.4 Suspended net from two multirotors, illustrating the wire vector $\mathbf{L}_i$ from the suspended load to the multirotor i wire attachment points. 28
4.5 Relative along-track velocity during collision run 31
4.6 Tension force in the wire connecting each multirotor to the suspended net 32
4.7 Snapshots from the collision 32

5.1 Two multirotors, c_1 and c_2 in $\{n\}$, defining the centroid $\bar{c}$ with position $\mathbf{p}^n_{\bar{c}}$ and heading $\psi_{\bar{c}}$. 35
5.2 Desired formation link $\mathbf{z}$ as a function of the zero heading formation $\mathbf{z}_{d,0}$ and the desired formation heading $\psi_{d,\bar{c}}$ 36
5.3 Centroid heading and velocity control for agent $1 \cdots i \cdots k$ 39
5.4 Centroid heading and MRAC velocity control for agent $1 \cdots i \cdots k$ . . . 40
5.5 Link gain $K_l(\tilde{z})$ as a function of the difference error norm $\tilde{z}$ 41
5.6 Comparing position and velocity difference with link gain scheduling enabled and disabled. 42

6.1 Path control with marine craft simulator 47
6.2 Multirotor positions in $\{n\}$ with reference simulator enabled and disabled, the waypoints are given as the blue stars connected by the black lines. 48
6.3 Comparing responses with reference simulator enabled and disabled . . 48
6.4 Reference simulator states and the height. 49
6.5 Multirotor and centroid positions in the North-East plane using MRAC for path-control, the blue stars gives the desired waypoints connected by the black lines. 50
6.6 Velocities of the multirotors in $\{n\}$ during the path-maneuver, comparing the reference, desired and the actual state. 50
6.7 Multirotor adaptive gains during the path-maneuver. 51

7.1 Virtual-runway $\{p\}$ defined in the inertial frame $\{n\}$ 54
7.2 Virtual runway path frame $\{p\}$ in the North-East plane with the fixed-wing UAV $\{a\}$ and the multirotor centroid $\{\bar{c}\}$ during recovery, the dotted lines indicates the boundaries of the virtual runway. 55
7.3 Desired jerk profile $j(t)$, $\Delta t = 2(\Delta_1 + \Delta_2 + \Delta_1) + \Delta_3$ 58
7.4 Comparing reference trajectories, target position x_f marked with the dotted red line 60
7.5 The vehicles in the cross-track frame $\{p_*\}$, bounded by the dotted virtual runway with height h and width w. Where the black dot is the fixed-wing and the circle is the center of the net. The height difference between the centroid $\{\bar{c}\}$ and the net center is given as z_L. 62
7.6 State machine for the recovery maneuver 65
7.7 Recovery structure figure 66
7.8 Comparing along-track responses, the dotted lines indicates the constraints. 67
7.9 Position and orientation of the bodies in $\{n\}$, the center multirotor gives the centroid, the black dotted line gives the boundaries of the virtual-runway. 68
7.10 Positions of centroid and fixed-wing in the *Virtual-Runway* frame $\{p\}$. 69
7.11 Centroid velocities denoted in $\{p\}$ 69
7.12 Positions of the vehicles in the $\{n\}$ frame, the black dotted line gives the boundaries of the virtual-runway 70
7.13 Positions of centroid and fixed-wing in the virtual-runway frame $\{p\}$ . . 71
7.14 Centroid velocities denoted in $\{p\}$ 71
7.15 Velocities of the multirotors in $\{n\}$ during the recovery maneuver, comparing the reference, desired and the actual state 72
7.16 Multirotor adaptive gains during the recovery maneuver 72

8.1 DUNE system architecture overview 75
8.2 DUNE on the different platforms 77
8.3 DUNE control-loops overview 78
8.4 Overview of DUNE feedback 79
8.5 Structure of DUNE controllers implementation, IMC::ELS is short for IMC::EstimatedLocalState 80

8.6 SITL principle 81
8.7 SITL architecture in ArduPilot. *Image courtesy of dev.ardupilot.com* . . 82
8.8 SITL setup 83
8.9 Payload transport (SITL): Multirotor positions in $\{n\}$, centroid illustrated as the red multirotor. The blue stars indicate the way-points connected by solid black lines 85
8.10 Payload transport (SITL): Multirotor positions and centroid velocity. . . 86
8.11 Payload transport (SITL): Reference simulator states and the height. . . 86
8.12 Recovery (SITL): North-East overview of the recovery, where the green and the blue line is the path of the fixed-wing and the multirotors respectively. The centroid path is also plotted as the center multirotor. The black dotted lines represents the boundaries of the virtual-runway and the blue stars is the desired waypoints, connected with the solid black line. 87
8.13 Recovery (SITL): Along- and cross-track positions in the path-frame $\{p\}$, the boundaries of the virtual-runway is represented as the black-dotted line. 88
8.14 Recovery (SITL): Along- and cross-track positions in the path-frame $\{p\}$ a short period before the impact. 88
8.15 Recovery (SITL): The recovery state change timestamps during the operation symbolized as red stars along the fixed-wing path (black line), the blue circles is the position of the centroid at the specific timestamp.s 89
8.16 Recovery (SITL): Centroid velocities denoted in the virtual-runway frame $\{p\}$ 89

9.1 Overall plan in the North-East plane $\{n\}$ where $\{s\}$ presents body frame of the ship with heading ψ_s for illustration, and the approaching fixed-wing $\{a\}$ with heading ψ_a aligned with the path frame $\{p\}$ 92
9.2 A section of the net 94
9.3 Release mechanism and weight cell attached to a gimbaled suspension . 95
9.4 Screenshot from Neptus GCS during SITL, the fixed-wing UAV and the multirotor are illustrated as the green and white arrow respectively. . . . 96

10.1 Agdenes airfield. *Image courtesy of norskeflyplasser.no* 98
10.2 Simulated slave setup 99
10.3 Case 1: North-East positions, reference heading and surge 100
10.4 Case 1: Link distance expressed in the North-East plane $||\mathbf{z}_{1:2}||^2$ and the Down axis $|\mathbf{z}_3|$ respectively. 101
10.5 Case 2: Recovery setup 102
10.6 Case 2a: North-East overview over the recovery, where the green and the blue line is the path of the fixed-wing and the multirotor respectively. The black dotted lines represents the boundaries of the virtual-runway. The blue stars is the desired waypoints, connected with the solid black line. 103
10.7 Case 2a: The recovery state change timestamps during the operation symbolized as red stars along the fixed-wing path (black line), the blue circles is the position of the centroid at the specific timestamps 104

10.8 Case 2a: Along- and cross-track positions in the path-frame $\{p\}$, the boundaries of the virtual-runway is represented as the black-dotted line . 104
10.9 Case 2a: Along- and cross-track positions in the path-frame $\{p\}$ a short period before the impact 105
10.10 Case 2a: Multirotor velocities denoted in the virtual-runway frame $\{p\}$. 105
10.11 Case 2b: North-East overview over the recovery, where the green and the blue line is the path of the fixed-wing and the multirotor respectively. The black dotted lines represents the boundaries of the virtual-runway. The blue stars is the desired waypoints, connected with the solid black line. 106
10.12 Case 2b: Along- and cross-track positions in the path-frame $\{p\}$ a short period before the impact 107
10.13 Case 2b: Multirotor velocities denoted in the virtual-runway frame $\{p\}$. 107
10.14 Case 3a: Centroid body $\{\bar{c}\}$ velocities, comparing reference and desired values. 108
10.15 Case 3a: Link distance error $\tilde{z}$, expressed in the North-East plane and and Down axis respectively 109
10.16 Case 3a: North-East multirotors and centroid positions. 109
10.17 Case 3b: North-East positions and height 110
10.18 Case 3b: Link distance expressed in the North-East plane $\|\mathbf{z}_{1:2}\|^2$ and the Down axis $|\mathbf{z}_3|$ respectively 111
10.19 Case 3b: Centroid velocity and reference simulator states. 111

Abbreviations

UAV	=	Unmanned Aerial Vehicle
NTNU	=	Norges teknisk-naturvitenskapelige universitet (Norwegian University of Science and Technology)
DOF	=	Degrees Of Freedom
SITL	=	Software-In-The-Loop
GCS	=	Ground Control Station
LSTS	=	Laboratório de Sistemas e Tecnologias Subaquáticas (Underwater Systems and Technology Laboratory)
GLUED	=	GNU/Linux Uniform Environment Distribution
BBB	=	BeagleBone Black
DUNE	=	DUNE Uniform Navigational Environment
GUI	=	Graphical User Interface
IMC	=	Inter-Module Communication
IMU	=	Inertial Measurement Unit
GPS	=	Global Positioning System
GNSS	=	Global Navigation Satellite System
RTK	=	Real Time Kinematic
MTOW	=	Maximum Take-Off Weight
LOS	=	Line-Of-Sight
PID	=	Proportional-Integral-Derivate
CG	=	Center of Gravity
CP	=	Center of Pressure
UDP	=	User Datagram Protocol
TCP	=	Transmission Control Protocol
RK4	=	Runge-Kutta method of order 4
ETA	=	Estimated Time of Arrival
RC	=	Radio Control
ESC	=	Electronic Speed Control

Chapter 1

Introduction

1.1 Background and Motivation

Marine operations with small fixed-wing UAVs at sea often involve take-off and landing from an area with limited space. The take-off is often handled using a catapult or similar equipment. The landing on the other hand is often more complicated since it involves reducing the speed of the UAVs fast without destroying the equipment. The most popular approach uses a net attached to the end of the ship. However, it turns out to be hard to create robust autopilots for these landing. The environment out at sea is characterized as windy, along with a lot of ship movements, especially the heave motions are quite hard to handle. Given that the landing area on the ship is not very large, the margins are quite small and the high precession and accuracy are required.

A solution to these problems is to move the net away from the ship, this will make more room for landing, and the motion from the ship will not affect the landing maneuver. Therefore it is proposed to suspend a net between a group of multirotor UAVs such that the net is dynamic. Hence, it should be adaptable concerning the landing trajectory of the fixed-wing UAV. There are many advantages with such a solution. Firstly the relative velocity between the net and the fixed-wing UAV can be regulated, such that the amount of forces in the collision can be reduced. Further, changing the position of the net according to the fixed-wing will give a very adaptable system concerning abrupt changes in the environment.

1.2 Objectives and methods

In specific the coordination between the fixed-wing UAV and the multirotor UAVs will be addressed, and how the multirotors should maneuver in order to catch the fixed-wing safely in the net. Further a guidance method for safe transportation of the suspended payload both with and without the fixed-wing recovered into the net will be given. In order to lift a net, multiple multirotors should by used to distribute the force needed to hold the suspended net and fixed-wing. Therefore a coordination controller scheme is needed to keep the multirotor UAVs in a certain formation. The details and implementation of this is not within the scope of this report, however, a method to treat the multirotors as one body to simplify the guidance controllers will be considered. Lastly, some practical considerations regarding the recovery operation will be addressed.

This requires that the multi-body system is mathematically modeled for analyzing different control systems. Furthermore, the collision between the net and the fixed-wing UAV should be consider in order to find the physical requirements for the multirotors and the net. The total system with dynamics and control-algorithms should be simulated for further analysis, for prototyping the numerical computing environment Matlab [MATLAB, 2015] would be used.

For more extensive experiments the resulting control algorithms should be implemented in a software framework. The software implementation of the controllers should be tested using an external implementation of the dynamics of the fixed-wing UAV and the multirotor UAVs. Such a *Software-In-The-Loop* (SITL) setup will be able to validate the controller to some degree. Lastly, the total system will be verified by field experiments.

1.3 Previous work

Using a net suspended from multirotor UAVs to recover fixed-wing UAVs is a field lacking of research in the literature. However, multiple methods for autonomous fixed-wing UAVs recovery and landing exists. Many applications recover the fixed-wing in a static net such as in e.g. [Skulstad et al., 2015]. Moreover, a moving landing target is addressed in e.g. [Morais et al., 2015] where a vision based method for landing in a net on a moving ship is proposed. Some methods does not use a net, such as the SkyHook system developed by Insitu [Insitu.com, 2016]. Here the fixed-wing UAV ScanEagle developed by Insitu is recovered by hooking the wing tip to a vertical wire. The wire can be attached to a fixed structure, however, in [Ackerman, 2015] the wire has been attached to a multirotor and performed a successful recovery of the ScanEagle, this solution was also able to launch the fixed-wing from the multirotor.

A major challenge for the recovery is that a accurate and precise position system is required, either the recovery object is static or dynamics. In [Huh & Shim, 2009] the positioning system is based on a GPS (Global Positioning System) solution augmented with a vision-based algorithm to detect the recovery object when approaching. On the other hand, in e.g. [Kim et al., 2013] a system fully relied on vision-based positioning system during the recovery phase is presented. Moreover in [Skulstad et al., 2015] a fixed-wing UAV was recovered in a static net utilizing only a low-cost RTK (Real Time Kinematic) GPS solution as positioning system.

The maneuver requires a certain load to be suspended to multiple multirotor UAVs, this is a field of growing interest in the literature. [Bisgaard et al., 2010] and [Klausen et al., 2015] presents solutions to reduce the swinging motion of the load during slung load operations using a helicopter and multirotor respectively. Further, [Klausen et al., 2014] and [Bisgaard, 2008] discuss the mathematical modeling of multi-body system, in this case, suspended loads between multiple vehicles. The method proposed here is based on the Udwadia-Kalaba equation [Udwadia & Kalaba, 1992] where generalized coordinates are utilized, such that the constraint forces can be found explicitly.

The net has a detailed structure which in itself has quite complex dynamics, however, as net is extensively used in the fishing sector there exists rigorous numeric analysis of the dynamics as e.g. seen in [Li et al., 2006]. The external forces applied on the net such as drag is investigated in e.g. [Swift et al., 2006]. These methods is applied under water, but can be adapted to aerodynamically purposes.

Regarding the field of collision dynamics the literature is quite extensive as force dynamics is an important research field in the robot industry. Different contact force and impact models can be found in e.g. [I.I: Argatov, 2012], [Lankarani & Nikravesh, 1990], [Ravn, 1998] and [Chatterjee & Ruina, 1998]. They presents different approaches to model the dynamics of forces during a general collision.

Cooperative control of a set of bodies are introduced by among other [Bai et al., 2011], here a passivity-based method are used giving a general framework for different applications. In the Master's thesis of Jon-Håkon Bøe-Røli [Røli, 2015] this framework was used to implement a system for cooperative control for multirotors, where the viability was proven through field experiments.

1.4 Contributions

This thesis seeks to investigate the dynamic net recovery approach by looking into the control of multirotors. By incorporating existing control methods a fixed-wing UAV net recovery concept using multiple multirotors is presented.

The contribution of thesis can be subdivided into the following parts:

Multi-body dynamics: Mathematically model the coupled system consisting the multirotors and the suspended net, and the dynamics of the collision when the fixed-wing UAV is captured in the net.

Multi-body control: The multirotors should be controlled as one vehicle, hence the multirotor formation will be subjected to heading and translation control, assuming a multirotor formation controller is present.

Guidance: The multirotor formation should transport the suspended net along any desired path. The controller should be able to keep the net tensioned and avoid fluctuations.

Recovery: The fixed-wing should be instructed to follow a landing trajectory. The recovery system will ensure that the multirotor formation are able to initiate a maneuver in order to reach a desired rendezvous and capture the fixed-wing.

The main contribution to this thesis is the design and implementation of the total recovery system with multiple multirotors. Here the recovery, guidance and multi-body control methods are incorporated to present a total control system. Moreover, an implementation of the multi-body control architecture for field experiments has been stressed, hence practical implementation and usage of the system has also been a significantly contribution.

This thesis will not consider all aspects of the problem formulation, hence the following assumptions are made

- The attitude of multirotors are controlled, such that only translation motion of the multirotors are considered.
- A cooperative controller is present proven to render the multirotors to a desired formation.
- The fixed-wing UAV is instructed to follow a landing trajectory, hence fixed-wing dynamics and controllers will not be discussed.
- The UAV-Lab provides the necessary UAV platforms with the required payloads such that only a software implementation will be considered for field experiments.

It should be noted that the derivation of the multi-body dynamics together with the multi-body numerical simulation architecture[1] are based on work by Kristian Klausen as discussed in e.g. [Klausen et al., 2014]. Furthermore the work through this thesis and the pre-project during the autumn of 2015 has also contributed to a conference publication [Klausen et al., Submitted 2016] as presented on *The 2016 International Conference on Unmanned Aircraft Systems* (ICUAS'16).

1.5 Organization of the thesis

Firstly, the necessary background theory and notation will be briefly discussed in Chapter 2 followed by a description of the system architecture in Chapter 3 used for the experimental setup.

The mathematical model of the system will be derived together with the collision dynamics in Chapter 4. It should be noted that the derivation of the collision dynamics were a major part of the pre-project for this thesis, however, the model is restated here for completeness purposes.

The controller architecture follows in Chapter 5, Chapter 6 and Chapter 7. Here the multirotor velocity control, payload transport guidance and recovery maneuver are addressed respectively. For each of the controllers an analytical simulation study are presented. The controller are based on work through the pre-project and they are developed further and adapted to the cooperative multirotor control objective.

[1]The Matlab source code is available from the UAV-Lab GitLab server `https://uavlab.itk.ntnu.no/88` (requires authorized access)

The experimental setup, addressing the implementation of the control architecture on an embedded system as well as SITL simulation results follows in Chapter 8. For the purpose of practical usage of the recovery system some challenges and suggestions are discussed in Chapter 9.

Based on field experiments the most important results are presented in Chapter 10, followed by a discussion of the main results in Chapter 11 with some concluding remarks and suggestions for further work.

Lastly, the digital appendix included with this thesis contains illustration videos from some of the Matlab and SITL simulation, as well as the experimental results. The source code is not included, however, they can be obtained from the UAV-Lab Git server as well. On the other hand, a user manual for the DUNE source code is also attached in this digital appendix.

Chapter 2

Notation and background theory

2.1 Definitions

In Figure 2.1 a sketch of the platforms required in the operation are presented, further in Table 2.1 the notation for these bodies are defined.

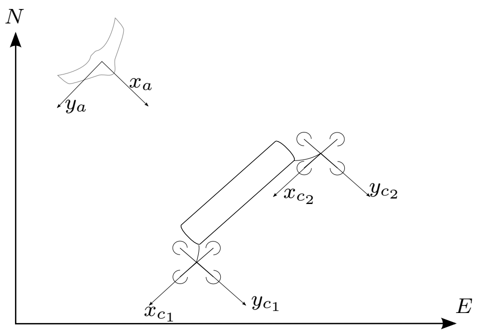


Figure 2.1: Sketch of the platforms in the North-East plane with the approaching fixed-wing airplane $\{a\}$ and the two multirotors $\{c_1\}$ and $\{c_2\}$ with a net suspended between them.

Body frame	Notation
Multirotor i	c_i
Centroid multirotor	$\bar{c}$[1]
Net (load)	l
Fixed-wing UAV	a

Table 2.1: Notation for the different body frames

[1]The centroid will be defined in Chapter 5

It is assumed that the distances between all the bodies in the recovery phase is sufficiently small enough to apply flat-earth approximation by a defining a common reference point as origin of a local tangent-plane North-East-Down (NED), noted as $\{n\}$ as seen in Figure 2.1. It should be noted that all frames are defined as right-hand systems such that for the frames in e.g. Figure 2.1 the z-axis are pointing into the paper (assuming the bodies are aligned with $\{n\}$). The multirotors are assumed to be attitude controlled where the control loops are assumed to be sufficiently fast enough such that the orientation of the multirotors can be neglected for the further analysis.

Notation

The notation as defined by SNAME in [The Society of Naval Architects and Marine Engineers (SNAME), 1950] are somehow modified to compensate for the multi-body approach presented in this work.

$$\mathbf{f}_j^i = \begin{bmatrix} f_{j,x}^i \\ f_{j,y}^i \\ f_{j,z}^i \end{bmatrix} \in \mathbb{R}^3 \quad \mathbf{m}_j^i = \begin{bmatrix} m_{j,x}^i \\ m_{j,y}^i \\ m_{j,z}^i \end{bmatrix} \in \mathbb{R}^3 \quad \mathbf{r}_{j/k}^i = \begin{bmatrix} r_{j/k,x}^i \\ r_{j/k,y}^i \\ r_{j/k,z}^i \end{bmatrix} \in \mathbb{R}^3 \quad \boldsymbol{\omega}_{j/k}^i = \begin{bmatrix} \omega_{j/k,x}^i \\ \omega_{j/k,y}^i \\ \omega_{j/k,z}^i \end{bmatrix} \in \mathbb{R}^3$$

where $\mathbf{f}_j^i$ and $\mathbf{m}_j^i$ gives the force and moment applied through and about the point o_j respectively, both expressed in the frame $\{i\}$ from Table 2.1. Mostly o_j denotes the origin of the body such that the notation will be used if the action point differs from this. Further $\mathbf{r}_{i/k}^i$ defines a vector defined from the point o_j with respect to $\{k\}$ and expressed in $\{i\}$ where $\mathbf{r}$ e.g. can be a position or velocity. Here the term k is excluded if $k = i$. For rotational motion and e.g. angular velocity $\boldsymbol{\omega}_{j/k}^i$ gives the angular velocity of the body $\{j\}$ with respect to $\{k\}$ and expressed in $\{i\}$.

Further the list in Table 2.1 is used to connect each parameter to its respective body, such that for any parameter the subscript notation $()_i$ is used, where i is one of the bodies in Table 2.1[2].

[2]The reader should note the difference between a element in the moment vector m_*^i and the mass of body i denoted as m_i.

Rotation matrices

The three principal rotation matrices are defined with the following notation based on the Euler angle representation roll (ϕ), pitch (θ) and yaw (ψ) as discussed in e.g. [Fossen, 2011].

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.1}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.2}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

Skew-symmetric matrix

The skew-symmetric matrix $\mathbf{S}(\boldsymbol{\lambda}) \in \mathbb{R}^{3\times3}$ where $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}^T$ defines the cross product between $\boldsymbol{\lambda}$ and $\mathbf{a} \in \mathbb{R}^3$ such that $\boldsymbol{\lambda} \times \mathbf{a} = \mathbf{S}(\boldsymbol{\lambda})\mathbf{a}$ giving the the following definition

$$\mathbf{S}(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix} \tag{2.4}$$

Diagonal matrix

A diagonal matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ can be constructed from a vector $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots a_n \end{bmatrix} \in \mathbb{R}^n$ holding the diagonal elements by the function diag(). Such that

$$\mathbf{A} = \text{diag}(\mathbf{a}) = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{bmatrix} \tag{2.5}$$

Moore-Penrose pseudoinverse

Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ the Moore-Penrose pseudoinverse $()^\dagger$ gives a generalized definition of the inverse matrix and is defined as follows

$$\mathbf{A}^\dagger = \begin{cases} \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T & \text{if} \quad \mathbf{A}^T\mathbf{A} \text{ is invertible (\textit{left inverse})} \\ \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1} & \text{if} \quad \mathbf{A}\mathbf{A}^T \text{ is invertible (\textit{right inverse})} \end{cases} \tag{2.6}$$

which gives the following properties

$$\begin{aligned} \mathbf{A}^\dagger\mathbf{A} = \mathbf{I} \quad &\text{if} \quad \mathbf{A}^T\mathbf{A} \text{ is invertible } (\textit{left inverse}) \\ \mathbf{A}\mathbf{A}^\dagger = \mathbf{I} \quad &\text{if} \quad \mathbf{A}\mathbf{A}^T \text{ is invertible } (\textit{right inverse}) \end{aligned}$$

Trace

The trace function $\mathrm{tr}(\mathbf{A})$ of an square matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is defined as the sum of all the elements on the diagonal, that is,

$$\mathrm{tr}(\mathbf{A}) = a_{11} + a_{22} + \cdots + a_{nn} = \sum_{i=1}^{n} a_{ii} \tag{2.7}$$

Nonlinear saturation function

The saturation function sat(x, x_{min}, x_{max}) saturate the signal x with respect to the upper and lower bound as given by x_{max} and x_{min} respectivly such that

$$\text{sat}(x, x_{min}, x_{max}) := \begin{cases} x_{max} & \text{if } x > x_{max} \\ x_{min} & \text{if } x < x_{min} \\ x & \text{else} \end{cases} \tag{2.8}$$

2.2 Classical mechanics

Newton's second law of motion from classical mechanics in linear Equation (2.9) and rotational motion Equation (2.10) giving the force $\mathbf{f}$ and moment $\mathbf{m}$ respectivly in inertial space is well known, and can be found in e.g. [Hugh D. Young & A. Freedman, 2011, Ch.8 Vol 1]

$$\mathbf{f} = m\mathbf{a} = \frac{d\mathbf{P}}{dt} \tag{2.9}$$

$$\mathbf{m} = \mathbf{I}\boldsymbol{\alpha} = \frac{d\mathbf{L}}{dt} \tag{2.10}$$

where m is the mass, $\mathbf{a}$ the acceleration, $\mathbf{I}$ the moment of inertia matrix and $\boldsymbol{\alpha}$ the angular acceleration of the body. This defines the linear momentum $\mathbf{P}$ and angular momentum $\mathbf{L}$ in Equation (2.11) and Equation (2.12) respectively.

$$\mathbf{P} = m\mathbf{v} \tag{2.11}$$

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega} = \mathbf{r} \times \mathbf{P} \tag{2.12}$$

where $\mathbf{v}$ and $\boldsymbol{\omega}$ defines the velocity and angular velocity respectively, here the vector $\mathbf{r}$ gives the location of the rotating object with respect to its axis of rotation[3].

[3]Note that the linear momentum $\mathbf{P}$ is presented in capital text to avoid confusion with the position vector $\mathbf{p}$.

Further the idealized linear impulse law is shown in Equation (2.13) by using Equation (2.9), and gives the impulse $\mathbf{J}$ for the sum of forces $\sum \mathbf{f}$ acting over the timespan from t_1 to t_2

$$\mathbf{J} = \int_{t_1}^{t_2} \sum \mathbf{f}\, dt = \int_{t_1}^{t_2} \frac{d\mathbf{P}}{dt}\, dt = \int_{P_1}^{P_2} d\mathbf{P} = \mathbf{P}_2 - \mathbf{P}_1 = \Delta \mathbf{P} \tag{2.13}$$

the same relation holds for angular momentum giving the angular impulse $\mathbf{H}$

$$\mathbf{H} = \int_{t_1}^{t_2} \sum \mathbf{m}\, dt = \int_{t_1}^{t_2} \frac{d\mathbf{L}}{dt}\, dt = \mathbf{L}_2 - \mathbf{L}_1 = \Delta \mathbf{L} \tag{2.14}$$

by assuming constant force $\sum \mathbf{f} = \bar{\mathbf{f}}$ during the impulse, Equation (2.13) can be simplified to

$$\mathbf{J} = \bar{\mathbf{f}} \Delta t \tag{2.15}$$

where $\Delta t := t_2 - t_1$ is defined as the duration of the collision.

Again by assuming constant torque $\sum \mathbf{m} = \bar{\mathbf{m}}$ during the impulse, Equation (2.14) can be simplified to

$$\mathbf{H} = \bar{\mathbf{m}} \Delta t \tag{2.16}$$

2.3 Aerodynamic drag

Each body in the system is influenced by the aerodynamic forces, these are defined in e.g. [Beard & McLain, 2012], here the aerodynamic drag forces are of most interest. The drag will act on the center of pressure (CP) of the body and are defined in Equation (2.17) expressed in the body frame $\{i\}$. Here ρ is the density of air in kg/m^3, $\mathbf{V}^i_{air} = \begin{bmatrix} V_{air,x} & V_{air,y} & V_{air,z} \end{bmatrix}^T$ is the airspeed in the body frame, $C_{D,i}$ are nondimensional coefficients and S_i is the reference surface which the coefficients $C_{D,i}$ are computed for.

$$\mathbf{f}^i_{CP} := \mathbf{f}^i_D = \frac{1}{2} \rho S_i C_{D,i} \begin{bmatrix} V^2_{air,x} & V^2_{air,y} & V^2_{air,z} \end{bmatrix}^T \tag{2.17}$$

The airspeed V^i_{air} is defined as the relative velocity between the ground velocity of the body $\mathbf{v}^i_{CP,i}$ and the wind velocity $\mathbf{v}^i_w$ such that $\mathbf{V}^i_{air} = \mathbf{v}^i_{CP,i} - \mathbf{v}^i_w = \begin{bmatrix} V_{air,x} & V_{air,y} & V_{air,z} \end{bmatrix}^T$. The introduction of wind as a external disturbance will give different orientations of the wind frame and the body frame, this will not be further addressed as the orientation of the multirotors will not be considered in this work. However, for more rigorious analysis of the drag effect on the net, this should be considered.

2.4 Rigid-body dynamics

For all vehicles the local frame $\{n\}$ is assumed to be inertial, such that Newton's law can be applied in the $\{n\}$ frame.

Kinematics

Using the SNAME definition as defined in e.g. [Fossen, 2011] the following notation will be utilized

$$\boldsymbol{\eta}_b = \begin{bmatrix} \mathbf{p}^n_{b/n} \\ \Theta_{nb} \end{bmatrix} \qquad \mathbf{p}^n_{b/n} = \begin{bmatrix} x^n & y^n & z^n \end{bmatrix}^T \qquad \boldsymbol{\Theta}_{nb} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$$

$$\boldsymbol{\nu}^b = \begin{bmatrix} \mathbf{v}^b_{b/n} \\ \omega^n_b \end{bmatrix} \qquad \mathbf{v}^b_{b/n} = \begin{bmatrix} u & v & w \end{bmatrix}^T \qquad \omega^b_{b/n} = \begin{bmatrix} p & q & r \end{bmatrix}^T$$

The Euler angles $\boldsymbol{\Theta}_b$ is used to express the rotation of the body given the rotation sequence zyx which gives the rotation matrix $\mathbf{R}^n_b$ which rotates the frame $\{b\}$ to $\{n\}$

$$\mathbf{R}^n_b = \mathbf{R}^n_b(\boldsymbol{\Theta}) := \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{2.18}$$

then the following relation holds

$$\dot{\boldsymbol{\eta}}_b = \mathbf{J}_\Theta \boldsymbol{\nu}_b \tag{2.19}$$

where

$$\mathbf{J}_\Theta = \begin{bmatrix} \mathbf{R}^n_b(\boldsymbol{\Theta}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{T}_\Theta \end{bmatrix} \tag{2.20}$$

here $\mathbf{T}_\Theta$ is defined as follows (see e.g. [Egeland & Gravdahl, 2002] for details)

$$\mathbf{T}_\Theta := \begin{bmatrix} 1 & \sin(\phi)\sin(\theta)/\cos(\theta) & \cos(\phi)\sin(\theta)/\cos(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)/\cos(\theta) & \cos(\phi)/\cos(\theta) \end{bmatrix} \tag{2.21}$$

it should be noted that $\mathbf{T}_\Theta$ is not defined for $\theta = \pm\frac{\pi}{2}$, however, for both the multirotor and the net such orientation is not within the scope of this operation. Therefore the assumption that $\theta \neq \pm\frac{\pi}{2}$ is valid.

Further, in order to transform inbetween the different notations the derivative of Equation (2.20) is derived

$$\dot{\mathbf{J}}_\Theta = \begin{bmatrix} \mathbf{S}(\boldsymbol{\omega}^b_{b/n})\mathbf{R}^n_b(\boldsymbol{\Theta}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \dot{\mathbf{T}}_\Theta \end{bmatrix} \tag{2.22}$$

Kinetics

Using Newton's law of motion discussed in e.g. [Fossen, 2011] the following system can be derived

$$\mathbf{M}_b\dot{\boldsymbol{\eta}}_b + \mathbf{C}_b(\boldsymbol{\nu}^b)\boldsymbol{\nu}^b + \mathbf{g}_b(\boldsymbol{\eta}_b) = \boldsymbol{\tau}_b \tag{2.23}$$

Given the mass m_b of the body and the moment of inertia about the center of gravity of the body $\mathbf{I}_b$ the mass matrix $\mathbf{M}_b$ is defined as follows

$$\mathbf{M}_b = \begin{bmatrix} m_b\mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_b \end{bmatrix} \tag{2.24}$$

further the coriolis matrix $\mathbf{C}_b(\boldsymbol{\nu}^b)$ is defined

$$\mathbf{C}_b(\boldsymbol{\nu}^b) = \begin{bmatrix} m_b\mathbf{S}(\mathbf{v}^b_{b/n}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -\mathbf{S}(\mathbf{I}_b\omega^b_{b/n}) \end{bmatrix} \tag{2.25}$$

given the gravitational constant g the gravitational field vector in $\{n\}$ is defined as $\mathbf{g}^n = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$. As the dynamics are given in the body-frame, the vector field is transformed to $\{b\}$, giving $\mathbf{g}_b(\boldsymbol{\eta}_b)$

$$\mathbf{g}_b(\boldsymbol{\eta}_b) = -\begin{bmatrix} m_b\mathbf{R}^b_n\mathbf{g}^n \\ \mathbf{0}_{3\times1} \end{bmatrix} \tag{2.26}$$

Then the rigid-body dynamics are summarized as follows

$$\dot{\boldsymbol{\eta}}_b = \mathbf{J}_\Theta\boldsymbol{\eta}_b \tag{2.27}$$

$$\mathbf{M}_b\dot{\boldsymbol{\nu}}^b + \mathbf{C}_b(\boldsymbol{\nu}^b)\boldsymbol{\nu}^b + \mathbf{g}_b(\boldsymbol{\eta}_b) = \boldsymbol{\tau}_b \tag{2.28}$$

Chapter 3

System

This chapter will present the architecture of the system used for simulation studies and field experiments. Firstly, an overview of the total system architecture will be given, then the different platforms will be presented followed by a short introduction to the payload and software used on the platforms.

3.1 Overview

In order to fulfill the desired operation the different vehicles used in the system needs a custom payload which should be able to determine the different states of the vehicles and communicate between the others. Further in order to control the total system pilots with Radio Control (RC) systems responsible for take-off and landing of the vehicles are necessary. Furthermore, they should also be able to abort the mission in case of any undesirable behavior of the system. Moreover the operation should be monitored and controlled from a Ground Control Station (GCS) by a operator familiar with the autonomous maneuver. The positions and velocities of the vehicles are estimated using a high-precision relative navigation method which requires a base station.

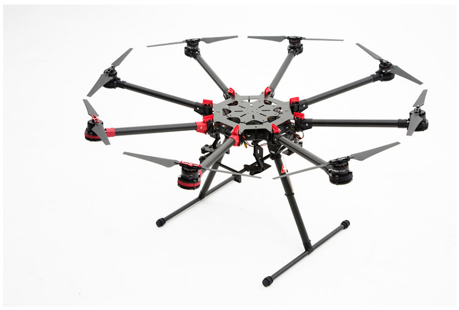
3.2 Platforms

The autonomous net recovery operation requires a small fixed-wing UAV, multirotors and a suspended net.

The X-8 Flying Wing from the Skywalker Technology Co. as seen in Figure 3.1a has been used widely at the UAV-lab[1] at NTNU for small fixed-wing UAV operations (e.g. [Skulstad et al., 2015]) and would be the desired fixed-wing to recover. It is equipped with two control-surfaces as well as a rotor attached behind the fuselage for propulsion.

To lift all the equipment a powerful and robust multirotor-platform is needed. Therefore an octacopter configuration, with eight arms and rotors was considered. The S1000+ from Da-Jiang Innovations Science and Technology Co., Ltd. (DJI) as seen in Figure 3.1b was selected for its ability to carry heavy payloads and its compact octacopter configuration. For the initial experiments a more compact and lighter multirotor with a hexarotor configuration from 3DR Robotics [3DRobotics, 2015] as seen in Figure 3.1c was selected due to its successful usage in multirotor experiments at the NTNU UAV-lab.

(a) Skywalker X-8 Flying Wing. *Image courtesy of itk.ntnu.no*


(b) DJI S1000+. *Image courtesy of dji.com*

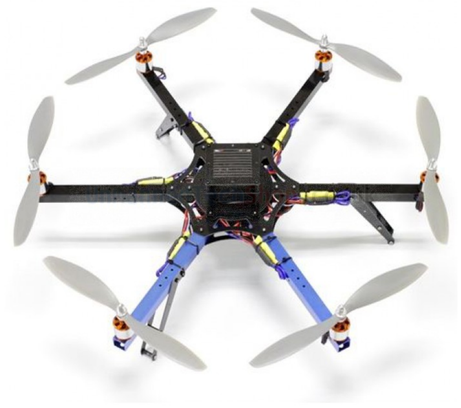

(c) 3DR Robotics Hexa-b. *Image courtesy of 3dr.com*

Figure 3.1: The vehicle platforms

[1]see the homepage of the organization `http://www.itk.ntnu.no/english/lab/unmanned` for an introduction

3.3 Software

This section will briefly introduce the most important software components in the system. In this work the software would mostly be considered, and will eventually be deployed on the hardware. However, the hardware will also be addressed briefly for completeness of the architecture presentation.

3.3.1 LSTS Software Toolchain

The Laboratóro de Sistemase Tecnologias Subaquáticas (Underwater Systems and Technology Laboratory) (LSTS) has developed an open-source software toolchain specialized for unmanned vehicles[2]. The toolchain consist among others of the software-framework DUNE, the communication protocol IMC, the ground station control software Neptus and a GNU/Linux Uniform Environment Distribution (GLUED) for embedded computers.

DUNE

DUNE Unified Navigation Environment (DUNE) is an on-board software framework written in C++ independent of both CPU architecture and operating system. The system is responsible for all interaction with the peripherals, supervision of the vehicle and communication among others. In the framework a complete open-source guidance, navigation and control scheme are already implemented. The basic idea is to divide the system into smaller tasks each running as separate processes. The tasks communicate with each other using the concept of message passing, where each task choose to listen for certain messages and sends others.

Neptus

Neptus is a distributed ground station software. The system provides a Graphical User Interface (GUI) to the operator were all vehicles can be controlled and monitored. The GUI consists of among others a map with real-time data from the vehicles, mission planning and reviewing of missions.

IMC

The Inter-Module Communication (IMC) is a message communication protocol. The protocol defines all messages dispatched to the communication-bus between all DUNE tasks both internally in the vehicle and between different vehicles, it also defines the communication interface to the ground software Neptus.

[2]see [LSTS, 2015] for information about the LSTS toolchain, the source code is available from GitHub at `https://github.com/LSTS`

3.3.2 ArduPilot software

ArduPilot is an open-source autopilot system developed by the DIY Drones community[3]. The system supports multiple platforms such as multirotors and airplanes which gives the possibility to control the vehicles using different autopilot solutions. For this work the multirotor and airplane autopilot known as the APM:ArduCopter and APM:ArduPlane autopilot respectively were used. Ground station software for mission planning and monitoring of the vehicles is provided, interfaced by the Micro Air Vehicle Link (MAVLink) communication protocol. The MAVLink protocol also makes it possible to interface the ArduPilot software with DUNE for lower-level control. A wide range of autopilot boards are provided for the system, among other the Pixhawk autopilot.

In Figure 3.2 the overview of the software architecture is presented. Here the connection between the DUNE and ArduPilot system is illustrated.

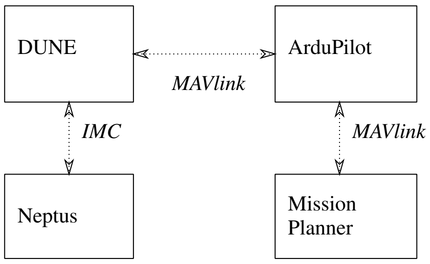


Figure 3.2: Software architecture, the different software systems are illustrated as the boxes, and the application protocols are given in the *italic text* over the dotted lines.

The benefits with this setup are not only that the ArduPilot autopilot can do the lower level control such as controlling the attitude of the multirotor. As the ArduPilot presents a total autopilot solution this is a safe backup system to use when developing new control system in the DUNE framework. As it will be discussed in Section 3.4 the two systems will also run on different hardware platforms giving the operator of the UAV the necessary fail-safe system, as well as giving the developer the possibility to implement higher level control scheme. Lastly the DUNE system can be monitored and controlled remotely utilizing a IMC message link from the Neptus GCS.

[3]ArduPilot source code available from GitHub at `https://github.com/diydrones/ardupilot`

3.4 Hardware

In Figure 3.3, a schematic over the hardware components required. Here the hardware embedded in the airframes are shown, as well as the *Base Station* equipped with a *Global Navigation Satellite System* (GNSS) antenna and a computer utilized as a common earth-fixed reference point for the moving airframes. Lastly the *Mission Control* segment gives the possibility to remotely control and monitor the airframes using a GCS typically deployed on a laptop computer and RC control handled by the pilot.

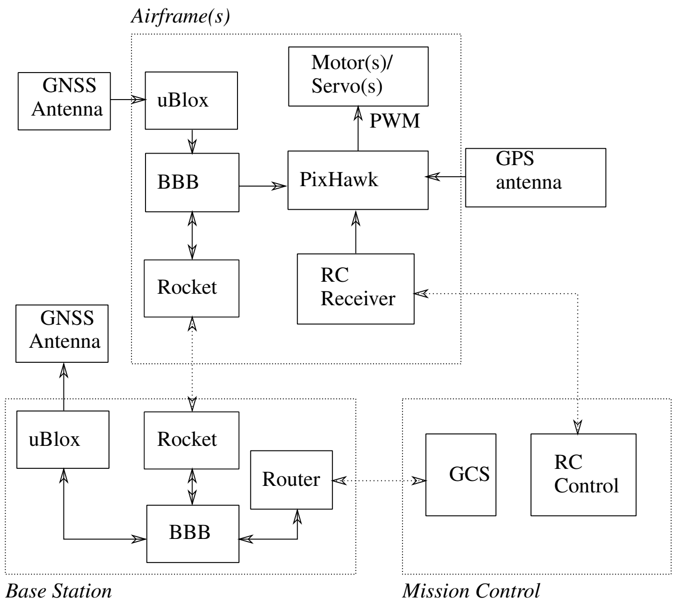


Figure 3.3: Hardware architecture, the dotted lines represents wireless communication

3.4.1 Autopilot

The autopilot system sets the desired velocities of the rotors for the multirotors and the control surfaces and the motor for the fixed-wing UAV. The Pixhawk autopilot designed by the open-source community PX4 and delivered by 3DR robotics [3DR, 2015a] is used for the lower-level control of the vehicles. This autopilot system runs the ArduPilot software and includes sensors for navigation. Internally there is an Inertial Measurement Unit (IMU) for measuring the rotation and acceleration and a barometer to measure relative altitude. An external Global Positioning System (GPS) and compass module [3DR, 2015c] is required for more precise navigation. The Pixhawk also receives commands from a RC operated by the pilot giving manual control[4] of the vehicles. Lastly the desired angular

[4]Full manual control of a multirotor is impossible, and a computer must always be in the loop, controlling the orientation by setting the individual rotor speeds

velocities of the multirotors are sent to each Electronic Speed Control (ESC) using Pulse Width Modulation (PWM). The ESCs are connected to each individual motor and controls the speed of the rotor.

3.4.2 Navigation

The standalone GPS solution presented by the Pixhawk with its external antenna is not accurate enough for cooperative operations. Therefore a differential method is used in this thesis, where an earth-fixed base station are placed in the area of operation giving corrections to all vehicles in the operation. In the cooperative experiments the *Real Time Kinematic* (RTK) GNSS method is used. Here the satellites carrier wave are utilized and the phase shift between the base station and the vehicle are calculated. In this setup satellite data is received from the RTK GNSS module uBlox [u-Blox, 2016]. The method will not be discussed further in this work, but the reader are referred to [Sørbø, 2016] for more details around the RTK setup.

3.4.3 Computer

The embedded computer is responsible for running the DUNE framework on the GLUED operating system, interfacing all the external sensors and the communication with the autopilot system. The BeagleBone Black (BBB) [BeagleBoard, 2015], a small, low-cost platform is used as the embedded payload computer. The BBB is connected over a serial port to the Pixhawk system which is responsible for the low-level control, further it communicates wirelessly with the base station and the mission control. Furthermore, the navigation algorithms on the base station are also deployed on the BBB.

3.4.4 Communication

The platforms used communicate wireless between each others. The main communication using DUNE IMC messages are performed utilizing a 5.8 GHz wireless link with the Rocket M5 solution from Ubiquity Networks [Ubiquiti-Networks, 2015], here a constant transfer delay are ensured using a *Time Division Multiple Access* medium control. This transfer medium handles all the wireless communication for among others telemetry and base station control. This data is transfered using the *User Datagram Protocol* (UDP) transport protocol, as the loss of a single packet is not critical for the performance. Furthermore, the RTK solution are also transfered wireless from the base station to the vehicles using the Rocket M5, however, this requires a high throughput and validity of data between the base station, therefore the *Transmission Control Protocol* TCP transport protocol are utilized here. Lastly, for the purpose of accessing the Pixhawk a simple 433 MHz link from the ArduPilot mission planner is created using a 3DR radio [3DR, 2015b].

3.4.5 Suspended payload sensors

The dynamics of the suspended net will be addressed and measurements of the angle and weight of the load are needed.

In order to measure the weight of the suspended load a load cell will be attached between the multirotor and the suspended net measuring the relative load weight. In Figure 3.4 the load cell of the type Futek LSB200 [Futek, 2015] can be seen, here it is suspended to a frame and has a load attached with ropes. This setup illustrate how the load cell can be used for the purpose of measuring the tension in the net. Further, in order to interface the load cell to the BBB a interface board from Phidget, PhidgetBridge [Phidgets, 2015] was utilized to convert the analog load cell signals to digital signals. Then the PhidgetBridge was connected to the Universal Serial Bus (USB) interface of the BBB.

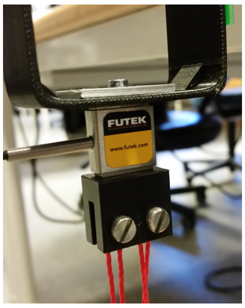


Figure 3.4: Load cell

Further it will be necessary to measure the angle of the net relative to the multirotor. As it will be discussed in Chapter 5 the measurement of the pitch and roll angle of the load is required to compensate for the load tension, and will be measured using a gimbaled setup similar to the picture in Figure 3.5. Here two rotary encoder of the type Piher MTS-360 encoder [Piher, 2015] is attached which measures the two required angles. With the load cell attached to the bottom of the gimbaled setup the tension in the net and angles will be determined.

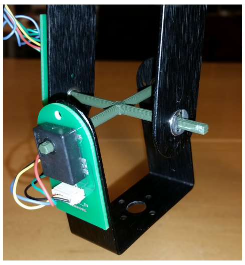


Figure 3.5: Angular encoder

Chapter 4

The multi-body dynamics

For the purpose of analyzing the multi-body coupled system the dynamics must be derived. The following chapter seeks to obtain the equations of motions in order to simulate the constrained dynamics, as well as a less rigorous analysis aiming to extract the most significant dynamics intended for control analysis. Also, the dynamics followed by the collision between the fixed-wing UAV and the suspended net will be derived. The dynamics of the fixed-wing UAV will not be considered as it is only treated as an external object actuating as well as adapting the dynamics of the suspended net during the impact, moreover, this thesis will not look into the matter of controlling the fixed-wing.

4.1 Unconstrained body dynamics

In general one must find the unconstrained dynamics for each body in the coupled multi-body system. This is required in order to state the general unconstrained dynamics $\ddot{\mathbf{q}}_u$ for the overall system as discussed in Appendix A.1.1. The coupled system consist of the net l suspended by the k multirotors using p wires. The dynamics of the bodies will be derived in the body frame $\{b\}$ and on the same form as in Section 2.4.

4.1.1 Multi-rotor dynamics

The total dynamics for multirotor i is given as

$$\dot{\boldsymbol{\eta}}_{c_i} = \mathbf{J}_\Theta \boldsymbol{\eta}_{c_i} \tag{4.1}$$

$$\mathbf{M}_{c_i}\dot{\boldsymbol{\nu}}^{c_i} + \mathbf{C}_{c_i}(\boldsymbol{\nu}^{c_i})\boldsymbol{\nu}^{c_i} + \mathbf{D}_{c_i}\boldsymbol{\nu}^{c_i} + \mathbf{g}_{c_i}(\boldsymbol{\eta}_{c_i}) = \boldsymbol{\tau}_{c_i} \tag{4.2}$$

where $\boldsymbol{\eta}_{c_i} = \begin{bmatrix}\mathbf{p}^n_{c_i/n} \boldsymbol{\Theta}_{nc_i}\end{bmatrix}$ gives the position vector $\mathbf{p}^n_{c_i/n}$ and the orientation $\boldsymbol{\Theta}_{nc_i} = \begin{bmatrix}\phi_i & \theta_i & \psi_i\end{bmatrix}^T$ as illustrated in Figure 4.1.

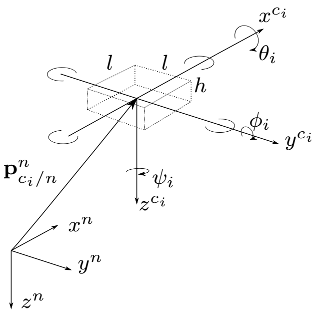


Figure 4.1: Multirotor body frame $\{c_i\}$ definition

The moment of inertia $\mathbf{I}_{c_i}$ is found by simplifying the geometry of the multirotor as a box with height h lengths l giving a total volume of $h \times l^2$, then it can be shown that

$$\mathbf{I}_{c_i} = \frac{1}{12} m_{c_i} \begin{bmatrix} 3l^2 + h^2 & 0 & 0 \\ 0 & 3l^2 + h^2 & 0 \\ 0 & 0 & l^2 \end{bmatrix} \tag{4.3}$$

further the multirotor is affected by translational aerodynamic drag during linear movements. These forces is introduced in Section 2.3, however, the relation between velocity and force is assumed to be linear as the multirotor will not operate in the velocity range where the non-linear effects is significant. Furthermore, the center of pressure (CP) is assumed to be located in the center of gravity (CG) such that the forces will be applied in CG. Then it follows that the damping matrix $\mathbf{D}_{c_i}$ can be defined as

$$\mathbf{D}_{c_i} = \begin{bmatrix} \mathbf{D}_{c,t} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{4.4}$$

where the translational aerodynamic drag matrix $\mathbf{D}_t$ is given as

$$\mathbf{D}_{c,t} = k_{t,c} \mathbf{I}_{3\times3} \tag{4.5}$$

where $k_{t,c}$ is a factor determining the relation between the aerodynamic drag and the velocity as explained in Section 2.3.

The external forces and moments $\boldsymbol{\tau}_{c_i}$ applied on the multirotor will now mainly consist of the control forces applied by the rotating propellers. The relation between $\boldsymbol{\tau}_{c_i}$ and the angular velocity of the propellers is dependent on the chosen multirotor configuration. Furthermore, as the multirotor attitude control is not within the scope of this thesis, the relation will not be derived here. The matter of control allocation and attitude control will not be discussed, and the reader is referred to e.g. [Mahony et al., 2012] for example of such control for a quadrotor configuration.

4.1.2 Net dynamics

The total dynamics for the net l are given on the same form as the multirotor

$$\dot{\boldsymbol{\eta}}_l = \mathbf{J}_\Theta \boldsymbol{\eta}_l \tag{4.6}$$

$$\mathbf{M}_l \dot{\boldsymbol{\nu}}^l + \mathbf{C}_l(\boldsymbol{\nu}^l)\boldsymbol{\nu}^l + \mathbf{D}_l \boldsymbol{\nu}^l + \mathbf{g}_l(\boldsymbol{\eta}_l) = \boldsymbol{\tau}_l \tag{4.7}$$

where $\boldsymbol{\eta}_l = \begin{bmatrix} \mathbf{p}^n_{l/n} & \boldsymbol{\Theta}_{nl} \end{bmatrix}$ gives the position vector $\mathbf{p}^n_{l/n}$ and the orientation $\boldsymbol{\Theta}_{nl} = \begin{bmatrix} \phi_l & \theta_l & \psi_l \end{bmatrix}^T$ as illustrated in Figure 4.2.

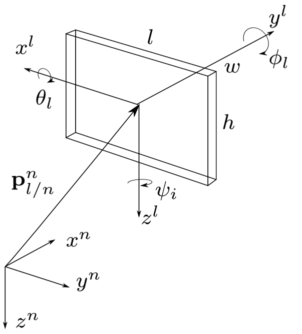


Figure 4.2: Net body frame $\{l\}$ definition

The geometry of the net is in a similiar fashion simplified to a box with height h, length l and width w such that the moment of inertia $\mathbf{I}_l = m_l \mathbf{I}^*_l$ can be found as

$$\mathbf{I}_l = \frac{1}{12} m_l \begin{bmatrix} w^2 + h^2 & 0 & 0 \\ 0 & l^2 + h^2 & 0 \\ 0 & 0 & w^2 + l^2 \end{bmatrix} \tag{4.8}$$

as the multirotor the net is affected by aerodynamic drag working in the center of pressure (CP) of the net. As no rigorous analysis of the aerodynamic properties net has been performed the location of CP is assumed to coincide with the CG. However, the net will also be subjected to rotational damping around CG as the surface area of the net is significant compared to the multirotor. Then it follows that the damping matrix $\mathbf{D}_l$ is on the following form

$$\mathbf{D}_l = \begin{bmatrix} \mathbf{D}_{l,t} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{D}_r \end{bmatrix} \tag{4.9}$$

where the translational $\mathbf{D}_{l,t}$ and rotational $\mathbf{D}_{l,r}$ aerodynamic drag matrix is given as

$$\mathbf{D}_{l,t} = k_{t,l} \mathbf{I}_{3\times3} \tag{4.10}$$

$$\mathbf{D}_{l,r} = k_{r,l} \mathbf{I}_{3\times3} \tag{4.11}$$

where $k_{t,l}$ is a factor determining the relation between the aerodynamic drag and the velocity as explained in Section 2.3, $k_{r,l}$ gives the relation between the damping moments and angular velocities around CG.

Now, the external forces and moments $\boldsymbol{\tau}_l$ applied on the net will mainly consist of the forces and moments applied by the fixed-wing UAV during the impact. These will be discussed in detail in Section 4.2.

4.2 Impact dynamics

The net and the fixed-wing UAV are the bodies which interacts directly during the collision. It is assumed that there is no significantly external forces and torques applied to the coupled system during the collision such that the linear and angular momentum are conserved. Further as the fixed-wing will stick to the net after the collision it is assumed that the collision is perfectly inelastic. Then the methods as derived in Appendix A.2 can be used to calculate the average collision force and moment applied on the net during the collision. Based on Equations (A.17)–(A.18) the following expression can be found

$$\bar{\mathbf{f}}_l = \frac{1}{\Delta t}\left(m_a \mathbf{v}_{a,-} - m_l \mathbf{v}_{l,-}\right) \tag{4.12}$$

$$\bar{\mathbf{m}}_l = \frac{1}{\Delta t}\left(\mathbf{I}_a \boldsymbol{\omega}_{a,-} - \mathbf{I}_l \boldsymbol{\omega}_{l,-}\right) \tag{4.13}$$

where all velocites $\mathbf{v}$ and angular velocites $\boldsymbol{\omega}$ is given in the net frame $\{l\}$, in total $\tau_{\text{coll}} = \begin{bmatrix}\bar{\mathbf{f}}_l & \bar{\mathbf{m}}_l\end{bmatrix}^T$. In Figure 4.3 the force applied on the net by the fixed-wing UAV is illustrated.

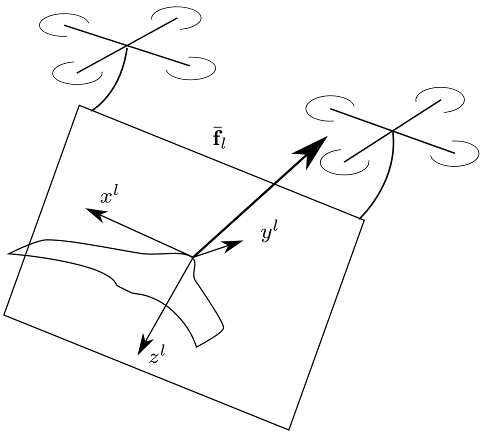


Figure 4.3: Average collision force $\bar{\mathbf{f}}_l$ during impact on the suspended net, assuming the fixed-wing UAV hits the CG of the net.

During the collision the dynamics of the net changes abruptly; the mass change simultanously as high external forces are applied. As the dynamics of the changes in mass is quite troublesome to find analytically it is assumed for simplicity that the mass of the incoming fixed-wing UAV adds to the mass of the net as a ramp from the start of the impact t_- to the end of the impact $t_- + \Delta t$. Then the piecewise linear mass function of the net is defined as follows

$$m(t) = \begin{cases} m_l & \text{if } t < t^- \\ m_l + \frac{m_a}{\Delta t}(t - t_-) & \text{if } t \in [t^-, t^+] \\ m_l + m_a & \text{if } t > t^+ \end{cases} \tag{4.14}$$

and the derivative

$$\dot{m}(t) = \begin{cases} 0 & \text{if } t < t^- \\ \frac{m_a}{\Delta t}(t - t_-) & \text{if } t \in [t^-, t^+] \\ 0 & \text{if } t > t^+ \end{cases} \tag{4.15}$$

Then by applying the Euler-Lagrangian method for the dynamics of the net as discussed in [Sagatun & Fossen, 1991] and summarized in [Fossen, 2011] the equations of motion of the net can be redefined as follows [1]. The derivation of the equations is given in Appendix A.3.

$$\mathbf{M}(t)\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu}, t)\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}, t) = \mathbf{0} \qquad \text{if } t \notin [t^-, t^+] \tag{4.16}$$

$$\mathbf{M}(t)\dot{\boldsymbol{\nu}} + \dot{\mathbf{M}}(t)\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu}, t)\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}, t) = \tau_{\text{coll}} \qquad \text{if } t \in [t^-, t^+] \tag{4.17}$$

such that the time-differentiated mass-matrix during the collision is found as

$$\dot{\mathbf{M}}(t) = \dot{m}(t) \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_l^* \end{bmatrix} \tag{4.18}$$

this follows from the assumption that the mass is the only time-dependent variable. The geometry of the net is assumed to be the same after the collision. Furthermore, the collision is assumed to be in the center of the net. This will in general not be true, however, the purpose with the analysis is to focus on the forces on the net which is assumed to be rigid, and the moments applied will be damped out in the net as it is able to flex. Therefore, a rigorous analyzis regarding the moments applied on the net will not be considered.

As seen in Section 2.4 and the definitions in Equations (2.24)–(2.26) of the mass $\mathbf{M}$, coriolis $\mathbf{C}$ and gravity $\mathbf{g}$ respectively the partly continuous dynamics can be defined by different masses and the inertia matrix as seen above.

4.3 Multi-body constrained dynamics for simulation

The following sections presents the derivation of the multi-body constrained dynamics with a suspended load from multiple multirotor agents as given in [Klausen et al., 2014], the method is adapted to the current system, and the derivation is given here for notation and completeness purposes.

[1] Note that the sub- and superscript l is left behind for ease of notation

In order to use the Udwadia-Kalaba (UK) equations as introduced in [Udwadia & Kalaba, 1992] and summarized in Appendix A.1.1 on the multi-body constrained system the unconstrained systems should be given on the generalized form

$$\mathbf{M}\ddot{\mathbf{q}}_u = \mathbf{Q} \tag{4.19}$$

As the rigid body dynamics is differentiated in the body frame $\{b\}$ it can not be used directly. Restating Equation (2.23) from Section 2.4 by collecting all external forces and moments into $\boldsymbol{\tau}_{RB}$ gives

$$\mathbf{M}_b\dot{\boldsymbol{\nu}}^b + \mathbf{C}_b(\boldsymbol{\nu}^b)\boldsymbol{\nu}^b = \boldsymbol{\tau}_{RB} \tag{4.20}$$

However, Newton's law of motion can be stated directly to the body frame by using the body-fixed acceleration $\mathbf{a}_b^b$ and $\boldsymbol{\alpha}_b^b$ angular acceleration such that

$$\mathbf{M}_b \begin{bmatrix} \mathbf{a}_b^b \\ \boldsymbol{\alpha}_b^b \end{bmatrix} = \boldsymbol{\tau}_{RB} \tag{4.21}$$

then $\begin{bmatrix} \mathbf{a}_b^b & \boldsymbol{\alpha}_b^b \end{bmatrix}^T$ can be used as the generalized unconstrained acceleration for each body. That gives the total unconstrained acceleration $\ddot{\mathbf{q}}_u$ for the total system

$$\ddot{\mathbf{q}}_u = \begin{bmatrix} \mathbf{a}_{c_1}^{c_1} & \boldsymbol{\alpha}_{c_1}^{c_1} & \cdots & \mathbf{a}_{c_k}^{c_k} & \boldsymbol{\alpha}_{c_k}^{c_k} & \mathbf{a}_l^l & \boldsymbol{\alpha}_l^l & \end{bmatrix}^T \tag{4.22}$$

In order to express the constraints on the form Equation (A.2) the generalized constrained coordinate $\mathbf{q}$ is defined

$$\mathbf{q} = \begin{bmatrix} \mathbf{p}_{c_1}^n & \Theta_{nc_1} & \cdots & \mathbf{p}_{c_k}^n & \Theta_{nc_k} & \mathbf{p}_l^n & \Theta_{ll} \end{bmatrix}^T$$

where the first k $\mathbf{p}_{c_i}^n$ and Θ_{nc_i} defines the position and orientation for multirotor i. The last elements $()_l$ defines the load position and orientation.

The wire i from the load to the multirotor j can be expressed as a vector

$$\mathbf{L}_i = \mathbf{p}_j - \mathbf{p}_L \tag{4.23}$$

where $\mathbf{p}_j$ and $\mathbf{p}_L$ are the attachment points of the wire to the multirotor and suspended net in $\{n\}$. Hence the wire-vector is a function of the generalized constrained coordinate $\mathbf{q}$. The wire vector is illustrated in Figure 4.4 for a relevant multirotor configuration with the suspended net.

By defining the length of wire i as d_i the constraint i can be given as

$$g_i := ||\mathbf{L}_i||^2 - d_i^2 = 0 \tag{4.24}$$

Now, g_j can be differentiated in order to get the constraints on the form as discussed in Appendix A.1.1.

$$\frac{dg_i}{dt} = 2\dot{\mathbf{L}}_i^T\mathbf{L}_i = 0 \tag{4.25}$$

$$\frac{d^2g_i}{dt^2} = 2\ddot{\mathbf{L}}_i^T\mathbf{L}_i + 2\dot{\mathbf{L}}_i^T\dot{\mathbf{L}}_i = 0 \tag{4.26}$$

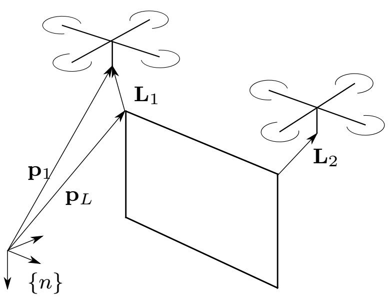


Figure 4.4: Suspended net from two multirotors, illustrating the wire vector $\mathbf{L}_i$ from the suspended load to the multirotor i wire attachment points.

It should be noted that the last constraint Equation (4.26) is a function of the generalized acceleration, that is $\frac{d^2 g_i}{dt^2} = f(\ddot{\mathbf{q}})$ such that this equation can be translated into $\mathbf{A}_i\ddot{\mathbf{q}} = \mathbf{b}_i$. Stacking the constraints from the p wires together gives

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_i \\ \vdots \\ \mathbf{A}_p \end{bmatrix}_{\in\mathbb{R}^{p\times n}} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_i \\ \vdots \\ b_p \end{bmatrix}_{\in\mathbb{R}^{p\times 1}} \tag{4.27}$$

The full derivation of the constraint equations will not be given here, but the reader are referred to [Klausen et al., 2014] for an example on how to proceed from the definition of the wire vector $\mathbf{L}_i$ to get the constraint Equation (4.24) on the standard form Equation (A.2).

In order to find the constrained dynamics, one must stack the total unconstrained system on the generalized form

$$\mathbf{M}\ddot{\mathbf{q}}_u = \mathbf{Q} \tag{4.28}$$

where $\mathbf{Q}$ is the generalized forces for all bodies and $\mathbf{M}$ is a diagonal concatenation of the individual mass matrices. The unconstrained accelerations $\ddot{\mathbf{q}}_u$ is on the same form as the constrained accelerations $\ddot{\mathbf{q}}$. The constrained dynamics can then be found using the solution as derived in Appendix A.1.1 and summarized in Equation (A.5), by inserting $\ddot{\mathbf{q}}_u = \mathbf{M}^{-1}\mathbf{Q}$. The expression is restated here

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_u + \mathbf{M}^{-\frac{1}{2}}\left(\mathbf{A}\mathbf{M}^{-\frac{1}{2}}\right)^{\dagger}(\mathbf{b} - \mathbf{A}\ddot{\mathbf{q}}_u)$$

However, as $\ddot{\mathbf{q}}$ is not expressed in the body-differentiated form it must be transformed back. It can be shown that the total body differentiated acceleration for all bodies $\dot{\boldsymbol{\nu}}$ is

$$\dot{\boldsymbol{\nu}} = -\mathbf{M}^{-1}\mathbf{C}(\boldsymbol{\nu}) + \ddot{\mathbf{q}} \tag{4.29}$$

by inserting the expression for the constrained accelerations $\ddot{\mathbf{q}}$, such that
$\dot{\boldsymbol{\nu}} = \begin{bmatrix}\dot{\boldsymbol{\nu}}^{c_1} & \cdots & \dot{\boldsymbol{\nu}}^{c_k} & \dot{\boldsymbol{\nu}}^{l}\end{bmatrix}^T$ can be extracted.

4.4 Dynamics for control analysis

By assuming the inner attitude control loops are sufficiently fast enough, the orientation of the multirotor can be neglected for control synthesis purposes. Then a desired force in inertial space can be applied by allocating a desired angular velocity on the respectively rotors.

Therefore the translational motion of multirotor c_i can be defined as seen in e.g. [Klausen et al., 2015] by defining the position $\mathbf{p}^n$

$$\dot{\mathbf{p}}^n = \mathbf{v}^n \tag{4.30}$$

$$m\dot{\mathbf{v}}^n = m\mathbf{g}^n + \mathbf{f}^n + \mathbf{f}_L^n \tag{4.31}$$

given the inertial control force $\mathbf{f}^n$, the gravitational force $\mathbf{g}^n = [0, 0, g]^T$, and the mass of the multirotor m.

The load force $\mathbf{f}_L^n$ is a function of the pitch angle θ_L and roll angle ϕ_L of the load as discussed in [Klausen et al., 2014], by redefining $\mathbf{R}_l^{c_i} = \mathbf{R}_x(\phi_L)\mathbf{R}_y(\theta_L)$ and assuming the multirotor is a point mass in $\{n\}$ ($\mathbf{R}_{c_i}^n = \mathbf{I}_{3\times3}$) such that $\mathbf{R}_l^n = \mathbf{R}_l^{c_i}$ the load force is given as

$$\mathbf{f}_L^n = \mathbf{R}_l^n \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} = \begin{bmatrix} T\sin\theta_L \\ -T\sin\phi_L\cos\theta_L \\ T\cos\theta_L\cos\phi_L \end{bmatrix} \tag{4.32}$$

where T is the magnitude of the load force along the wire connecting the multirotor and the suspended load.

4.5 Collision simulation

An analytical simulation study was performed using the derived mathematical model for the constrained system. The purpose was to investigate the dynamics of the tension forces acting between the net and the multirotors during the collision. The magnitude of these forces are crucial as the multirotors must be able to hold the desired height during the collision as well.

For the simulations a net with length $l = 5$ m and height $h = 3$ m was suspended from two multirotors with two wires of length 1.5 m attached to the top corners of the net. The masses of the bodies were set as seen in Table 4.1a. Lastly the multirotors and the fixed-wing was instructed to follow the desired surge velocities as seen in Table 4.1b, with a desired multirotor distance[2] of 5 m.

The equations of the system were solved numerically using a fixed-step Runge-Kutta method of order 4 (RK4) [Egeland & Gravdahl, 2002, Ch. 14] running at 50 Hz. The fixed-wing UAV dynamical model and controller will not be presented here, but the model

[2]The multirotor velocity and cooperative controller will be discussed in more detail in Chapter 5

m_{c_i}	3 kg
m_a	0.5 kg
m_l	0.5 kg

(a) Masses of the vehicles involved

$v_{\bar{c}}$	3 m/s
v_a	18 m/s

(b) Desired along-track velocities

Table 4.1: Masses and velocity set-points

and the simulator[3] details can be seen in [Gryte, 2015], were the dynamics of the X-8 [Skywalker Technology Co., 2015] has been investigated. In Figure 4.5 the resulting relative along-track velocity between the fixed-wing and the centroid of the multirotors $\Delta v_x = v_{a,x} - v_{\bar{c},x}$ is given. It can be observed that the impact occurs after approximately 30 s with a relative velocity of approximately 15 m/s.

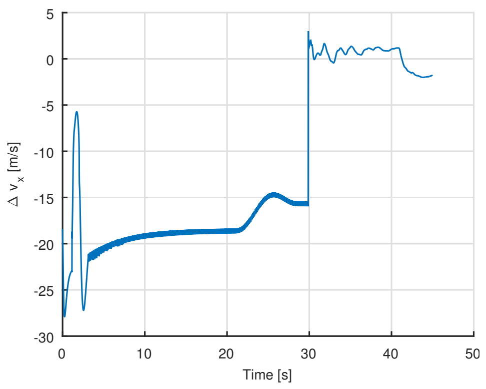
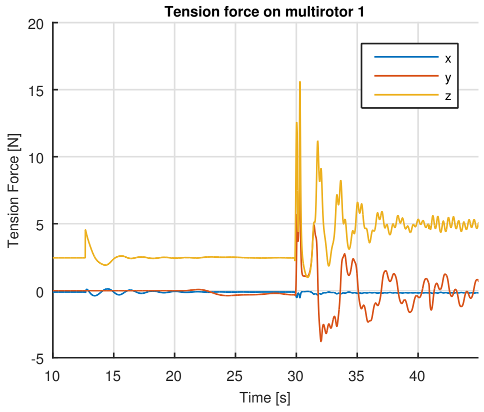


Figure 4.5: Relative along-track velocity during collision run

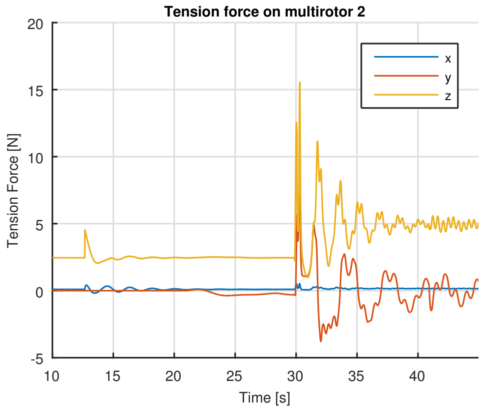
The expression for the general constrained forces $\mathbf{Q}_c$ as given in Equation (A.6) combined with the derivation in Section 4.3 gives the forces acting between the constrained bodies in the system, hence the constrained forces acting between the net and each multirotor can be expressed directly as seen in Figure 4.6. Here a tension spike during collision almost 3 times the nominal force with the fixed-wing suspended in the net can be observed. Also, the oscillations are present for almost 10 s, a period in which the tension is of a greater magnitude than the nominal force. One can conclude that the multirotors must be able to lift fairly more than the tension as given by the nominal gravitational pull from the suspended load.

Lastly, some snapshots from the collision can be seen in Figure 4.7 revealing that the impact might induce great fluctuations in the orientation of the net. However, as previously stated, the net is assumed to be rigid in this analysis. In reality the net will be deformed and absorb some of the energy.

[3]The source-code can be obtained from the UAV-Lab GitLab server `https://uavlab.itk.ntnu.no/88` (requires authorized access)



(a) Multirotor 1



(b) Multirotor 2

Figure 4.6: Tension force in the wire connecting each multirotor to the suspended net

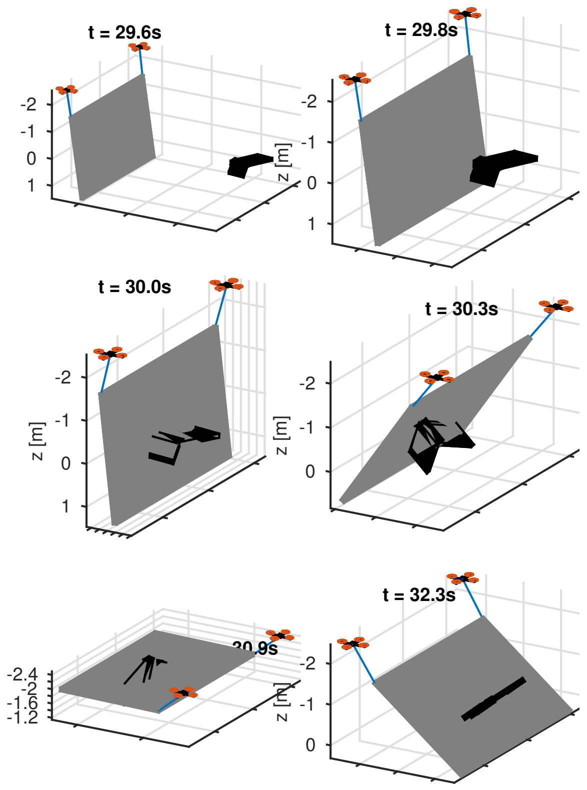


Figure 4.7: Snapshots from the collision

Chapter 5

Cooperative control

This chapter will explain how cooperative motion control has been utilized in order to move the multirotor formation as a tightly coupled vehicle. The formation has been treated as one *virtual vehicle* to create a modularized control scheme where the higher level control seek to control one vehicle, namely the *virtual vehicle*. The following methods will not emphasize the low-level cooperative control which ensures the different agents keep a desired formation, however, a brief summary will be gi given.

5.1 Centroid - *the virtual vehicle*

The following section will define the 6 degrees-of-freedom (DOF) states and its derivatives for the *virtual vehicle* following the same notation as in Chapter 4.

When considering the multirotor formation this work seeks to treat the formation as one coupled vehicle noted as the virtual vehicle. Multiple methods can be used to define the states of the virtual vehicle, such as defining one vehicle as the master agent where all other slave agents must move according to the current state of the master. However, for the purpose of performing a recovery maneuver, multiple agents must hold a suspended net together, requiring a symmetric formation to extend the net to its full size. Hence, a more advantageous method is to define the centroid of the agent formation as the virtual vehicle. Furthermore, when doing the recovery there must exist a point within the formation which the formation should seek to control according to the fixed-wing UAV position. Lastly, the center of the net will be located somewhere near the centroid with a height offset as illustrated in Figure 7.5, hence proving the benefits by using the centroid.

Position

The centroid position $\mathbf{p}_{\bar{c}}^n := \bar{\mathbf{p}}^n$ in $\{n\}$ for k agents with position $\mathbf{p}_{c_i}^n$ is then defined as follows

$$\mathbf{p}_{\bar{c}}^n := \frac{1}{k}\sum_{i}^{k} \mathbf{p}_{c_i}^n \tag{5.1}$$

which is illustrated in Figure 5.1 for two multirotors. Furthermore the velocity $\dot{\mathbf{p}}_{\bar{c}}^n$ and acceleration $\ddot{\mathbf{p}}_{\bar{c}}^n$ in $\{n\}$ can be found by differentiating all the succesive terms in the Equation (5.1) directly.

Orientation

For the general case with k agents it is not straight forward to define the orientation of the centroid as it is dependent of the formation shape. For some shapes it might make sense to define a rolling and pitching angle, for other these definitions might not be applicable. Therefore only two agents will be considered as all experiments will be conducted with two agents only. However, the methods used in later chapters can easily be expanded with more agents by defining the orientation for the given agent formation. For the two-agent formation, only the heading angle will be considered, as all further experiments seek to keep the same height on both agents to keep the net stable.

Firstly by defining the vector $\Delta\mathbf{p}_{c_{12}}^n$ from agent 1 to agent 2 as $\Delta\mathbf{p}_{c_{12}}^n := \mathbf{p}_{c_2}^n - \mathbf{p}_{c_1}^n$, the heading $\psi_{\bar{c}}$ of the centroid will be defined as

$$\psi_{\bar{c}} := -\text{atan2}(\Delta\mathbf{p}_{c_{12},x}^n, \Delta\mathbf{p}_{c_{12},y}^n) \tag{5.2}$$

where atan2(y, x) is the four-quadrant version of arctan(y/x). The centroid heading for two multirotors is illustrated in Figure 5.1 for two multirotors.

Furthermore the angular- velocity and acceleration can be found by normalizing the cross product between the arm from the centroid to one of the agent with the velocity and acceleration respectively

$$\boldsymbol{\omega}_{\bar{c}/n}^{\bar{c}} = \frac{\mathbf{S}(\mathbf{r}^{\bar{c}})\mathbf{v}^{\bar{c}}}{\|\mathbf{r}^{\bar{c}}\|^2} \tag{5.3}$$

$$\boldsymbol{\alpha}_{\bar{c}/n}^{\bar{c}} = \frac{\mathbf{S}(\mathbf{r}^{\bar{c}})\mathbf{a}^{\bar{c}}}{\|\mathbf{r}^{\bar{c}}\|^2} \tag{5.4}$$

where $\mathbf{r}^{\bar{c}} = \mathbf{R}_n^{\bar{c}}(\mathbf{p}_{c_i}^n - \mathbf{p}_{\bar{c}}^n)$ is the vector from the centroid to the agent given in $\{\bar{c}\}$ as seen in Figure 5.1 for two multirotors. Note that the angular- velocity and acceleration should be approximately the same for each agent i one use for the calculation. Further the velocity $\mathbf{v}^{\bar{c}}$ and acceleration $\mathbf{a}^{\bar{c}}$ is defined as the velocity of agent i in the centroid frame relative to the centroid, such that $\mathbf{v}^{\bar{c}} = \mathbf{R}_n^{\bar{c}}(\dot{\mathbf{p}}_i^n - \dot{\mathbf{p}}_{\bar{c}}^n)$ and $\mathbf{a}^{\bar{c}} = \mathbf{R}_n^{\bar{c}}(\ddot{\mathbf{p}}_i^n - \ddot{\mathbf{p}}_{\bar{c}}^n)$. However, it should be noted that these expressions assumes the arm $\mathbf{r}^{\bar{c}}$ is constant, this is in general not the case, but is sufficient in order to control the rotation of the formation where the arm can be assumed to be close to constant with slow movements.

Then the yaw- velocity and acceleration of the centroid can be found by extracting the z-component from the calculated angular- velocity and acceleration $\boldsymbol{\omega}$ and $\boldsymbol{\alpha}$ respectively.

$$r_{\bar{c}} = \boldsymbol{\omega}_z \tag{5.5}$$

$$\dot{r}_{\bar{c}} = \boldsymbol{\alpha}_z \tag{5.6}$$

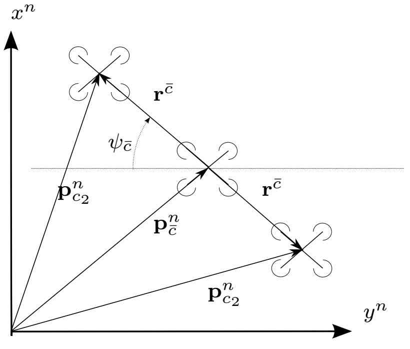


Figure 5.1: Two multirotors, c_1 and c_2 in $\{n\}$, defining the centroid $\bar{c}$ with position $\mathbf{p}^n_{\bar{c}}$ and heading $\psi_{\bar{c}}$.

6-DOF states
Firstly, as only the heading is defined, the Jacobian matrix for the system is simplified to the following

$$\mathbf{J}_{\psi} = \begin{bmatrix} \mathbf{R}^n_{\bar{c}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{5.7}$$

where $\mathbf{R}^{\bar{c}}_n = \mathbf{R}_z(\psi)$. Thereby it follows that $\dot{\psi}_{\bar{c}} = r_{\bar{c}}$ and $\ddot{\psi}_{\bar{c}} = \dot{r}_{\bar{c}}$

By inserting the expressions above into the existing notation it follows that

$$\boldsymbol{\eta}_{\bar{c}} = \begin{bmatrix} (\mathbf{p}^n_{\bar{c}})^\top & 0 & 0 & \psi_{\bar{c}} \end{bmatrix}^\top \tag{5.8}$$

then these expressions can be transformed back to body frame with the following transformations

$$\boldsymbol{\nu}^{\bar{c}} = \mathbf{J}_{\Theta}(\boldsymbol{\eta}_{\bar{c}})^{-1}\dot{\boldsymbol{\eta}}_{\bar{c}} \tag{5.9}$$

$$\dot{\boldsymbol{\nu}}^{\bar{c}} = \mathbf{J}_{\Theta}(\boldsymbol{\eta}_{\bar{c}})^{-1}\left[\ddot{\boldsymbol{\eta}}_{\bar{c}} - \dot{\mathbf{J}}_{\Theta}(\boldsymbol{\eta}_{\bar{c}})\boldsymbol{\nu}^{c_i}\right] \tag{5.10}$$

5.2 Centroid control

The purpose with the centroid definition is to be able to treat the *virtual vehicle* as one single body. This section will introduce the concept of centroid heading and velocity control, as well as a short introduction to the formation control.

5.2.1 Heading and formation control

The heading of the centroid as defined in Equation (5.2) is a function of the current agent formation. That is, in order to control the heading, the desired formation must be changed according to the desired heading.

In general, [Bai et al., 2011] handles the formation control by defining a matrix of the desired intervehicle relative positions. As the general case is not of interest and since this subject is covered in [Røli, 2015], the special case for two multirotor formation control will be summarized briefly here and is discussed in more detail in [Klausen et al., Submitted 2016].

The formation controller seeks to hold a desired intervehicle position. Hence by defining $\mathbf{z} := \mathbf{p}_2 - \mathbf{p}_1$ as the vector between the two multirotors in $\{n\}$ one seek a desired zero-heading vector $\mathbf{z}_{d,0} = \begin{bmatrix} 0 & l & 0 \end{bmatrix}$, where l is the desired distance between the two agents. Furthermore, in order to define a desired heading, the desired vector $\mathbf{z}_d$ will be given by rotating $\mathbf{z}_{d,0}$ by the desired centroid heading $\psi_{d,\bar{c}}$ around the z-axis

$$\mathbf{z}_d = \mathbf{R}_z(\psi_{d,\bar{c}})\mathbf{z}_{d,0} \tag{5.11}$$

then $\mathbf{z}_d = \mathbf{z}_{d,0}$ defines the formation at zero heading $\psi_{d,\bar{c}} = 0$ as illustrated in Figure 5.2.

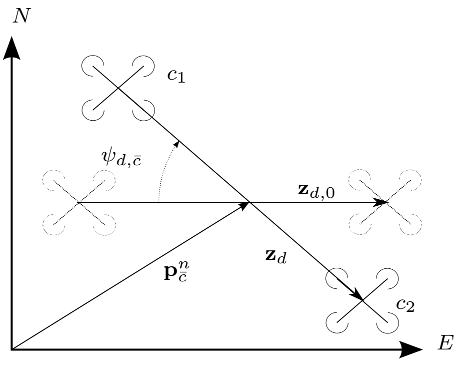


Figure 5.2: Desired formation link $\mathbf{z}$ as a function of the zero heading formation $\mathbf{z}_{d,0}$ and the desired formation heading $\psi_{d,\bar{c}}$

By setting the desired agent velocity $\mathbf{v}_{i,df}^n$ the controller will render the formation error $\tilde{\mathbf{z}} := \mathbf{z} - \mathbf{z}_d$ to zero. The reader is referred to [Røli, 2015] or [Bai et al., 2011] for derivation and stability proof for the following controller.

$$\mathbf{v}_{i,df}^{n} = d_i K_l(\mathbf{z} - \mathbf{z}_d) \tag{5.12}$$

where $d_1 = 1$, $d_2 = -1$ and $K_l > 0$ is the controller gain, referred to as the *link gain*.

5.2.2 Link gain scheduling

The formation controller in Section 5.2.1 will hold the desired formation. However, to keep the net steady it is required that the formation controller is agile and keeps the formation at all time. As the link gain K_l is static, one will experience more aggressive behavior when the error $\tilde{\mathbf{z}}$ is large as seen in Section 5.3.1. Therefore a gain scheduling scheme was proposed. Here there exists two cases, either the agents are lining up to the desired formation during startup, or they are in formation. Therefore a logistic function was nominated, where the two extrema points defines two different gain levels connected by a smooth transition as a function of the norm of the distance error $\tilde{z} := \|\tilde{\mathbf{z}}\|^2$

$$K_l(z) = K + \frac{K_f - K}{1 + e^{-k(\tilde{z} - \tilde{z}_0)}} \tag{5.13}$$

where K and K_f are constants, k gives the steepness of the transition slope and $\tilde{z}_0$ is the transition point.

The adaptive link gain function in Equation (5.13) has the following properties

$$K_l(0) = K + \frac{K_f - K}{1 + e^{k(\tilde{z}_0)}} = K_c \tag{5.14}$$

$$\lim_{z \to \infty} K_l(z) = K_f \tag{5.15}$$

$$K_l(\tilde{z}_0) = \frac{K_f + K}{2} \tag{5.16}$$

such that K_f and K_c gives the gain with large errors $\tilde{z}$ and zero error (in desired formation) respectively. Then the constant K can be expressed as follows

$$K = \frac{1}{e^{k\tilde{z}_0}}(K_c - K_f) + K_c$$

For practical implementations it is hard to physically interpret the parameter k. However, k can be determined by requiring a certain gain at some link error. For example, by requiring that the link gain is a fraction $r \in [0, 1]$ of the in-formation gain K_c at some desired link error z_r, giving $K_l(z_r) = rK_c$. This equality can then be solved giving an explicit expression for the steepness factor k.

5.2.3 Velocity control

For each multirotor i a velocity controller is implemented which gives the desired force in the body frame $\{c_i\}$. For simplified notation, i is used to denote c_i in the following derivation.

The desired heading and velocities from the guidance controllers are given in the centroid frame $\{\bar{c}\}$ and must be rotated back to each individual multirotor frame $\{i\}$ by the following rotation

$$\mathbf{R}^i_{\bar{c}} = \mathbf{R}^i_n \mathbf{R}^n_{\bar{c}} = (\mathbf{R}^n_i)^\top \mathbf{R}_z(\psi_{\bar{c}}) \tag{5.17}$$

where $\psi_{\bar{c}}$ is the actual heading of the centroid. Then a velocity controller is proposed which gives $\mathbf{f}^i$ the desired force applied on multirotor i given the velocity feedback $\mathbf{v}^i$

$$\mathbf{f}^i = \mathbf{K}_p(\mathbf{R}^i_{\bar{c}}\mathbf{v}^{\bar{c}}_d + \mathbf{R}^i_n \mathbf{v}^n_{i,df} - \mathbf{v}^i) + m_i \mathbf{R}^i_{\bar{c}}\dot{\mathbf{v}}^{\bar{c}}_d \tag{5.18}$$

where $\mathbf{K}_p \in \mathbb{R}^{3\times 3}$ is a positive definite tuning matrix. Further the desired velocity $\mathbf{v}^{\bar{c}}_d$ and acceleration $\dot{\mathbf{v}}^{\bar{c}}_d$ for the centroid is given from the guidance law, lastly $\mathbf{v}^n_{i,df}$ gives the desired velocity from the formation controller as seen in Section 5.2.1.

Then the desired force $\mathbf{f}^i$ can be rotated to $\{n\}$

$$\mathbf{f}^n_i = \mathbf{R}^n_i \mathbf{f}^i \tag{5.19}$$

Feed-forward

Using the results from Section 4.4, one assumes the multirotors can be modeled as point masses with a positions $\mathbf{p}^n$ in the inertial space subjected to a suspended payload. Expanding Equation (4.30)–(4.32) gives for each agent i

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = m\begin{bmatrix}0\\0\\g\end{bmatrix} + \begin{bmatrix}T\sin\theta_L\\ -T\sin\phi_L\cos\theta_L\\ T\cos\theta_L\cos\phi_L\end{bmatrix} + \begin{bmatrix}F_x\\F_y\\F_z\end{bmatrix} \tag{5.20}$$

Further by compensating for the gravity of the multirotor

$$\mathbf{f}^*_{c_i} = \mathbf{f}^n_i - m_i\mathbf{g}^n \tag{5.21}$$

By measuring the load an optional feed-forward (FF) term $\mathbf{f}_{FF} = -\mathbf{f}_L$ can be added for compensation which gives the two controller schemes

$$\mathbf{f}^n_{c_i} = \begin{bmatrix}F_x & F_y & F_z\end{bmatrix}^T = \begin{cases}\mathbf{f}^*_{c_i} & \text{if FF disabled}\\ \mathbf{f}^*_{c_i} + \mathbf{f}_{FF} & \text{if FF enabled}\end{cases} \tag{5.22}$$

Lastly, the structure of the low-level control scheme is summarized in Figure 5.3

5.2.4 Adaptive control

As the system is subjected to a major external force, in which the system dynamics changes abruptly a robust control scheme is necessary. However, the model parameters are not known explicitly and they are in general hard to find, therefore a model dependent control scheme is not desirable. The simplified dynamics as given in Section 4.4 extracts the major

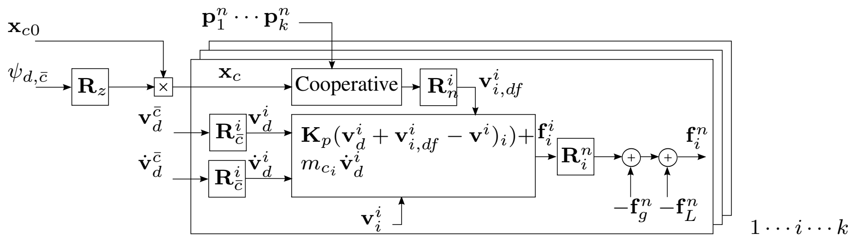


Figure 5.3: Centroid heading and velocity control for agent $1 \cdots i \cdots k$

dynamics of the system, however, other effects may rise and the measurement of the load force $\mathbf{f}_L^n$ may not be exactly correct. Such effects can be compensated by introducing integral effect in the control scheme. Here an adaptive control scheme will be proposed in order to adapt the gains of the controllers to compensate for these effects. A direct MRAC scheme as introduced in Appendix C.1 was used for this purpose.

1-st order MRAC

The control objective is to render the velocities of the multirotors to a desired velocity, hence a 1-st order MRAC was proposed. In order to use the MRAC scheme the system model should be given on a state-space form as described in Appendix C.1. The translational model for multirotor i as seen in Equation (4.30)–(4.31) can be represented on a state space form by defining the state vector $\mathbf{x} = \mathbf{v}^n \in \mathbb{R}^3$, then by defining

$$\mathbf{A} = \mathbf{0}_{3\times 3} \tag{5.23}$$

$$\boldsymbol{\alpha} = \mathbf{f}^n + \mathbf{f}_L^n + \mathbf{f}_g^n \tag{5.24}$$

$$\mathbf{B} = \frac{1}{m}\mathbf{I}_{3\times 3} \tag{5.25}$$

the system can be set on the following state-space form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}(\mathbf{u} + \boldsymbol{\alpha}) \tag{5.26}$$

Further it can be shown that $(\mathbf{A}, \mathbf{B})$ is controllable by calculating the row rank of the *controllability matrix* $\mathbb{C}$ as stated in Equation (C.11) in Appendix C.2. Hence a reference model and an adaptive control scheme can be found. Then a first order reference model is required, by defining the reference state $\mathbf{x}_r = \mathbf{v}_r^n \in \mathbb{R}^3$ as the model state and let the input be the desired velocity $\mathbf{r} = \mathbf{v}_d^n + \mathbf{v}_{i,df}^n$ be the sum of the desired velocity from the reference model $\mathbf{v}_d^n$ and the desired coordination control velocity $\mathbf{v}_{i,df}^n$. Then the reference model can be given as a first order model a state-space form as in Equation (C.2) by defining

$$\mathbf{A}_r = -\frac{1}{T}\mathbf{I}_{3\times 3} \tag{5.27}$$

$$\mathbf{B}_r = \frac{1}{T}\mathbf{I}_{3\times 3} \tag{5.28}$$

Adaptive laws

For the following derivation it is given that the absolute value of the load force $T = |\mathbf{f}_L^n|$ as defined in Equation (4.32) is measurable. Further, let the load force $\mathbf{f}_L^n$ be redefined as $\mathbf{f}_L^n = T\mathbf{\Phi}$ where $\mathbf{\Phi} = \mathbf{\Phi}(\phi_L, \theta_L) \in \mathbb{R}^3$ is a measured vector

$$\mathbf{\Phi} = \begin{bmatrix} \sin\theta_L \\ \sin\phi_L \cos\theta_L \\ \cos\theta_L \cos\phi_L \end{bmatrix} \tag{5.29}$$

then the adaptive control law $\mathbf{u}$ can be given as follows

$$\mathbf{u} = -\tilde{\mathbf{K}}(t)\mathbf{x} + \tilde{\mathbf{L}}(t)\mathbf{r} - T\mathbf{\Phi} - \mathbf{f}_g^n + \mathbf{f}_{i,df}^n \tag{5.30}$$

where the adaption laws for the adaptive gain matrices $\mathbf{K}(t)$ and $\mathbf{L}(t)$ is as given in Equation (C.8)–(C.9) and restated here

$$\dot{\tilde{\mathbf{K}}} = \mathbf{B}_r^T \mathbf{P} \mathbf{e} \mathbf{x}^T \operatorname{sgn}(l)$$
$$\dot{\tilde{\mathbf{L}}} = -\mathbf{B}_r^T \mathbf{P} \mathbf{e} \mathbf{r}^T \operatorname{sgn}(l)$$

where $\mathbf{f}_g^n$ and $\mathbf{f}_{i,df}^n$ is the gravity compensation and desired force from the cooperative controller. Further $\mathbf{e} = \mathbf{x} - \mathbf{x}_r$ is the error state and $\mathbf{P}$ is given by the Lyapunov equation Equation (C.7). Hence, the response of the system can be tuned by setting the matrix $\mathbf{Q}$ from the Lyapunov equation and the initial estimates of the gain matrices $\tilde{\mathbf{K}}(t_0) = \tilde{\mathbf{K}}_0$ and $\tilde{\mathbf{L}}(t_0) = \tilde{\mathbf{L}}_0$.

Then the alternative low-level MRAC for velocity control scheme can be seen in Figure 5.4. Comparing with Figure 5.3 one can observe that the controller now uses the references in $\{n\}$.

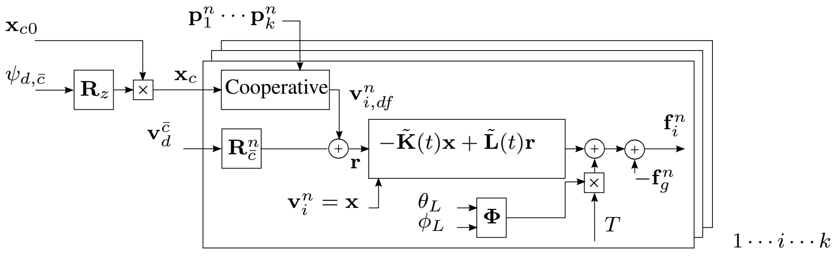


Figure 5.4: Centroid heading and MRAC velocity control for agent $1 \cdots i \cdots k$

5.3 Simulations

5.3.1 Adaptive link gain

The benefits of using an adaptive link gain scheme were shown with the following simulation study. Using the parameters as summarized in Table 5.1, two multirotors were simulated using RK4 on 50 Hz with the cooperative controller from Section 5.2.1.

Link gain static	0.5
Gain close, K_c	0.5
Gain far, K_f	0.05
$\tilde{z}_0$	7.0 m
$z_{0.9}$	4.5 m
Desired link length, l	3.0 m
Desired heading, $\psi_{d,\bar{c}}$	0 °

Table 5.1: Link gain-scheduling parameters

The predefined link gain function $K_l(\tilde{z})$ is represented in Figure 5.5 and gives the desired link gain $K_l = K_l(\tilde{z})$. It follows that the formation controller will be more aggressive to keep the desired formation, and less aggressive when the distance error is large. This will ensure a less aggressive initial multirotor maneuver when the formation controller is enabled on with a large distance error. Furthermore, the critical points $\tilde{z}_0$ and $z_{0.9}$ must be chosen such that a high link gain is kept for all maneuvers with the formation controller active.

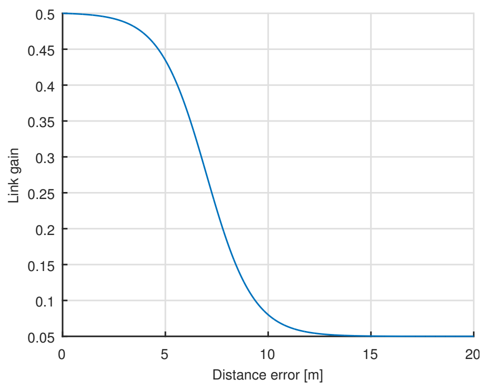
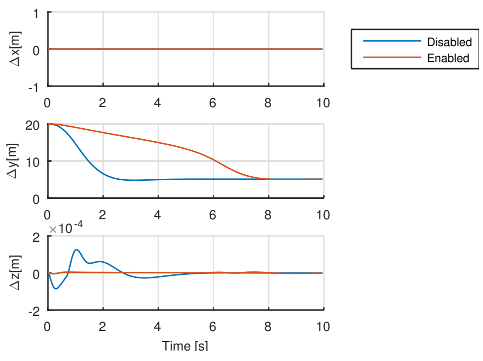
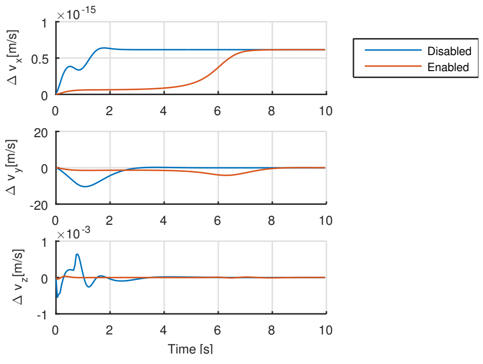


Figure 5.5: Link gain $K_l(\tilde{z})$ as a function of the difference error norm $\tilde{z}$

Two simulations were conducted with the link gain scheduling enabled and disabled and the results are given in Figure 5.6[1]. With an initial link distance of 20 m the position and velocity difference responses were compared. A smoother response towards the desired formation can be observed with the gain scheduling active. However, the gain scheduling also rises an minor abrupt change in the relative velocity as can be observed in the Δv_y graph in Figure 5.6b. This is as expected, as the link gain will switch to a higher gain when closing up towards the desired formation.



(a) Position difference $\Delta \mathbf{p} = \mathbf{p}_2 - \mathbf{p}_1$



(b) Velocity difference $\Delta \dot{\mathbf{p}} = \dot{\mathbf{p}}_2 - \dot{\mathbf{p}}_1$

Figure 5.6: Comparing position and velocity difference with link gain scheduling enabled and disabled.

[1]Two videos with the link gain scheduling enabled and disabled can be seen in *Simulations/Link Gain Scheduling - Enabled.mp4* and *Simulations/Link Gain Scheduling - Disabled.mp4* respectivly in the *Digital Appendix*.

Chapter 6

Payload transport

The purpose with the cooperative guidance scheme is to smoothly move the net along a desired path. In this setting, smoothly is defined as a trajectory with minimal sideway motion and damped along-track velocity profile. Moreover, the centroid heading should not change abruptly to avoid oscillations in the heading of the net. As stated in Chapter 5 the following chapter will control the *virtual vehicle*, namely the centroid.

6.1 Centroid LOS steering law

A Line-Of-Sight (LOS) steering law was implemented in order to follow a straight path given by two waypoints and a desired along-path velocity. This approach is addressed in the two dimensional horizontal North-East plane for marine applications in [Fossen, 2011, Ch. 10] which gives different control schemes for this purpose. Expanding the method to three dimension is trivial and the same methods can be applied as discussed in e.g. [Caharija et al., 2012].

It should be noted that the LOS steering law seek to generate a reference centroid body velocity $\mathbf{v}^{\bar{c}}_{ref}$ and heading $\psi_{\bar{c},ref}$ which will be the input for the reference simulator generating a desired smooth trajectory for the low level control systems.

By defining waypoint k as $\mathbf{p}^n_k = [p_{k,x}\ p_{k,y}\ p_{k,z}]^\top$ and the leg $\Delta\mathbf{p}^n_k = \mathbf{p}^n_{k+1} - \mathbf{p}^n_k$ one can calculate the path-tangential angle α_k and the pitching angle θ_k with respect to the inertial frame $\{n\}$

$$\alpha_k = \arctan\left(\frac{\Delta p_{k,y}}{\Delta p_{k,x}}\right) \tag{6.1}$$

$$\theta_k = -\arctan\left(\frac{||\Delta p_{k,1:2}||^2}{\Delta p_{k,z}}\right) + \frac{\pi}{2} \tag{6.2}$$

The rotation matrix from the path frame $\{p\}$ to the NED frame $\{n\}$ is defined as $\mathbf{R}_n^p = \mathbf{R}_z(-\theta_k)\mathbf{R}_y(\alpha_k)$ then the cross-track errors $\boldsymbol{\epsilon}(t) = [s\ e_y\ e_z]^\top$ can be calculated based on the current position $\mathbf{p}^n$ and waypoint $\mathbf{p}_k^n$ in the $\{n\}$ frame.

$$\boldsymbol{\epsilon}(t) = (\mathbf{R}_p^n)^\top(\mathbf{p}^n - \mathbf{p}_k^n) \tag{6.3}$$

further the cross-track error can be derivated with respect to time to get the time-changing errors.

$$\dot{\boldsymbol{\epsilon}}(t) = (\mathbf{R}_p^n)^\top\dot{\mathbf{p}}^n \tag{6.4}$$

The steering law was implemented as based on the methods from [Fossen, 2011, Ch. 10] which gives the desired course $\chi_{ref}(t)$ and pitching angle $\theta_{ref}(t)$. The lookahead distances Δ_y and Δ_z are design-variables to tune how fast the system should approach the desired path

$$\chi_{ref}(t) = \alpha_k + \arctan\left(\frac{-e_y}{\Delta_y}\right) \tag{6.5}$$

$$\theta_{ref}(t) = \theta_k + \arctan\left(\frac{-e_z}{\Delta_z}\right) \tag{6.6}$$

Further, a mechanism for switching waypoints was needed. In [Fossen, 2011, Ch. 10] this is solved by monitoring the Euclidean distance to the next waypoint $||\mathbf{p}_{k+1}^n - \mathbf{p}^n||^2$ and switch when the distance is lower than some radius R_k, refereed to as the *waypoint acceptance radius*. However, as the purpose with this steering law is not to reach the waypoint, but to create a feasible path the along-track distance is monitored instead. Therefore k should be incremented when the following inequality holds

$$|(\mathbf{R}_p^n)^\top(\mathbf{p}_{k+1}^n - \mathbf{p}^n)_x| \leq R_{k+1}^2 \tag{6.7}$$

6.1.1 Reference LOS velocity

Traditionally the reference velocity in $\{n\}$ can be generated given the desired along-track surge velocity. That is given the rotation matrix from the desired path to the local $\{n\}$ frame $\mathbf{R}_{p,d}^n = \mathbf{R}_z(-\theta_d)\mathbf{R}_y(\chi_d)$ and the desired along-track velocity $v_{x,d}^p$ one get the desired velocity $\mathbf{v}_{ref}^n$ in the local $\{n\}$ frame

$$\mathbf{v}_{ref}^n = \mathbf{R}_{p,d}^n\left[v_{x,d}^p\ 0\ 0\right]^\top \tag{6.8}$$

6.1.2 Surge and heading

As it is desired to avoid to much cross-track movements with the net during transport, another approach is to use the desired along-track surge directly as the reference velocity. Then the LOS steering law can be used to generate a reference heading.

Firstly, the along-track surge velocity must be rotated back to $\{n\}$ by rotating with the current centroid heading $\psi_{\bar{c}}$. This gives the velocity $\mathbf{v}_*^n$, the path surge velocity given in $\{n\}$.

$$\mathbf{v}_*^n = \mathbf{R}_z(\psi_{\bar{c}}) \left[v_{x,d}^p \; 0 \; 0 \right]^\top \tag{6.9}$$

The North-East components from $\mathbf{v}_*^n$ can be used to generate the North-East desired velocity. However, as only the heading will be controlled the Down-component must be generated in another fashion in order to be able to control the height. Here the Down-component from $\mathbf{v}_{ref}^n$ can be used, which generates a smooth LOS reference velocity.

Then the total reference velocity in $\{\bar{c}\}$ $\mathbf{v}_{ref}^{\bar{c}}$ is found

$$\mathbf{v}_{ref}^{\bar{c}} = \mathbf{R}_n^{\bar{c}} \left[(\mathbf{v}_{*,1:2}^n)^\top \; \mathbf{v}_{ref,z}^n \right]^\top \tag{6.10}$$

Lastly, the reference heading ψ_{ref} is given as the desired course $\chi_{ref}(t)$ ensuring that the desired path in the North-East plane is followed.

$$\psi_{ref} = \chi_{ref}(t) \tag{6.11}$$

6.2 Marine craft simulator

A closed loop simulator was considered in order to constrain the formation dynamics. The reference centroid heading and velocity in Section 6.1.2 ensures minimal sideway motion during transit by setting the reference sway velocity to zero. However, one might get abrupt movements in the net when changing way-points, where the desired heading might step to a new set-point, or if the reference surge speed steps to another set-point during transit. Hence, a closed loop reference simulator on the surge speed and heading was proposed.

The concept of using a closed loop simulator as a reference generator are discussed in e.g. [Fossen, 2011], here an surge and heading simulator is also presented. The dynamics of the centroid can now be compared with an underactuated ocean surface vessel where only surge and heading can be controlled using a rudder and fixed propellers. Therefore the simulator will mimic the dynamics of such a vessel and use a heading and surge controller to reach the reference heading and surge. Then the centroid will use the actual heading and surge of the vessel as the desired trajectory which can be tuned to fulfill the requirements for a smooth movement of the net.

6.2.1 Surge dynamics and control

The surge dynamics is subjected to a non-linear drag term interpreting a constrained change in surge velocity.

$$m\dot{u}_d + d|u_d|u_d = \tau \tag{6.12}$$

here u_d is the surge speed of the simulated vessel, m gives the mass of vessel, and d the amount of drag. By increasing the mass of the vessel the linear momentum will increase and it will require more energy to change the current gained surge velocity. Furthermore the higher drag coefficient d, the more energy is required to accelerate the vessel.

A PI-controller is proposed to control the surge velocity by applying a thrust τ. The reference surge u_{ref} given from the guidance scheme is the desired set-point. By defining the error signal $e_u = u_{ref} - u_d$ the thrust τ is given as

$$\tau = K_{p\tau} e_u + K_{i\tau} \int_0^t e_u \, dt \tag{6.13}$$

rendering u_d to u_{ref}, where $K_{i\tau} > 0$ for $i \in \{p, i\}$.

6.2.2 Heading dynamics and control

For the heading dynamics of ψ_d a first-order Nomoto model was used.

$$\dot{\psi}_d = r_d \tag{6.14}$$

$$T\dot{r}_d + r_d = K\delta \tag{6.15}$$

where T is the time-constant for the yawing motion. The heading is changed by applying a yawing moment $K\delta$, where δ is the rudder angle and K is the rudder-gain.

Here a PID-controller is proposed to control the heading by applying a rudder angle δ. Defining the error signal $e_\psi = \psi_{ref} - \psi_d$, where ψ_{ref} is the desired heading given by the guidance scheme the rudder angle is given as

$$\delta = K_{p\delta} e_\psi + K_{i\delta} \int_0^t e_\psi \, dt + K_{d\delta} \dot{e}_\psi \tag{6.16}$$

In order to effectively tune the controller, a tuning procedure from [Fossen, 2011, Ch. 12] was used. By setting the up the closed loop dynamics without integral effects

$$T\ddot{\psi}_d + (1 + KK_{d\delta})\dot{\psi}_d + KK_{p\delta}\psi_d = KK_{p\delta}\psi_{ref} \tag{6.17}$$

and comparing Equation (6.17) with the standard form of a 2nd order dynamics

$$\ddot{\psi}_d + 2\zeta_\psi \omega_{0,\psi} \dot{\psi}_d + \omega_{0,\psi}^2 \psi_d = \omega_{0,\psi}^2 \psi_{ref} \tag{6.18}$$

the following relation between can be found, where the rule-of-thumb proposed in [Fossen, 2011, Ch. 12] is to set the integral gain 10 times lower than the closed loop natural frequency $\omega_{0,u}$.

$$K_{p\delta} = \frac{T}{K}\omega_{0,\psi}^2 \tag{6.19}$$

$$K_{i\delta} = \frac{\omega_{0,\psi}}{10} \tag{6.20}$$

$$K_{d\delta} = \frac{1}{K}(2\zeta_\psi\omega_{0,\psi}T - 1) \tag{6.21}$$

then the response can be tuned to fit the desired standard form in Equation (6.18).

The overall payload transport guidance scheme is summarized in Figure 6.1, emphasizing the input and outputs to the LOS controller and simulator respectivly.

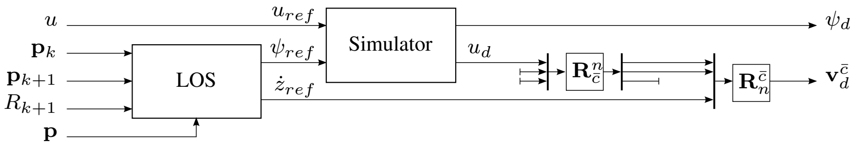


Figure 6.1: Path control with marine craft simulator

6.3 Simulations

6.3.1 Reference simulator

A simulation study was conducted showing the feasability of the guidance law from Section 6.1 and the reference simulator in Section 6.2. Again, the system was solved using RK4 running at 50 Hz. Furthermore, the LOS controllers look-ahead distances Δ_y and Δ_z were both set to 10. The reference simulator parameters are summarized in Table 6.1. Lastly the inner-loop velocity gain matrix were set to $\mathbf{K}_p = 2m_{c_i}\mathbf{I}_{3\times3}$.

Reference surge, u_{ref}	0.7 m/s
Nomoto, timeconstant: T	0.9 s
Nomoto, rudder gain: K	1
Surge, mass: m	10 kg
Surge, damping: d	1
Heading control damping: ζ_ψ	1
Heading control bandwidth: $\omega_{0,\psi}$	0.7 rad/s
Surge controller gains: $K_{p\tau}$,$K_{i\tau}$	5,1

Table 6.1: Key reference simulator parameters in simulation

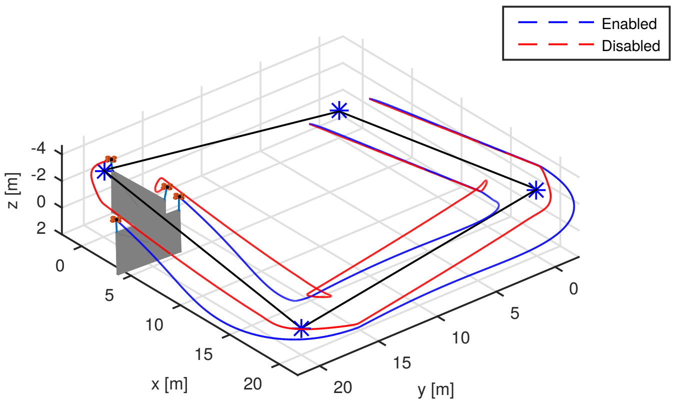
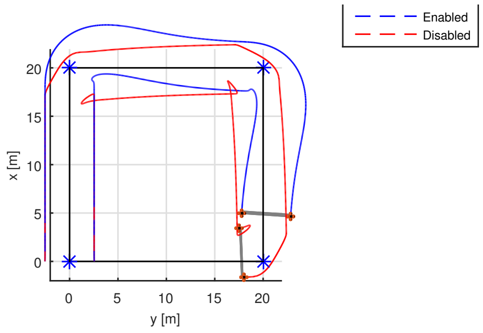


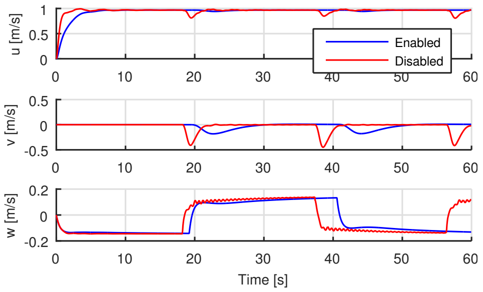
Figure 6.2: Multirotor positions in $\{n\}$ with reference simulator enabled and disabled, the waypoints are given as the blue stars connected by the black lines.

The guidance scheme was simulated using a path defined by four waypoints located at different heights as seen in Figure 6.2.

Two simulations were conducted with the reference simulator enabled and disabled[1]. As expected the reference simulator gives a slower response and a much higher turning radius which is seen more clearly in Figure 6.3a. However, this is the purpose with the reference simulator, as it is desired to keep the sidemotion of the net at a minimum. If one consider the body velocities of the centroid in Figure 6.3b it is clear that the sideway motion is less aggressive and of lower magnitude, also, the surge speed is less agressive and rough.



(a) North-East multirotor positions with reference simulator enabled and disabled

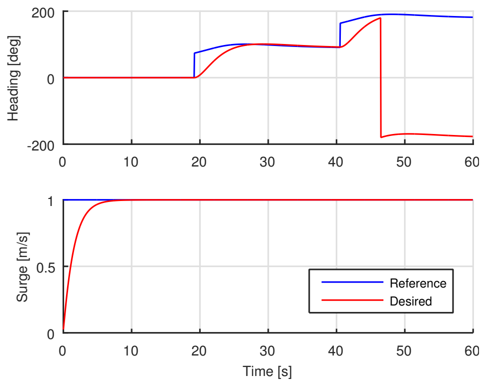


(b) Centroid body frame $\{\bar{c}\}$ velocities comparing reference and desired values with the reference simulator enabled

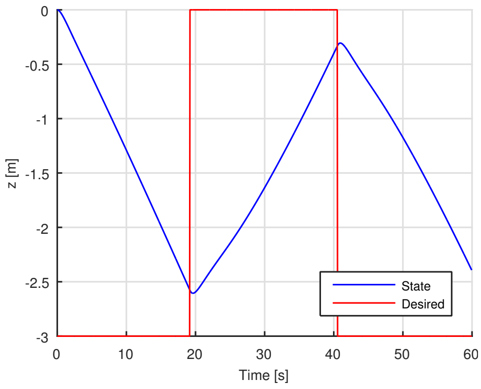
Figure 6.3: Comparing responses with reference simulator enabled and disabled

[1]Two videos showing the position and orientation of the multirotors with the reference simulator enabled and disabled can be seen in *Simulations/Reference Simulator - Enabled.mp4* and *Simulations/Reference Simulator - Disabled.mp4* respectivly in the *Digital Appendix*

The performance of the reference simulator can be seen in Figure 6.4a where a smoother and less aggressive desired heading and surge are dispatched to the inner coordination and velocity controller. Lastly the results of the LOS height controller are given in Figure 6.4b and Figure 6.3b where it can be seen that the controller is decoupled from the reference simulator as expected, furthermore, it is not able to reach the desired height before switching waypoint. This has to do with choice of look-ahead distance Δ_z, the distance between the waypoint and the desired switching radius R_k. It should also be noted that the purpose with this guidance scheme is not to reach the desired way-points, but to construct a rough desired path for transportation purposes. For the purpose of reaching a certain way-point a position controller of the form as discussed in Section 7.4 should be considered.



(a) Comparing reference simulator inputs (reference) and output (desired); heading ψ and surge u.



(b) Actual height compared against the reference height

Figure 6.4: Reference simulator states and the height.

6.3.2 MRAC

A simulation with the MRAC scheme from Section 5.2.4 was conducted as an alternative to the inner loop velocity controller with the reference simulator enabled for a smooth desired heading for the cooperative control. A RK4 method at 100 Hz were used to ensure stability. The MRAC controller response are dependent of the initial gain matrices $\hat{\mathbf{K}}_0$ and $\hat{\mathbf{L}}_0$, also the transient response can be tuned with the tuning matrix $\mathbf{Q}$, for this simulation the parameters as given in Table 6.2 were used.

$\mathbf{Q}$	$3\mathbf{I}_{3\times 3}$
$\hat{\mathbf{K}}_0$	$-3\mathbf{I}_{3\times 3}$
$\hat{\mathbf{L}}_0$	$3\mathbf{I}_{3\times 3}$

Table 6.2: Key MRAC path-control parameters in simulation

Four waypoints with the same height were defined forming a square as seen in Figure 6.5. Further it can be seen that the MRAC is able to follow the desired path, given the desired velocity[2]. As the initial gain matrices are set close to the desired gain matrices as seen in Figure 6.7 the multirotors follows the initial desired reference closely.

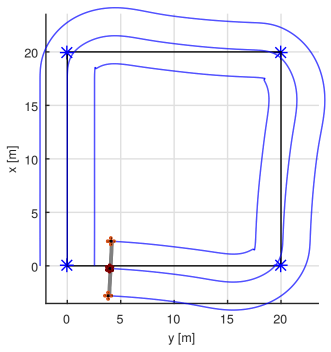
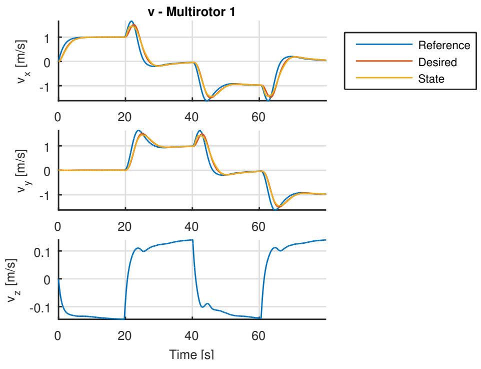
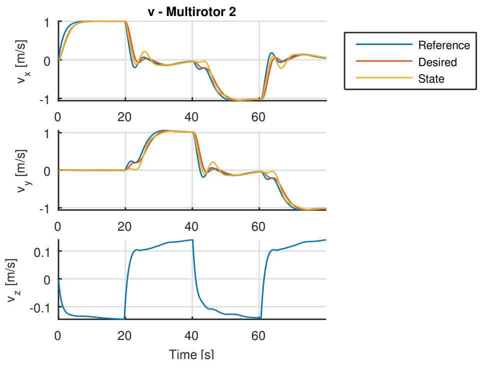


Figure 6.5: Multirotor and centroid positions in the North-East plane using MRAC for path-control, the blue stars gives the desired waypoints connected by the black lines.

On the other hand, the MRAC is a velocity controller, and the responses of both multirotors are seen in Figure 6.6. It is clear that some time is required for the gain matrices to stabilize, however, both multirotors seem to be able to follow the desired velocity. Some transients can be observed, but these are located at the points were the desired formation changes due to corners of the path and a turn must be performed.



(a) Multirotor 1

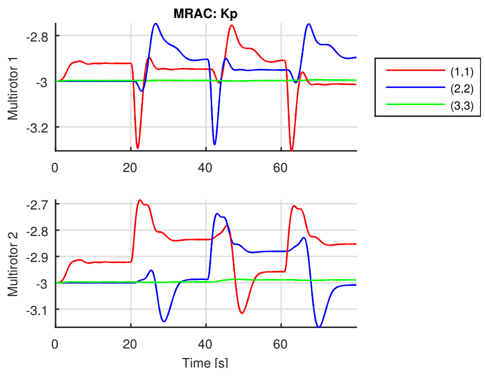


(b) Multirotor 2

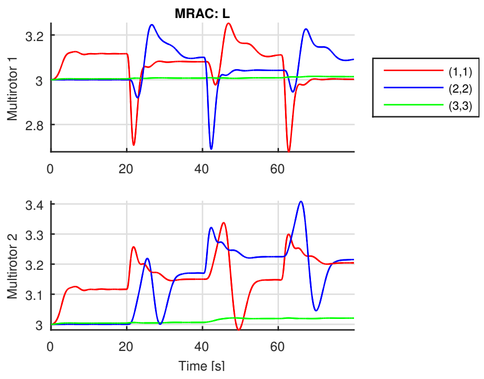
Figure 6.6: Velocities of the multirotors in $\{n\}$ during the path-maneuver, comparing the reference, desired and the actual state.

[2]A video showing the multirotor positions and orientations can be seen in *Simulations/Path Control - MRAC.mp4* in *Digital Appendix*.

Lastly, the diagonal elements of the MRAC gain matrices can be seen in Figure 6.7. The observations from the previous figures can be explained with the responses seen here. It is clear that some time is required for the matrices to stabilize. Furthermore, a correlation between the transient responses of the matrices and the transients in the velocity of the multirotors can be seen. Again, this is related to the turning points, were the coordination controller also tries to hold the current formation.



(a) Diagonal elements of K_p



(b) Diagonal elements of L

Figure 6.7: Multirotor adaptive gains during the path-maneuver.

Chapter 7

The recovery maneuver

The guidance scheme in Chapter 6 presents a solution for the transportation of a suspended load using multiple multirotors, whether the fixed-wing UAV is recovered in the net or not. This chapter seeks to find a parallel guidance scheme for the purpose of safely recover the fixed-wing UAV. Modularity can be achieved by requiring that the control objective is to reach a desired centroid heading and velocity as in Chapter 6. For all further analysis the fixed-wing UAV is assumed to be the master agent for the maneuver, thereof, the movement of the fixed-wing UAV will trigger an eventual recovery. Furthermore, fixed-wing UAV control schemes will not be considered.

For safety reasons the movement of the centroid is constrained to be within some cuboid named the *virtual-runway*. By instructing the fixed-wing to move and engage its landing maneuver towards the virtual-runway, the centroid control objective can be subdivided into an along- and cross-track guidance scheme. Firstly, track the cross-track position of the fixed-wing and then engage an along-track trajectory to recover the fixed-wing within the virtual-runway.

7.1 Virtual-runway

The virtual-runway is defined with an origin $\mathbf{p}^n_{p/n}$ and an orientation $\mathbf{\Theta}_{np}$ as illustrated in Figure 7.1. In general the path can have a desired heading ψ_p and pitching angle θ_p, however, for all tests performed the pitching angle is set to zero. Furthermore, the concept of the virtual-runway already gives the possibility for the copters to move freely inside the box, such that the height of the net can be adjusted according to the height of the fixed-wing UAV. Therefore, for an initial concept, a pitching angle of the path is not necessary. Hence, it follows that $\mathbf{\Theta}_{np} = [0 \; 0 \; \psi_p]^\top$ and the rotation matrix from $\{n\}$ to $\{p\}$ is $\mathbf{R}^n_p = R_z(\psi_p)$.

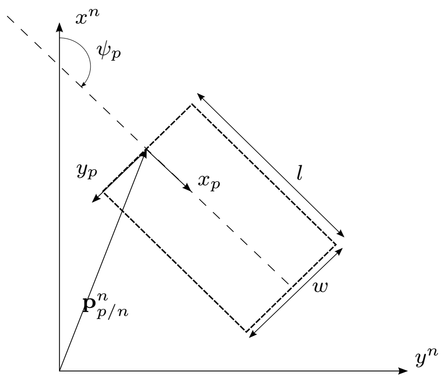


Figure 7.1: Virtual-runway $\{p\}$ defined in the inertial frame $\{n\}$

The transformation from the 6-DOF centroid state $\boldsymbol{\eta}_{\bar{c}}$ of the centroid in $\{n\}$ to the virtual-runway path frame $\{p\}$ is then defined as follows

$$\dot{\mathbf{p}}^p_{\bar{c}/p} = \mathbf{R}^p_n(\mathbf{p}^n_{\bar{c}/n} - \mathbf{p}^n_{p/n}) \tag{7.1}$$

$$\boldsymbol{\Theta}_{p\bar{c}} = \boldsymbol{\Theta}_{n\bar{c}} - \boldsymbol{\Theta}_{np} \tag{7.2}$$

then the 6-DOF state $\boldsymbol{\eta}^p_{\bar{c}}$ denoted in the virtual-runway frame $\{p\}$ is given as follows

$$\boldsymbol{\eta}^p_{\bar{c}} = \begin{bmatrix}\mathbf{p}^p_{\bar{c}/p} \; \boldsymbol{\Theta}_{p\bar{c}}\end{bmatrix}^\top \tag{7.3}$$

These states can be differentiated, as the virtual-runway frame is static it follows that $\dot{\boldsymbol{\Theta}}_{np} = \mathbf{0}_{3\times 1}$ and $\dot{\mathbf{R}}^p_n = \mathbf{0}_{3\times 3}$, hence

$$\dot{\mathbf{p}}^p_{\bar{c}/p} = \mathbf{R}^p_n\dot{\mathbf{p}}^n_{\bar{c}/n} \tag{7.4}$$

$$\dot{\boldsymbol{\Theta}}_{p\bar{c}} = \dot{\boldsymbol{\Theta}}_{n\bar{c}} \tag{7.5}$$

Further it should be noted, as the virtual runway frame is static, it follows that no transformation is necessary for the derivatives given in the body frame $\boldsymbol{\nu}^{\bar{c}}$ and $\dot{\boldsymbol{\nu}}^{\bar{c}}$.

$$\boldsymbol{\nu}^{\bar{c}}_p = \boldsymbol{\nu}^{\bar{c}} \tag{7.6}$$

$$\dot{\boldsymbol{\nu}}^{\bar{c}}_p = \dot{\boldsymbol{\nu}}^{\bar{c}} \tag{7.7}$$

7.2 Along-track trajectory

For the purpose of controlling the along-track position of the net according to the fixed-wing UAV one should accelerate the net to a desired velocity and keep this velocity until the impact occurs. However, the net should not go on without any constraints one the along-track distance it is allowed to move. Therefore one should be able to set a desired along-track collision point r_{coll} as illustrated in Figure 7.2 with a desired velocity. In order to keep the net stable and to ensure smooth movements one should constrain the maximum acceleration. The velocity profile should also be critically damped ensuring a stable net during transit.

The virtual runway is given in the North-East plane as shown in Figure 7.2 where the path-frame $\{p\}$ is defined. Further, the virtual runway is bounded by a box such that the multirotor has a limited operation area for safety purposes, the runway is also bounded in height by some lower and upper limit.

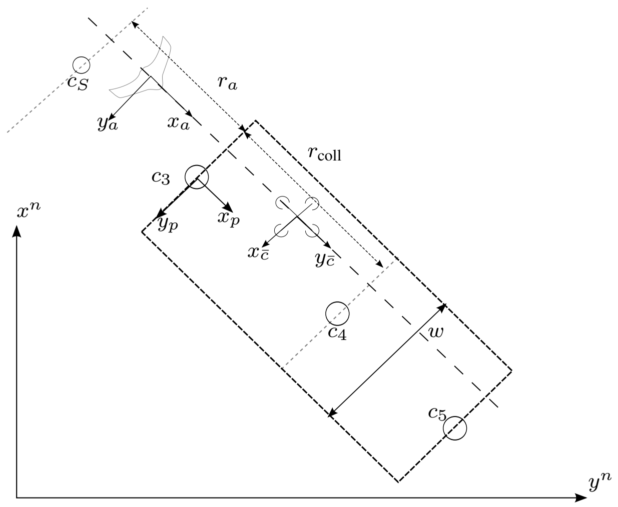


Figure 7.2: Virtual runway path frame $\{p\}$ in the North-East plane with the fixed-wing UAV $\{a\}$ and the multirotor centroid $\{\bar{c}\}$ during recovery, the dotted lines indicates the boundaries of the virtual runway.

It follows that it is necessary to find the Estimated Time of Arrival (ETA) for the desired collision point for both the suspended net and the fixed-wind UAV. Based on the ETAs the desired point c_S to initiate the recovery maneuver can be found. In the following section different methods for generating feasible along-track trajectories will be considered followed by methods for finding the ETA. Lastly the methods will be compared and discussed.

In order to generate a trajectory the following notation will be used; the full reference state $\mathbf{q}(t) \in \mathbb{R}^4$ gives all the time dependent states of the scalar along-track values

$$\mathbf{q}(t) = \begin{bmatrix} x(t) & \dot{x}(t) & \ddot{x}(t) & \dddot{x}(t) \end{bmatrix}^T \tag{7.8}$$

$$= \begin{bmatrix} x(t) & v(t) & a(t) & j(t) \end{bmatrix}^T \tag{7.9}$$

where $x(t)$ defines the along-track position in the path-frame $\{p\}$. The velocity controllers in Section 5.2 will be used in order to follow this trajectory, hence the only the desired velocity and acceleration, $\mathbf{q}_2(t) = v(t)$ and $\mathbf{q}_3(t) = a(t)$ respectivly will be used for along-track velocity control. However, the full state definition is neccesary for this analysis.

The trajectory will be defined between two timestamps t_0 and t_f which gives the time to start and the time at the desired collision point respectively. As the net will be stationary and wait for the aircraft until the time t_0, the initial state $\mathbf{q}(t_0)$ is

$$\mathbf{q}(t_0) = \mathbf{q}_0 = \mathbf{0} \tag{7.10}$$

assuming that the standby-position is at $x = 0$ in the path-frame $\{p\}$. Further at the time of collision the net should have a stationary along-track velocity such that the acceleration and jerk is zero giving

$$\mathbf{q}(t_f) = \mathbf{q}_f = \begin{bmatrix} x_f & v_f & 0 & 0 \end{bmatrix}^T \tag{7.11}$$

where x_f and v_f is the desired collision point and velocity respectively.

7.2.1 Polynomial

As described in Appendix B.1 a feasible trajectory can be generated by fitting the coefficient in a N order polynomial to the given constraints. Given $\mathbf{q}(t_0)$ and $\mathbf{q}(t_f)$ the number of constraints $M^* = \sum_{i=0}^{n} m_i = 4 + 4 = 8$, requiring a polynomial of order $N = m - 1 = 7$.

Following the notation from Appendix B.1, the vectors $\mathbf{b} \in \mathbb{R}^{8x1}$ and $\mathbf{a} \in \mathbb{R}^{8x1}$ is defined as.

$$\mathbf{b} = \begin{bmatrix} (\mathbf{q}_0)^T & (\mathbf{q}_f)^T \end{bmatrix}^T \tag{7.12}$$

$$\mathbf{a} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{bmatrix}^T \tag{7.13}$$

The coefficient vector $\mathbf{a}$ is calculated solving the system as seen in Equation (B.7)–(B.8)

$$\mathbf{a} = \mathbf{M}^{-1}\mathbf{b} \tag{7.14}$$

where $\mathbf{M}$ is defined in Equation (B.11) in Appendix B.2.

By setting $\tau = t - t_0$ the trajectory $\mathbf{q}(\tau)$ can be found

$$\mathbf{q}(\tau) = \mathbf{M}(\tau, 4)\mathbf{a} \tag{7.15}$$

where $\mathbf{M}(\tau, 4)$ is defined in Equation (B.10) in Appendix B.2. Such that the extracted version of Equation (7.15) is as follows.

$$\begin{bmatrix} x(\tau) \\ v(\tau) \\ a(\tau) \\ j(\tau) \end{bmatrix} = \begin{bmatrix} a_0 + a_1\tau + a_2\tau^2 + a_3\tau^3 + a_4\tau^4 + a_5\tau^5 + a_6\tau^6 + a_7\tau^7 \\ a_1 + 2a_2\tau + 3a_3\tau^2 + 4a_4\tau^3 + 5a_5\tau^4 + 6a_6\tau^5 + 7a_7\tau^6 \\ 2a_2 + 6a_3\tau + 12a_4\tau^2 + 20a_5\tau^3 + 30a_6\tau^4 + 42a_7\tau^5 \\ 6a_3 + 24a_4\tau + 60a_5\tau^2 + 120a_6\tau^3 + 210a_7\tau^4 \end{bmatrix} \tag{7.16}$$

For the polynomial case it is not necessary to calculate the trajectory duration, on the contrary the timespan must be specified in order to solve for the coefficient vector $\mathbf{a}$. By specifying the start and end time t_0 and t_f respectively the duration Δt is found

$$\Delta t = t_f - t_0 \tag{7.17}$$

7.2.2 Piecewise

Another approach is to specify a piecewise derivative (e.g. acceleration or jerk), then by integrating analytically a smooth position and velocity trajectory can be obtained. Here follows two examples of such an approach, firstly a piecewise acceleration profile, followed by a piecewise jerk profile. Both profiles must define a velocity profile from zero to the desired recovery velocity, and the acceleration should always be non-negative to avoid fluctuations in the net during acceleration.

Acceleration

Given desired acceleration $a(t)$ a piecewise function can be specified as follows

$$a(t) = \begin{cases} 0 & \text{if} \quad t < t_0 \\ \alpha & \text{if} \quad t \in [t_0, t_1) \\ 0 & \text{if} \quad t > t_1 \end{cases} \tag{7.18}$$

then the velocity profile follows by integration

$$v(t) = \begin{cases} 0 & \text{if} \quad t < t_0 \\ \alpha(t - t_0) & \text{if} \quad t \in [t_0, t_1) \\ \alpha(t_1 - t_0) & \text{if} \quad t > t_1 \end{cases} \tag{7.19}$$

further the position profile $x(t)$ is found by integrating once more

$$x(t) = \begin{cases} 0 & \text{if} \quad t < t_0 \\ \frac{1}{2}\alpha(t - t_0) & \text{if} \quad t \in [t_0, t_1) \\ \frac{1}{2}\alpha(t_1 - t_0) + \alpha(t_1 - t_0)t & \text{if} \quad t > t_1 \end{cases} \tag{7.20}$$

then the full trajectory $\mathbf{q}(t) = \begin{bmatrix} x(t) & v(t) & a(t) & 0 \end{bmatrix}$ is given analytically.

The response of the trajectory can be tuned with the slope parameter α and the duration $t_1 - t_0$. However, the constraints $\mathbf{q}(t_0) = \mathbf{q}_0$ and $\mathbf{q}(t_f) = \mathbf{q}_f$ must also be fulfilled. That is $v(t_f) = v_f$ and $x(t_f) = x_f$.

Jerk

Given desired jerk $j(t)$, a piecewise function can be specified by ramping up to a positive jerk followed by a ramp down to a negative value and back to zero as illustrated in Figure 7.3. This jerk profile ensures a non-negative acceleration profile as the definite integral from t_0 to $t_0 + \Delta t$ of $j(t)$ is zero as the profile is symmetric.

In the same manner as for the piecewise acceleration profile, the jerk profile can be integrated to get the full trajectory state $\mathbf{q}(t)$. The parameters can be tuned adapting to the same constraints $\mathbf{q}(t_0) = \mathbf{q}_0$ and $\mathbf{q}(t_f) = \mathbf{q}_f$.

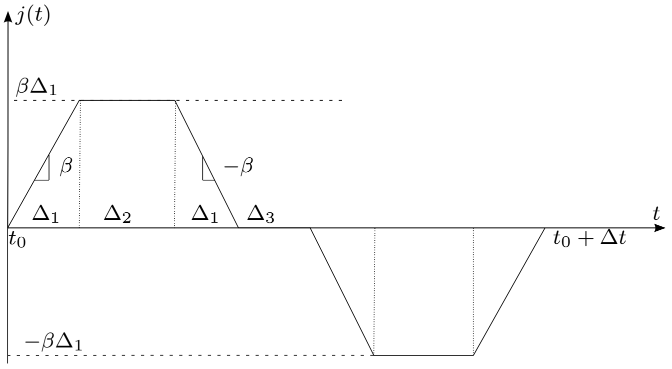


Figure 7.3: Desired jerk profile $j(t)$, $\Delta t = 2(\Delta_1 + \Delta_2 + \Delta_1) + \Delta_3$

Lastly, the trajectory duration must be specified for both expressions, which can be found by solving the equality analytically $x(t_f) = x_f$, giving the total timespan Δt

$$\Delta t = t_f - t_0 \tag{7.21}$$

7.2.3 Reference model

The reference model concept alouds the desired recovery velocity v_f to be used directly into a dynamical system which will smooth the reference step to a feasible reference trajectory. Furthermore, by analyzing the reference model the adaptive scheme given in Section 5.2.4 can be used for along-track control. For the following section a scalar reference model will be considered, as the along-track scalar response need to be analyzed in order to find an expression for the ETA.

A 1-st order reference model as in Section 5.2.4 was used such that the same adaptive scheme can be used as an alternativ velocity controller. The 1st-order reference model for the velocity along-track velocity v_r can be given on the form as discussed in Section 5.2.4

$$T\dot{v}_r + v_r = r \tag{7.22}$$

where $r = v_f$ is the desired velocity at recovery and T is the desired time-constant.

For further analysis the differential equation Equation (7.22) must be solved analytically, and it can be shown that the solution gives the velocity $v_r(t)$ assuming $r = v_f$ is a step engaged at $t = t_0$

$$v_r(t) = v_f\left(1 - e^{\frac{t}{T}}\right) \tag{7.23}$$

Then the trajectory vector is given as follows

$$\mathbf{q}(t) = \begin{bmatrix}\int_{t_0}^{t} v_r(\tau)d\tau & v_r(t) & \dot{v}_r(t) & \ddot{v}_r(t)\end{bmatrix}^\top \tag{7.24}$$

Solving the equality $x(t_f) = \int_{t_0}^{t_f} v_r(\tau)d\tau = x_f$ analytically gives t_f and lastly the duration Δt is found

$$\Delta t = t_f - t_0 \tag{7.25}$$

7.2.4 Comparing

Lastly the methods are compared using the parameters in Table 7.1. The analytical expressions as presented were solved using symbolic toolboxes, such that no explicit expressions will be given here. Furthermore, it should be noted that the duration Δt is only specified for the polynomial approach. Hence, the duration of the other methods is not specified explicitly, however, given a set of parameters the duration Δt can be calculated explicitly given the desired trajectory. The resulting responses are shown in Figure 7.4, as expected,

$\Delta t = t_f - t_0$	8 s (for polynomial)
x_f	20 m
v_f	4 m/s
a_{max}	1 $\mathrm{m/s^2}$
β	1 $\mathrm{m/s^4}$
T	1 s

Table 7.1: Trajectory parameters

the time required to reach the target setpoint x_f varies depending on the method, however all methods fulfill the requirements for the desired trajectory as specified in the beginning of this section.

Comparing the methods reveals the fact, that only the piecewise trajectories (piecewise acceleration and jerk) are able to hold the acceleration constraints. This is because the other methods has no explicit methods to constrain the acceleration, other than setting the parameters such that the constraint is held. It should also be noted that the reference model requires a large step in the initial acceleration in order to follow the trajectory, which is not optimal, but can be solved by reducing the timeconstant T of the model. Summing up, the piecewise jerk and polynomial trajectory gives the smoothest trajectory which the multirotors is more likely to be able to follow. A simulation study for these methods is given in Section 7.6.1.

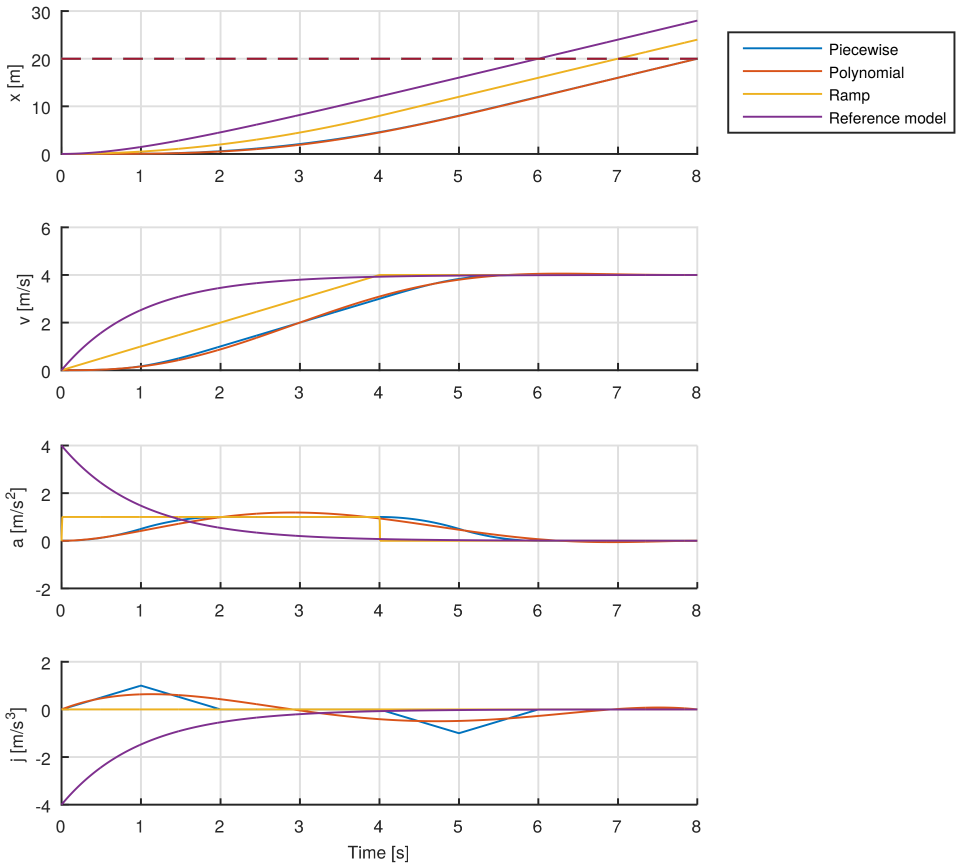


Figure 7.4: Comparing reference trajectories, target position x_f marked with the dotted red line

7.2.5 Fixed-Wing ETA

All methods so far have found analytical expression for the timespan Δt required to reach the recovery point, however, the ETA for the fixed-wing ETA_a must be derived such that the timestamp to start the along-track trajectory profile can be found.

Multiple methods can be used in order to find the ETA_a. These includes among others analyzing the landing profile and the desired velocity trajectory in the same manner as done for the centroid. However, these methods will not be consider in this thesis. For this purpose, the along-track velocity of the fixed-wing is assumed to be constant such that ETA_a can be calculated by monitoring the along-track path position x_a and velocity v_a.

$$ETA_a = \frac{|x_a|}{v_a} \tag{7.26}$$

The ETA for the centroid $ETA_{\bar{c}}$ is given as the timespan Δt added to the timestamp at the start of the trajectory t_0

$$ETA_{\bar{c}} = t_0 + \Delta t \tag{7.27}$$

in order to reach the recovery-point at the same time as the fixed-wing the following equality must hold $ETA_{\bar{c}} = ETA_a$. Then the time to start the along-track trajectory profile t_0 is found as

$$t_0 = ETA_a - \Delta t \tag{7.28}$$

7.3 Cross-track control

This section shows the implementation of a steering-law which were applied during the net recovery maneuver in order to control the position of the multirotor according to the approaching fixed-wing UAV.

The purpose with the pure-pursuit guidance is to control the vehicle to a desired position as the LOS - steering law discussed in Section 6.1. However, the purpose with this controller is to only control the position in the cross-track plane in the path frame $\{p\}$. Hence a new frame $\{p_*\} \in \mathbb{R}^2$ as can be defined as the cross-track plane in the path frame $\{p\}$, such that there exist a mapping for a vector $r^p = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}^T$ to $r^{p_*} = r^p_{2:3}$. The purpose is to render the position $\mathbf{p}^{p_*}$ to the target position $\mathbf{p}^{p_*}_d$, such that the error $\tilde{\mathbf{p}}^{p_*}$ is minimized

$$\tilde{\mathbf{p}}^{p_*} := \mathbf{p}^{p_*}_d - \mathbf{p}^{p_*} \tag{7.29}$$

this gives a target tracking path following scheme as discussed in e.g. [Fossen, 2011, Ch. 10].

The desired position $\mathbf{p}^{p_*}_d = \mathbf{p}^{p_*}_a$ is defined as the current position of the fixed-wing UAV projected to the cross-track plane. Using the notation from Section 6.1 the position of the fixed-wing UAV in the path-frame $\mathbf{p}^p_a$ is given as

$$\mathbf{p}^p_a = \mathbf{R}^p_n \mathbf{p}^n_a = (\mathbf{R}^n_p)^T \mathbf{p}^n_a \tag{7.30}$$

which gives the cross-track position $\mathbf{p}^{p_*}_a = \mathbf{p}^p_{a,2:3}$ of the fixed-wing UAV.

Further the desired position is bounded by the rectangle defined by a height h and width w with origin in the origin of the cross-track path frame $\{p_*\}$. Then the fixed-wing UAV position $\mathbf{p}_d^{p_*}$ is saturated to get the desired position $\mathbf{p}_d^{p_*}$, note that the desired centroid height is given with a certain offset z_L. This offset is the nominal height difference between the centroid and the center of the net as illustrated in Figure 7.5. As the center of the net will fluctuate during the maneuver it is desired to define a static offset rather than measuring the actual height difference.

$$\mathbf{p}_{d,1}^{p_*} = \text{sat}(\mathbf{p}_{a,1}^{p_*}, -\frac{w}{2}, \frac{w}{2}) \tag{7.31}$$

$$\mathbf{p}_{d,2}^{p_*} = \text{sat}(\mathbf{p}_{a,2}^{p_*} + z_L, -\frac{h}{2}, \frac{h}{2}) \tag{7.32}$$

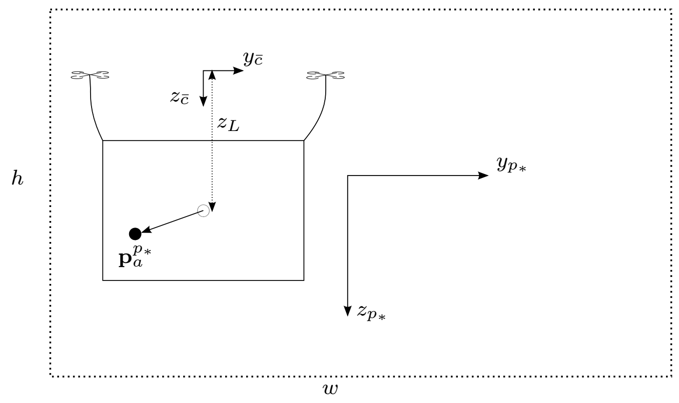


Figure 7.5: The vehicles in the cross-track frame $\{p_*\}$, bounded by the dotted virtual runway with height h and width w. Where the black dot is the fixed-wing and the circle is the center of the net. The height difference between the centroid $\{\bar{c}\}$ and the net center is given as z_L.

Equation (7.29) is minimized by by modifying the pure-pursuit guidance scheme as discussed in e.g. [Fossen, 2011, Ch. 10], such that the desired velocity in the cross-track plane $\mathbf{v}_d^{p_*}$ is found by the following steering law

$$\mathbf{v}_d^{p_*} = \mathbf{K}_{p,p}\tilde{\mathbf{p}}^{p_*} + \mathbf{K}_{d,p}\dot{\tilde{\mathbf{p}}}^{p_*} + \mathbf{K}_{i,p}\int_0^t \tilde{\mathbf{p}}^{p_*}\, dt \tag{7.33}$$

where the matrices $\mathbf{K}_{i,p} \in \mathbb{R}^{2x2}$ for $i \in \{p, i, d\}$ are diagonal matrices with all elements positive along the diagonal. This gives in total the desired velocity in the path-frame as

$$\mathbf{v}_{d,1}^{p} = \mathbf{v}_{d,x}^{p}(t) \tag{7.34}$$

$$\mathbf{v}_{d,2:3}^{p} = \mathbf{v}_d^{p_*} \tag{7.35}$$

where $\mathbf{v}_{d,x}^{p}(t)$ is given by the desired along-track velocity trajectory $\mathbf{q}_2(t)$. Then the desired velocity in the local $\{n\}$ frame can be found by the following rotation

$$\mathbf{v}_d^n = \mathbf{R}_p^n \mathbf{v}_d^p \tag{7.36}$$

7.4 Position hold

This section briefly presents a position controller in order to hold the position at the start of the virtual runway[1]. The control law in Equation (7.33) is modified with a constant reference point, additionally the along-track position is controlled as well. Such that given the position error $\tilde{\mathbf{p}}^p := \mathbf{p}_d^p - \mathbf{p}^p$ with the desired position $\mathbf{p}_d^p = \mathbf{0}_{3x1}$[2] one get the PID control law in Equation (7.37).

$$\mathbf{v}_d^p = \mathbf{K}_{p,ph}\tilde{\mathbf{p}}^p + \mathbf{K}_{d,ph}\dot{\tilde{\mathbf{p}}}^p + \mathbf{K}_{i,ph}\int_0^t \tilde{\mathbf{p}}^p \, dt \tag{7.37}$$

where the matrices $\mathbf{K}_{i,ph} \in \mathbb{R}^{3x3}$ for $i \in \{p, i, d\}$ still are diagonal matrices with all elements positive along the diagonal. Then the velocity $\mathbf{v}_d^n$, is still given by Equation (7.36).

7.5 Supervisor

The open- and closed-loop schemes presented can be combined to present an overall solution of the recovery maneuver problem. However, this requires an superior controller responsible for enabling the desired controllers and trajectories by controlling the reference input based on the current state of the fixed-wing. Such an superior controller can be realized using a *Finite-State-Machine*, wherein a set of states of the maneuver can be defined. Each state can then enable a certain controller and reference. Furthermore, the current position and velocity can be used to trigger a certain transition from one state to another, such that only one state will be active at a certain time.

This *Finite-State-Machine*, or simply the state-machine for short can be defined as seen in Figure 7.6. After engaging the recovery maneuver, the supervisor will initialize (INIT) and wait for all vehicles to connected. Then the multirotors will be instructed to move to the start of the virtual runway at checkpoint c_3 as illustrated in Figure 7.2. In this (STANDBY) state the position controller from Section 7.4 will be used to ensure the multirotor can hold the current position with external disturbances such as wind present.

The supervisor should do a transition to (APPROACH) when the fixed-wing is approaching towards the runway from behind within some along- and cross-track position and velocity boundaries. The cross-track position and velocity boundaries is given as $\mathbf{p}_{approach}^{p*}$ and $\mathbf{v}_{approach}^{p*}$ respectivly. At (APPROACH) the cross-track controller in Section 7.3 will control the cross-track position of the centroid (and thereby the position of the net) according to the current desired cross-track position instructed by the fixed-wing.

[1]The start of the runway and the origin of the path frame $\{p\}$ is given by checkpoint c_3 in Figure 7.2
[2]Position set to the origin of the path-frame, but can in general be any point in the given path frame $\{p\}$.

Meanwhile the supervisor will estimate the time of arrival of the fixed-wing ETA_a, such that desired time to start the recovery (START) and engage the along-track trajectory is as given by Equation (7.28). As the fixed-wing approaches the net, three different outcomes is possible: Hopefully, a successful recovery will be conducted (CATCH), else the fixed-wing may pass the net without a recovery or reach the end of the runway, both resulting in a stop of the maneuver (STOP).

After a catch, the cross-track reference point and the along-track velocity will be fixed in order to not induce extra swinging motion in the net. Another approach is to stop the along-track velocity control after the catch allowing the along-track swinging motion to decline without external disturbance, however, this method has not been investigated further. In order to do a second attempt for a recovery, the whole maneuver can simply be restarted.

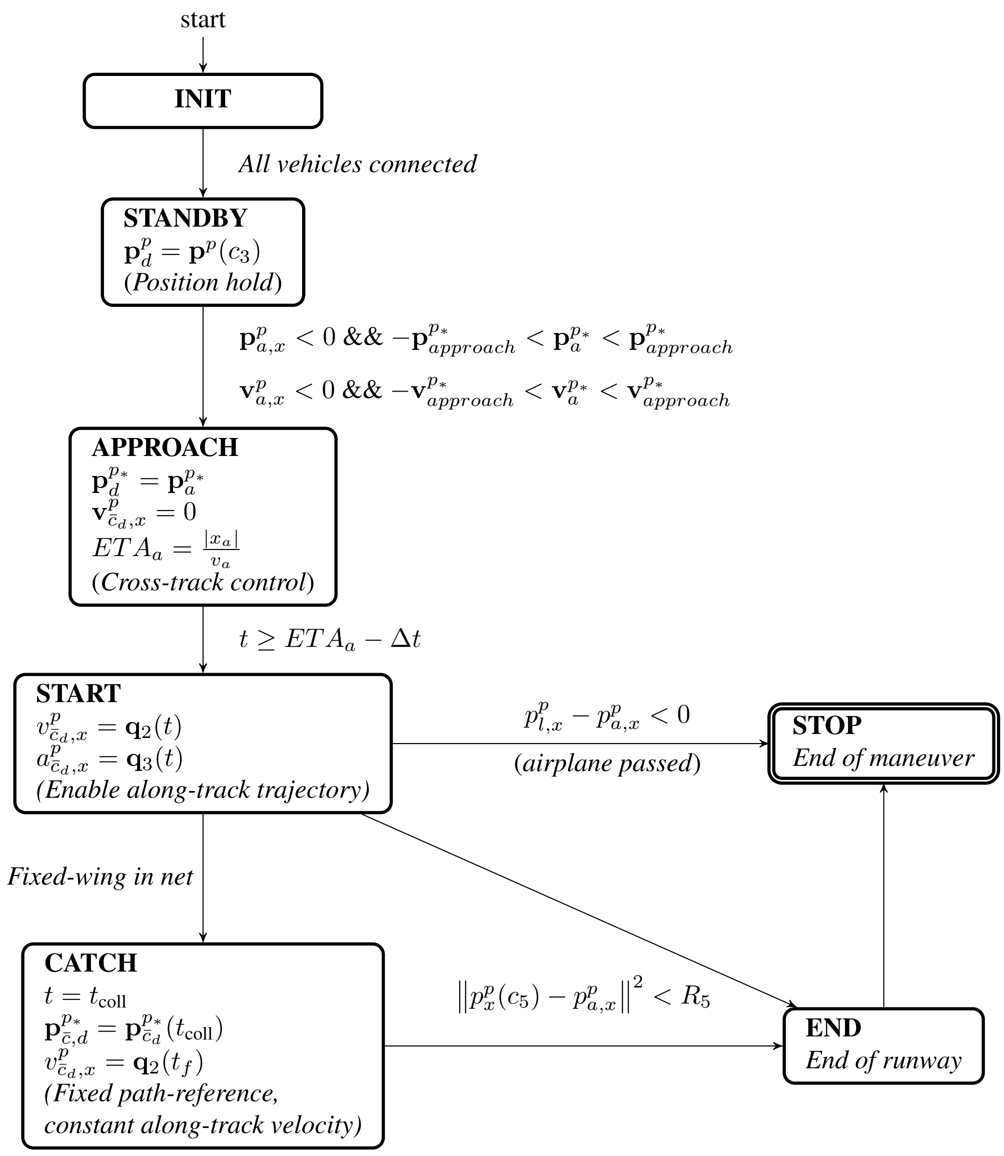

start
INIT
All vehicles connected
STANDBY
$\mathbf{p}_d^p = \mathbf{p}^p(c_3)$
(Position hold)
$\mathbf{p}_{a,x}^p < 0$ && $-\mathbf{p}_{approach}^{p_*} < \mathbf{p}_a^{p_*} < \mathbf{p}_{approach}^{p_*}$
$\mathbf{v}_{a,x}^p < 0$ && $-\mathbf{v}_{approach}^{p_*} < \mathbf{v}_a^{p_*} < \mathbf{v}_{approach}^{p_*}$
APPROACH
$\mathbf{p}_d^{p_*} = \mathbf{p}_a^{p_*}$
$\mathbf{v}_{\bar{c}_d,x}^p = 0$
$ETA_a = \frac{|x_a|}{v_a}$
(Cross-track control)
$t \geq ETA_a - \Delta t$
START
$v_{\bar{c}_d,x}^p = \mathbf{q}_2(t)$
$a_{\bar{c}_d,x}^p = \mathbf{q}_3(t)$
(Enable along-track trajectory)
$p_{l,x}^p - p_{a,x}^p < 0$
(airplane passed)
STOP
End of maneuver
Fixed-wing in net
CATCH
$t = t_{\text{coll}}$
$\mathbf{p}_{\bar{c},d}^{p_*} = \mathbf{p}_{\bar{c}_d}^{p_*}(t_{\text{coll}})$
$v_{\bar{c}_d,x}^p = \mathbf{q}_2(t_f)$
(Fixed path-reference, constant along-track velocity)
$\left\|p_x^p(c_5) - p_{a,x}^p\right\|^2 < R_5$
END
End of runway

Figure 7.6: State machine for the recovery maneuver

Lastly, the recovery control structure is summarize in Figure 7.7, illustrating the connections between the along- and cross-track schemes. Here the fixed-wing UAV states flows from the left, and on the right hand side, the desired acceleration and velocity of the centroid are given.

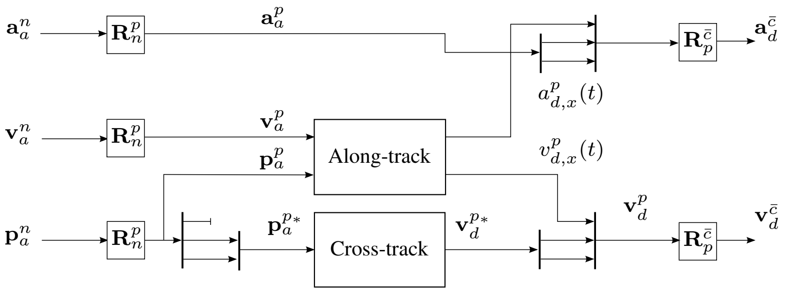


Figure 7.7: Recovery structure figure

7.6 Simulations

7.6.1 Trajectory comparing

The desired along-track trajectories as summarized and compared in Section 7.2.4 where applied on a single multirotor with the velocity controller from Section 5.2. The system equation were solved using RK4 running at 50 Hz, with the desired along-track trajectory parameters as given in Table 7.2.

$\Delta t = t_f - t_0$	8 s (for polynomial)
x_f	20 m
v_f	4 m/s
a_{max}	1 m/s^2
β	1 m/s^3

Table 7.2: Trajectory comparing simulation parameters

Furthermore, a fixed-wing were instructed to hold a constant course and height with a surge speed of $u_a = 18$ m/s towards the multirotor. The the along-track trajectories were enabled when the simulation time passed the start time as calculated in Equation (7.28).

The resulting along-track responses are given in Figure 7.4. Comparing with the responses in Section 7.2.4 one can see that all trajectories ensures that the multirotor reaches the desired recovery point at approximately the same time stamp. Hence given that the estimated time-arrival factor of the fixed-wing ETA_a can be estimated, one should be able to control the desired recovery point with these open loop trajectories. It should be noted that the first order reference model gives an aggressive acceleration profile, and comparing with the desired velocity and acceleration trajectory in Figure 7.4 one can also see that they are not feasible, as the multirotor is not able to follow them. However, for the purpose of using a MRAC scheme the model and estimation will be necessary.

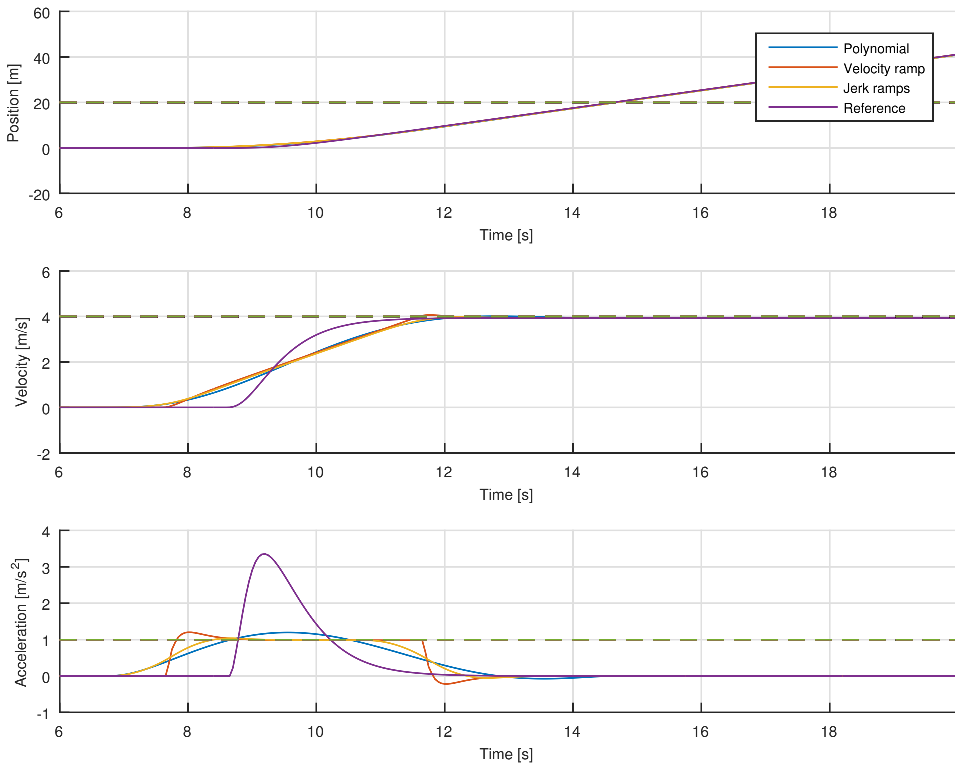


Figure 7.8: Comparing along-track responses, the dotted lines indicates the constraints.

7.6.2 Recovery maneuver

An overall recovery maneuver simulation study was conducted using the polynomial along-track trajectory as introduced in Section 7.2.1 with the desired parameters as given in Table 7.3.

$\Delta t = t_f - t_0$	8 s
x_f	20 m
v_f	4 m/s

Table 7.3: Recovery maneuver simulation parameters

Furthermore, the cross-track controller from Section 7.3 with $\mathbf{K}_{p,p} = 2\mathbf{I}_{2\times 2}$ and $\mathbf{K}_{p,d} = 0.2\mathbf{I}_{2\times 2}$ was used. Further, the supervisor in Section 7.5 handled the different controllers and references. An virtual-runway were defined with length 50 m and a cross-track plane of dimensions 5 m $\times$ 5 m, lastly, the fixed-wing was instructed to fly in parallel with the virtual-runway and in the center of the cross-track plane with an along-track surge speed of 18 m/s.

An overview of the resulting simulation are shown in Figure 7.9, where the bodies are shown with the virtual-runway after the impact with the fixed-wing suspended in the net[3].

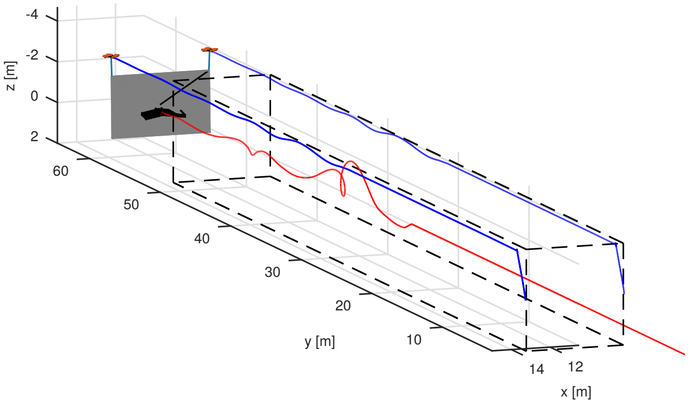
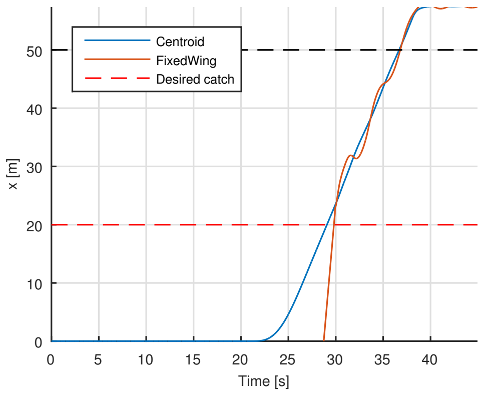


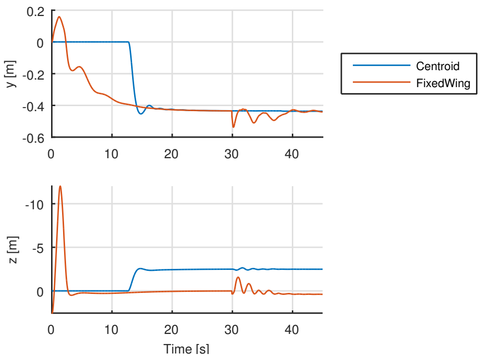
Figure 7.9: Position and orientation of the bodies in $\{n\}$, the center multirotor gives the centroid, the black dotted line gives the boundaries of the virtual-runway.

In Figure 7.10 the feasibility of the recovery scheme is proven. Firstly the multirotors are able to reach the desired recovery point at the same time as seen in Figure 7.10a, then Figure 7.10b illustrates how the cross-track controller ensures that the fixed-wing has an impact with the net. One should note that the cross-track controller does not start immediately as the supervisor instruct the multirotor to wait until the fixed-wing are approaching the virtual-runway. Also, the offset in height gives the height difference between the centroid and the center of the suspended net.

[3]A movie of the recovery simulation can be seen *Simulations/Recovery maneuver.mp4* in the *Digital Appendix*.



(a) Along-track positions



(b) Cross-track positions

Figure 7.10: Positions of centroid and fixed-wing in the *Virtual-Runway* frame $\{p\}$

Lastly, the performance of the velocity controller are illustrated in Figure 7.11, where one can see that the multirotor is able to follow the polynomial trajectory, as well as controlling the cross-track velocity. The step in the desired cross-track velocity follows from the cross-track deviation between the centroid and the fixed-wing when the cross-track controller is enabled by the supervisor.

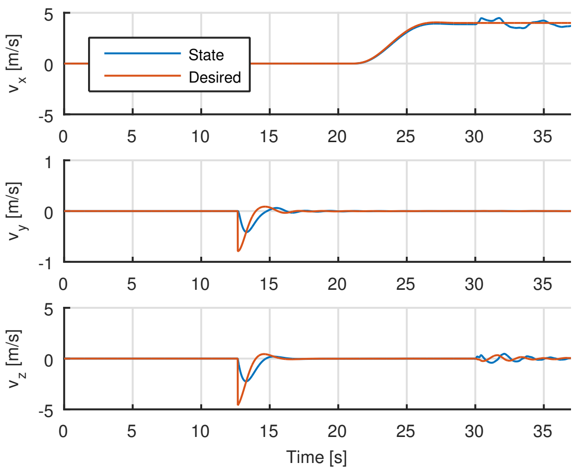


Figure 7.11: Centroid velocities denoted in $\{p\}$

7.6.3 MRAC

As for the path control in Chapter 6 a simulation study with the MRAC as velocity controller was conducted for the recovery maneuver. Here RK4 at 100 Hz was used with the key parameters as given Table 7.4.

x_f	20 m
v_f	2 m/s
a_{max}	0.5 m/s^2
u_a	18 m/s

(a) Net recovery simulation parameters

$\mathbf{Q}$	$3\mathbf{I}_{3\times3}$
$\hat{\mathbf{K}}_0$	$-3\mathbf{I}_{3\times3}$
$\hat{\mathbf{L}}_0$	$3\mathbf{I}_{3\times3}$

(b) MRAC parameters

Table 7.4: MRAC recovery key parameters

As in Section 7.6.2 the overall maneuver using the MRAC as the velocity controller is illustrated in Figure 7.12[4]

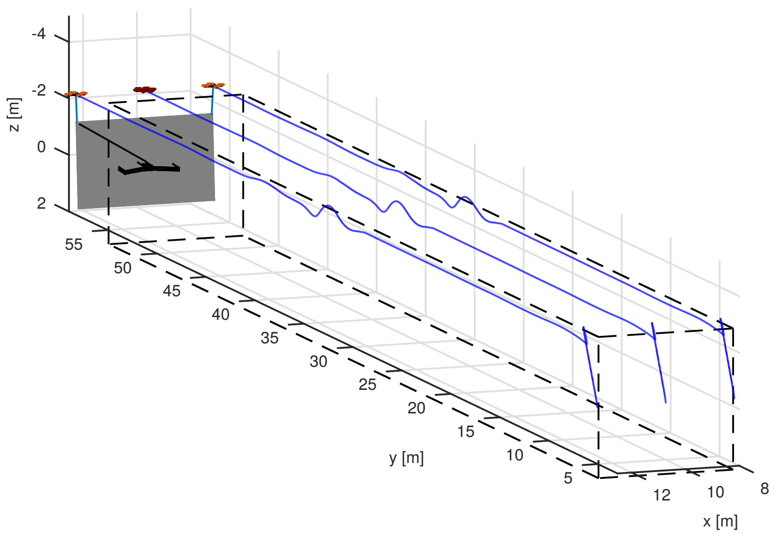
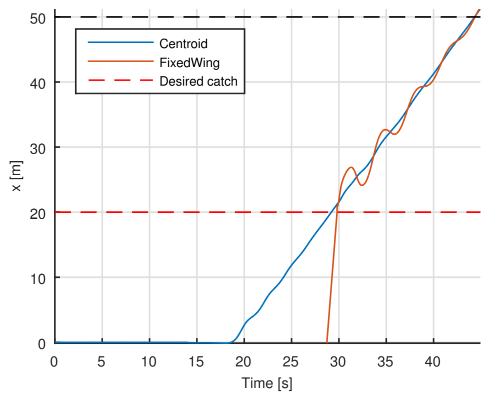


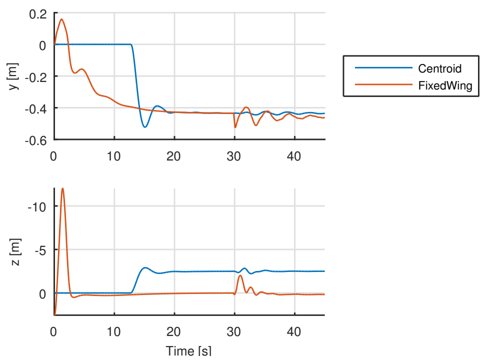
Figure 7.12: Positions of the vehicles in the $\{n\}$ frame, the black dotted line gives the boundaries of the virtual-runway

The same along- and cross-track response can be observed in Figure 7.13, the MRAC is able to control the centroid to the desired recovery point at when the fixed-wing hits the suspended net.

[4]A video of the recovery maneuver using the MRAC can be seen in *Simulations/Recovery - MRAC.mp4* in *Digital Appendix.*



(a) Along-track positions



(b) Cross-track positions

Figure 7.13: Positions of centroid and fixed-wing in the virtual-runway frame $\{p\}$

The centroid velocity in the virtual-runway frame given in Figure 7.14 illustrate the issues as discussed in Section 6.3.2, the MRAC is able to somehow follow the cross-track velocities with some fluctuation and lag compared to the desired trajectory, however, the along-track velocities oscillate towards the desired set-point after the collision due to the fluctuating net.

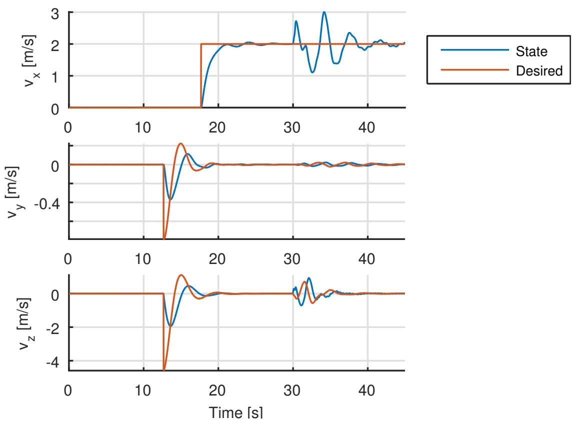
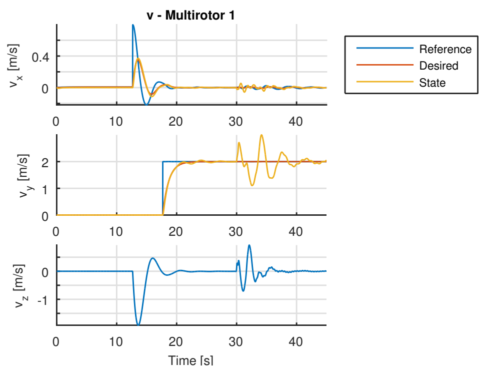
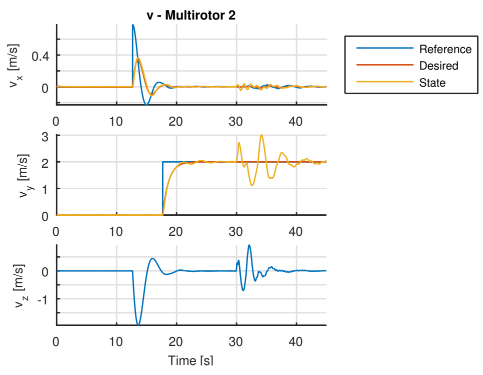


Figure 7.14: Centroid velocities denoted in $\{p\}$

Moreover, by observing the velocities of the individual multirotor in Figure 7.15 the same fluctuations can be observed after the collision. The cross-track velocities v_x (as the virtual runway is facing east) does not suffer from the fluctuations. On the other hand it should be noted that the first order reference model creates a more feasible reference trajectory. However, a first-order model will also increase the delay, such that the agility of the multirotors are reduced, the amount of delay can be adjusted with the time constant of the reference model.



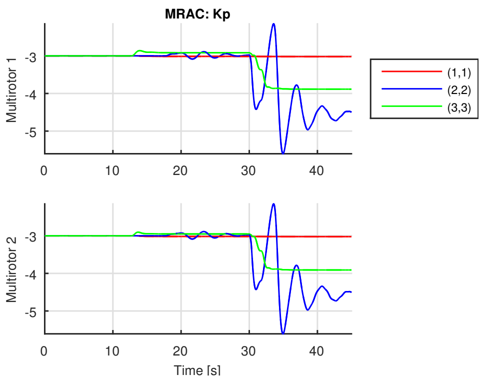
(a) Multirotor 1



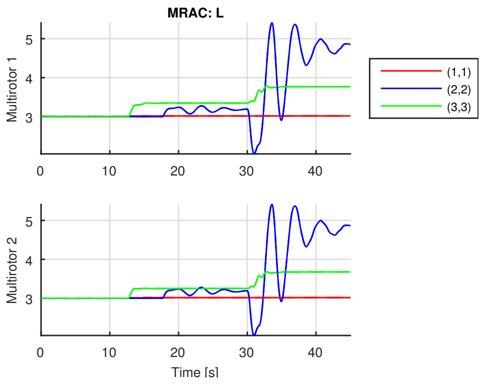
(b) Multirotor 2

Figure 7.15: Velocities of the multirotors in $\{n\}$ during the recovery maneuver, comparing the reference, desired and the actual state

By investigating the adaptive gains in Figure 7.16, one can see that the along-track gain components $(2, 2)$ fluctuates until the parameters converge towards the end of the simulation, specially after the collision. This is due to the oscillations of the net after the impact with the fixed-wing, where the adaptive laws tries to compensate. In this case, it seems that the adaption law is somehow to slow.



(a) Diagonal elements of K_p



(b) Diagonal elements of L

Figure 7.16: Multirotor adaptive gains during the recovery maneuver

An important aspect with the MRAC scheme, is that the inputs need to fulfill the concept of *persistent excitation* (PE). The general requirement for parameter convergence in an adaptive scheme is that the input signal is sufficiently rich as discussed in detail in [Ioannou & Sun, 2012]. For this specific MRAC scheme the inputs to the parameter estimation laws are the error state $\mathbf{e} = \mathbf{x} - \mathbf{x}_r$ for both laws, further the $\mathbf{x}$ and $\mathbf{r} = \mathbf{v}_d^n + \mathbf{v}_{i,df}^n$ are inputs for the adaption laws for $\mathbf{K}(t)$ and $\mathbf{L}(t)$ respectively. The virtual-runway along-track component reference signal r_y consists of a step from zero to the desired recovery velocity, which do not ensure PE. However, the formation law compensate greatly in this axis, as the suspended net and the desired link vector are parallel to the cross-track plane. Therefore, the adaption law converges for the $(1, 1)$ components converges fast as seen in Figure 7.16. On The

Chapter 8

Experimental setup

For prototyping the system was implemented and simulated in Matlab. However, in order to test the system on an embedded hardware platform for field experiments the methods implemented in Matlab must be ported to a more low-level software framework as Matlab is not suitable to do this directly. Therefore the control algorithms for the multirotor were implemented and written in the C++ programming language using the DUNE framework. This chapter explains the structure of the software implementation, for details around the practical usage of the system the reader is referred to the software user manual attached in the *Digital Appendix*.

The overall architecture of the system implementation is presented in Figure 8.1. The typical program flow is as follows: a plan is generated using the Neptus GCS which is dispatched to the respective vehicle. Each plan contains one or more maneuvers mostly attatched to one or more waypoints. The DUNE framework handles the internal switching between these maneuvers by reading the current status of the maneuver in the *IMC::ManeuverControlState* message as seen in the box leftmost in Figure 8.1, here the operator of the GCS can choose to start or stop a plan as desired.

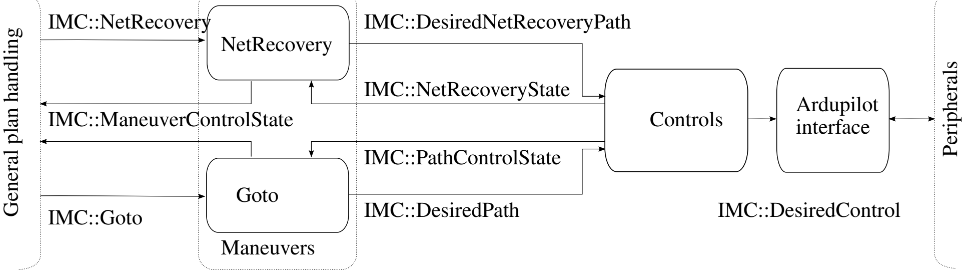


Figure 8.1: DUNE system architecture overview

DUNE supports multiple standard maneuvers, as e.g. the *Goto*-maneuver which sends a desired global waypoint[1] with a desired height and speed set-point as defined in the *IMC::DesiredPath* message. This data is received from the plan handler through the maneuver message *IMC::Goto*. Furthermore, a controller can be implemented using this information, and update the maneuver-task with the current state *IMC::PathControlState* regularly. It should be noted that the vehicle controllers uses the local $\{n\}$ reference system, hence all waypoints received from the Neptus GCS must be converted to this local frame. All vehicles uses a common reference point as the origin of this frame such that a conversion can be conducted.

However, for the purpose of implementing a recovery maneuver, a custom *NetRecovery* maneuver task were implemented. In order to keep the same hierarchically structure the maneuver reads the *IMC::NetRecovery* maneuver message containing all data to perform a specific recovery, a controller is then engaged and the maneuver data is forwarded through the *IMC::DesiredNetRecoveryPath* message. Lastly the recovery controller sends a *IMC::NetRecoveryState* message such that the maneuver task *NetRecovery* can monitor the progress and terminate the maneuver when done.

Lastly, the controllers dispatches the desired control force (in the local $\{n\}$ frame) to be applied on the vehicle to the Ardupilot through a interface implemented in the DUNE framework. The Ardupilot handles the attitude control, followed by the control allocation to distribute the desired angular velocity to each ESC controlling the speed of the rotors.

In the following sections the structure of the system will be discussed. Moreover, the different controllers will be discussed; the two different path controllers implementing the guidance scheme during the overall mission for the recovery and the transport maneuver. The methods are discussed in detail in Section 8.2.1 and Section 8.2.2 respectively. The output from these controllers is then sent to the velocity control responsible for sending the desired force to the Ardupilot interface as seen in Section 8.2.3.

8.1 Architecture

Firstly, in order to present the controller implementation, the structure of the overall system must be discussed in some detail. The inter-vehicle system must communicate and agree on a set of configurations, furthermore, the navigation solutions must be transformed to a common reference system.

8.1.1 Configuration

A configuration messaged named *IMC::CoordConfig* is sent from the master multirotor to the slave multirotors and the fixed-wing UAV. The message is dispatched from the *Coordination Configuration* task as seen in Figure 8.2 to distribute setup both on system boot-up,

[1]The position is given in latitude and longitude defined in the World Geodetic System 1984 (WGS-84) datum

but also for *on the fly* configuration[2]. This is very convenient especially for the multirotor setup, as the desired initial formation, tuning parameters and such can be changed dynamically. Also, as it will be seen in Figure 8.4, the configuration will be used to distribute the common global reference origin for the inertial reference frame $\{n\}$ for all vehicles.

The distribution of the different tasks implemented is also illustrated in Figure 8.2. As seen the master multirotor, in addition to running the configuration task, it is responsible for the overall path controllers, and dispatches the references to the slave(s). Still, all multirotors must run the velocity controller *Force* dispatching the desired force. The *Force* task consumes the current states of the other vehicles and itself dispatched from the *Local State Transport* task, as well as the states of the centroid dispatched from *Centroid* in order to control the formation. Lastly, the states of the fixed-wing UAV is dispatched to the master vehicle from the *Local State Transport* task running on the fixed-wing in order to do the recovery maneuver.

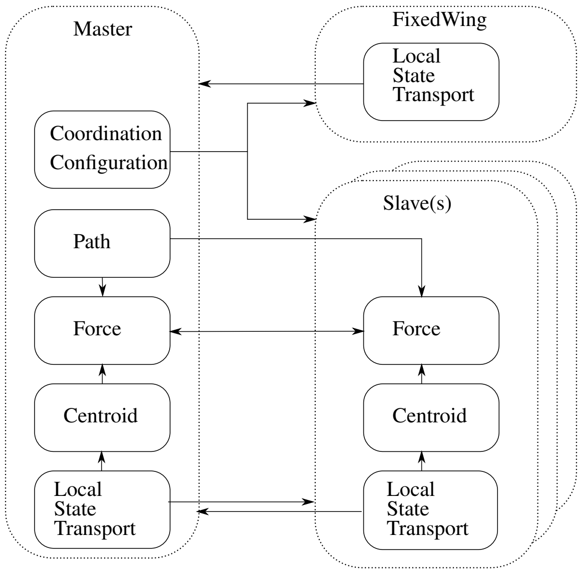


Figure 8.2: DUNE on the different platforms

[2]The ability to configure the system whilst running on the embedded computer.

Furthermore, the DUNE system incorporate a hierarchically system where the different controllers are enabled consecutively based on which controllers it need to run. That is, each controller task is enabled by reading a *IMC::ControlLoops* message containing a flag telling which control-loop type to enable, if a controller read the message and it is of the correct type it will enable itself. Furthermore, if the given controller needs a certain control-loop a new *IMC::ControlLoops* will be dispatched with the specific flag set. For this system the specific flags of type *IMC::CL_<desired controller>* are seen in Figure 8.3. The reader should note that each guidance controller is of the type *IMC::CL_PATH*, requiring a second flag preventing both controllers from being activated. This maneuver specific flag is set from the GCS such that for each maneuver, the operator must choose the desired controller.

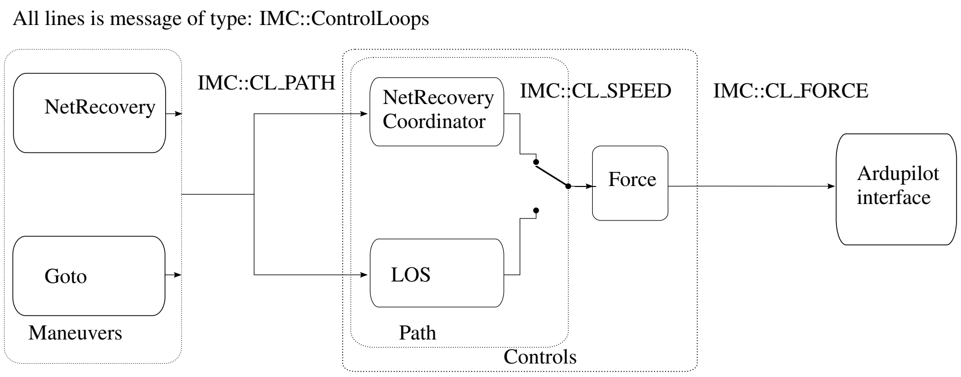


Figure 8.3: DUNE control-loops overview

8.1.2 Transport layers

The feedback signal is dispatched from the navigation and sensor -systems on each vehicle. The full state of the system, including among other the global position, local position, velocity and orientation is contained in the *IMC::EstimatedState* message, lastly the *IMC::Acceleration* message contains the acceleration data.

As the *IMC::CoordConfig* contains a fallback-reference the agent can choose to either use the local states directly from the *IMC::EstimatedState* message or transform the local states using the fallback reference. When using the RTK GNSS[3] navigation solution, the reference point will be the base-station which is common for all vehicles using the same station, then it is not necessary to transform the local navigation solution as it is already in the same reference system. However, if the RTK navigation solution is not valid, the local GPS solution for the vehicle will be used. Then one must transform all solutions to the common fallback reference point, as the reference points for the different vehicles will most likely deviate from each others.

[3]*Real Time Kinematic Global Navigation Satellite System* as introduced in Chapter 3

The new transformed *IMC::EstimatedState* message together with the *IMC::Acceleration* will be incorporated in a new message *IMC::EstimatedLocalState* as seen in Figure 8.4. Further the *IMC::EstimatedLocalState* from the *Local State Transport* task will be sent locally between the vehicles as seen in Figure 8.2. Lastly the centroid state message will be constructed locally on each multirotor in the *Centroid* task.

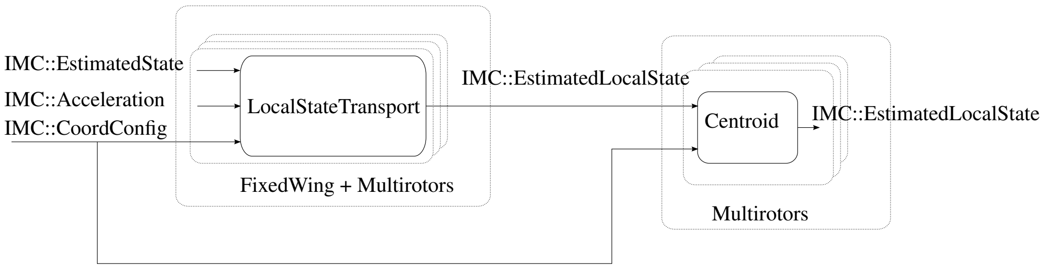


Figure 8.4: Overview of DUNE feedback

8.1.3 Transport configuration

As described in Chapter 3 the inter-vehicle communication is executed using the UDP transport protocol based on the Internet Protocol (IP). In order to manually configure the communication all transport layers must connect to the transport layer on the destination vehicle. Each layers is identify with the IP address of the vehicle and a certain port, if simulating all vehicles on the same computer all IPs will be the same, and unique port numbers must be specified. It is troublesome to manually configure such a setup, therefore a *DiscoverVehicle* task was implemented responsible for setting up the IPs and ports for all transport layers. By listening for newly connected vehicles and linking the IP and port to each transport layer automatic configuration where achieved, furthermore, by listening for lost connection, the system is able to stop data transport to the disconnected vehicles.

8.2 Controllers

The overall scheme of the controllers implemented are presented in Figure 8.5. It should be noted that a master/slave model is chosen, such that the master multirotor does all the guidance calculations (*Path Control*) and dispatches the configuration message as illustrated in Figure 8.2. However, the fixed-wing acts as the uppermost master, dictating the overall maneuver flow. Note that all path-controllers uses the centroid *IMC::EstimatedLocalState* dispatched from the *Centroid* task. The coordinated velocity control is the only task requiring the states from all individual multirotors.

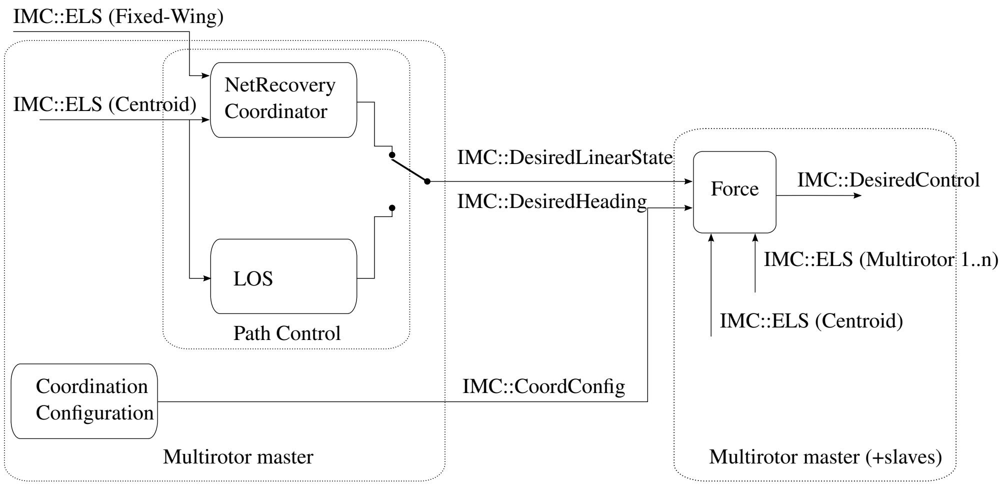


Figure 8.5: Structure of DUNE controllers implementation, IMC::ELS is short for IMC::EstimatedLocalState

8.2.1 Recovery Coordinator

The recovery coordinator *NetRecovery Coordinator* as seen in Figure 8.5 implements the coordinated recovery maneuver of the fixed-wing. The routine monitors the states of the multirotors and the fixed-wing in order to control the movements of the net according to the fixed-wing motions. It implements the recovery maneuver as discussed in Chapter 7 which uses the the different controllers as supervised by the supervisor as discussed in Section 7.5.

8.2.2 Path Control

The path-controller *LOS* as seen in Figure 8.5 implements the guidance scheme as introduced in Chapter 6. Furthermore, it ensure a smooth movement of the net combining the LOS guidance law from Section 6.1 and the reference simulator in Section 6.2. The controller is implemented as a periodic task with a fixed frequency ensuring a discrete controller domain.

The LOS guidance law in Section 6.1 is implemented directly, using a set of waypoints defined from the Neptus GCS to define the paths. As the reference simulator is a dynamical system, and as it is not possible to solve the system analytically a numerical solution is required. A fixed-step Runge-Kutta of order 4 was implemented to solve the system numerically.

8.2.3 Coordinated Velocity Control

The *Force* task in Figure 8.5 implements the velocity controller and the cooperative control as introduced in Section 5.2.

The coordinated control scheme includes multiple tasks interacting as a distributed scheme ensuring abort mechanisms if the multirotors are too close, lost intervehicle connection or similar cases. The reader is again referred to [Røli, 2015] for more information, the implementation from this thesis is the basis for the agent velocity controller.

The input states is the desired centroid body velocity, acceleration and heading as contained in the *IMC::DesiredLinearState* and *IMC::DesiredHeading*. Implemented as a periodic task with a fixed frequency the task implements the controller from Section 5.2.3 directly. For heading control and link-gain scheduling the approaches discussed in Section 5.2.1 and Section 5.2.2 is used.

To ensure a dynamically system, the global configuration message *IMC::CoordConfig* contains all the necessary parameters for the coordinated control setup, such that e.g. the desired link gain or desired formation can be changed while the system is operative.

8.3 Software in The Loop (SITL)

Further to test the algorithms a SITL setup where configured. In a SITL setup the software supposed to control the system is used on an external simulator as seen in Figure 8.6, where the controller sends the desired control input to the simulator and receives the consecutive states from the simulator. The benefits with this setup is that the control algorithms can be tested in the same manner as it would if it where controlling the real system.

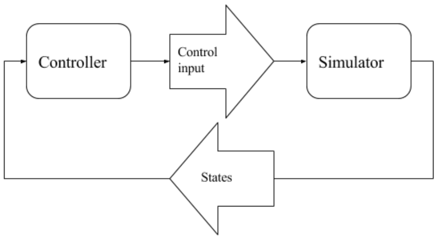


Figure 8.6: SITL principle

A more thorough and realistic setup would be to run the control algorithms on the actual hardware and send the desired input to the system to a simulated sensor platform that sends simulated measured states back, however, this setup known as *Hardware-In-the-Loop* (HIL) will not be addressed further.

JSBSim and ArduCopter simulator

The ArduPilot software framework as introduced in Section 3.3.2 contains a built-in simulators, but one is also able to use external simulators as seen in Figure 8.7. For the multirotor the built-in simulator was used together with the low-level attitude control-loops in ArduCopter. Then the desired force in the local $\{n\}$ frame was used as input, treating the multirotor as a controllable point in the local frame. The fixed-wing UAV was controlled using the landing system as discussed in [Nevstad, 2016] together with the ArduPlane controllers. Here JSBSim, an open-source Flight Dynamic Model (FDM) library modeling the 6-DOF airplane dynamics was used [JSBSim, 2015] using the X-8 dynamics as implemented in [Gryte, 2015].

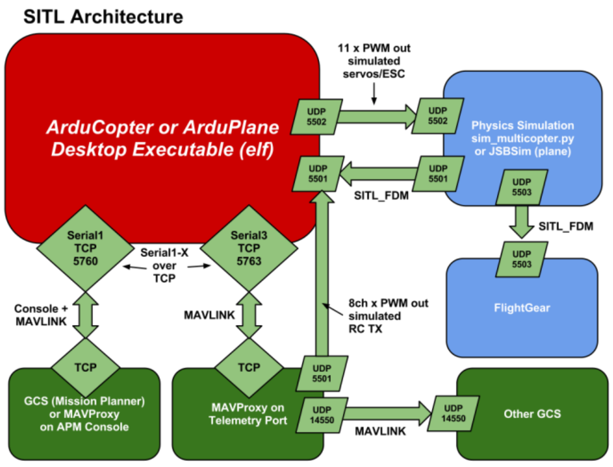

SITL Architecture
ArduCopter or ArduPlane Desktop Executable (elf)
11 x PWM out simulated servos/ESC
UDP 5502
UDP 5502
Physics Simulation sim_multicopter.py or JSBSim (plane)
UDP 5501
UDP 5501
SITL_FDM
UDP 5503
SITL_FDM
UDP 5503
FlightGear
Serial1 TCP 5760
Serial1-X over TCP
Serial3 TCP 5763
Console + MAVLINK
MAVLINK
8ch x PWM out simulated RC TX
TCP
TCP
UDP 5501
GCS (Mission Planner) or MAVProxy on APM Console
MAVProxy on Telemetry Port
UDP 14550
MAVLINK
UDP 14550
Other GCS

Figure 8.7: SITL architecture in ArduPilot. *Image courtesy of dev.ardupilot.com*

SITL setup

The total SITL setup is given Figure 8.8. In order to simulate the total recovery operation a multirotor and a fixed-wing UAV simulator is needed, this is achieved by running two instances of DUNE where each of them communicate with the ArduCopter and ArduPlane simulator respectively, here the MAVlink protocol is used by an internal ArduPilot interface task in DUNE. The fixed-wing instance of DUNE sends the current states to the multirotor instance of DUNE as the multirotor should control its location according to the fixed-wing. The simulation results are presented in Section 8.4.

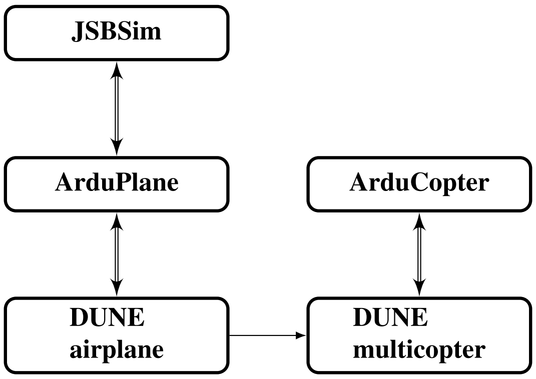


Figure 8.8: SITL setup

8.4 SITL results

Using the setup as introduced above, a set of SITL experiments were conducted. In the following, the different rates as summarized in Table 8.1 were used. It should be noted that the inner velocity controller has a higher rate than the outer path control and recovery coordinator loop to ensure that the outer loop can neglect the inner loop dynamics as discussed in Appendix D.2. The feedback states are the numerical solution from the external simulator. For the following results, the label *Estimated* will refer to the numerical feedback solution from the simulator. Further, these experiments does not include the suspended payload between the multirotor, hence, no collision dynamics or suspended load tension are included.

Feedback	50 Hz
Path control	10 Hz
Recovery coordinator	10 Hz
Velocity controller	30 Hz

Table 8.1: SITL - Controller and feedback frequencies

8.4.1 Payload transport

A SITL study on the transport control scheme was performed in a similar fashion as the simulation study in Section 6.3.1. Now, with the parameters as listed in Table 8.2 a simulation was performed, proving the usability of the implemented system.

Gain close, K_c	1.5
Gain far, K_f	0.6
$\tilde{z}_0$	3.0 m
$z_{0.9}$	2.5 m
Desired link distance, l	3.0 m
Velocity control gain, $\mathbf{K}_p$	$3\mathbf{I}_{3\times3}$

(a) Formation parameters

Reference surge, u_{ref}	0.7 m/s
Nomoto, timeconstant: T	0.9 s
Nomoto, rudder gain: K	1
Surge, mass: m	10
Surge, damping: d	1
Heading control damping: ζ_ψ	1
Heading control bandwidth: $\omega_{0,\psi}$	0.7 rad/s
Surge controller gains: $K_{p\tau}$,$K_{i\tau}$	5,1
Δ_y	10
Δ_z	10

(b) Reference simulator and LOS

Table 8.2: Payload transport (SITL): Parameters

Firstly, a set of waypoints were defined using the Neptus ground control station. Here four points forming a square with different height set-point were selected for comparability with the simulation results. The way-points, together with the multirotors positions in the local $\{n\}$ can be seen in Figure 8.10a[4]. As the theoretical simulation study proved the

[4]A video of the SITL path control experiment can be seen *SITL/Payload transport.mp4* in *Digital Appendix*.

reference simulator to be necessary it was enabled for this SITL experiment.

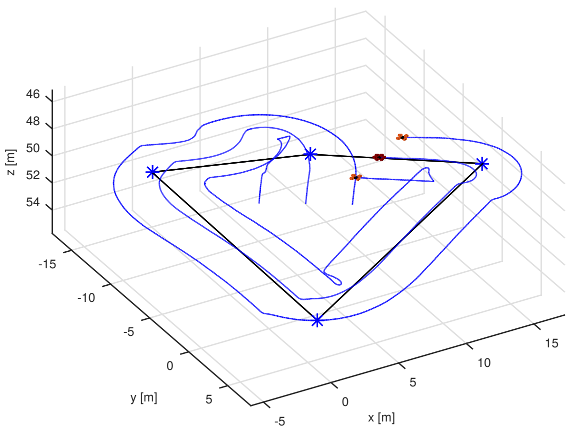
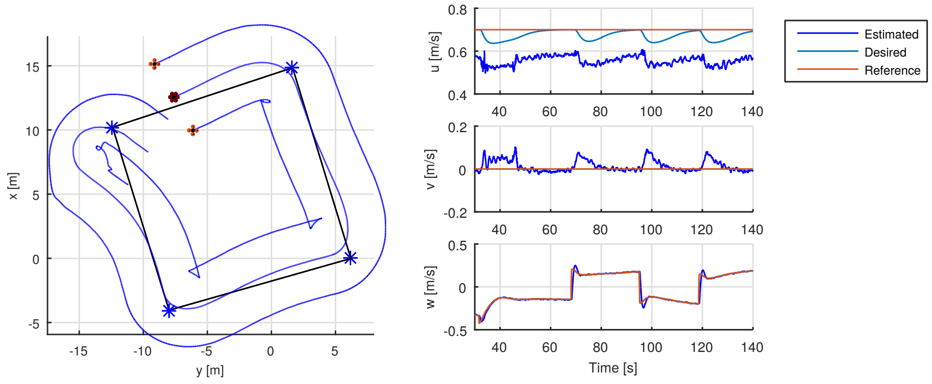


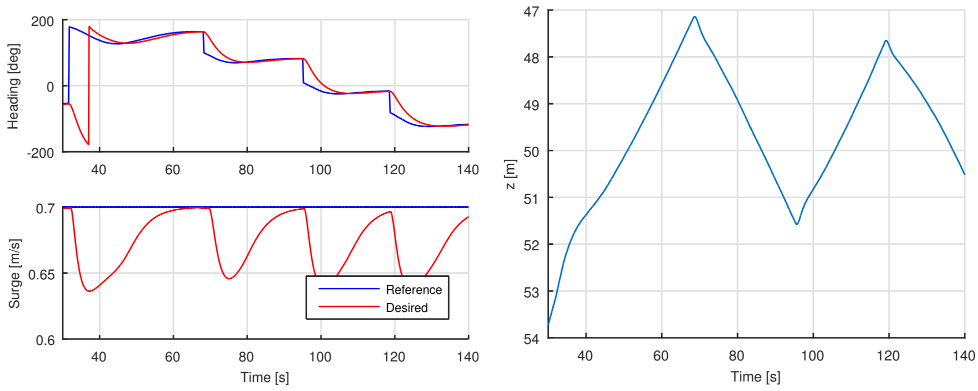
Figure 8.9: Payload transport (SITL): Multirotor positions in $\{n\}$, centroid illustrated as the red multirotor. The blue stars indicate the way-points connected by solid black lines

Observing the maneuver from above in Figure 8.10a reveals that controllers are able to render the centroid position towards the path given by two subsequent waypoints. Furthermore, sideway motions are minimized as stated in Figure 8.10b and the heave velocity is closely followed giving the height slopes as seen in Figure 8.11b. On the other hand, the velocity controller is not able to reach the desired surge velocity, in this setting it should be noted that neither the guidance or the velocity controller has integral action implemented. Lastly the reference simulator heading and surge response are revealed in the same fashion as in the theoretical simulation as Figure 8.11a states.



(a) North-East multirotor and centroid positions. **(b)** Centroid body $\{\bar{c}\}$ velocities, comparing reference and desired values.

Figure 8.10: Payload transport (SITL): Multirotor positions and centroid velocity.



(a) Comparing reference simulator inputs (reference) and output (desired); heading ψ and surge u. **(b)** Centroid Downwards height in $\{n\}$.

Figure 8.11: Payload transport (SITL): Reference simulator states and the height.

8.4.2 Recovery maneuver

Based on the theoretical simulation study setup in Section 7.6.2 a SITL recovery simulation was conducted. The parameters from Table 10.3 were applied using the along-track velocity ramp trajectory from Section 7.2.2.

x_f	30 m
v_f	3 m/s
a_{max}	1.5 m/s^2
Virtual-runway volume: $w \times h \times l$	$10 \times 15 \times 60$ m^3
$\mathbf{K}_{p,p}$	$1\mathbf{I}_{2\times 2}$
$\mathbf{K}_{d,p}$	$0.2\mathbf{I}_{2\times 2}$

Table 8.3: Recovery (SITL): Parameters

The virtual runway, as well as the desired waypoints towards and back from the runway were defined using the ground control station. Further, the fixed-wing UAV was instructed to do multiple approaches towards the runway, enabling multiple recovery maneuvers to be conducted. The reader is refered to [Nevstad, 2016] for details about the fixed-wing maneuver. An overview of the waypoints and the runway with the multirotors are shown in Figure 8.12[5], where the collision occured approximatly in the center of the runway illustrated as the point where the fixed-wing is directly above the centroid.

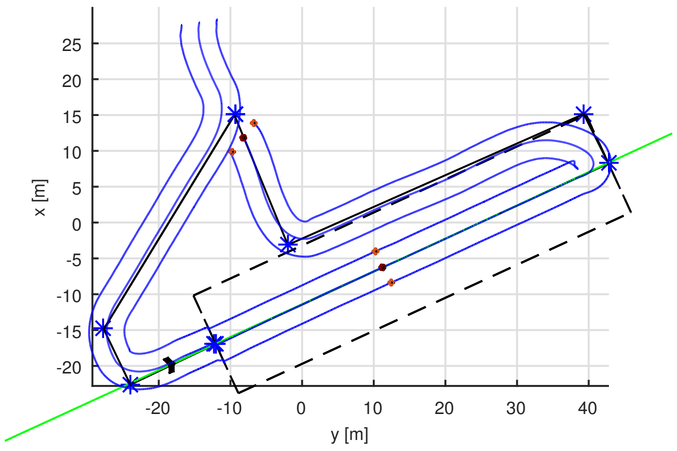
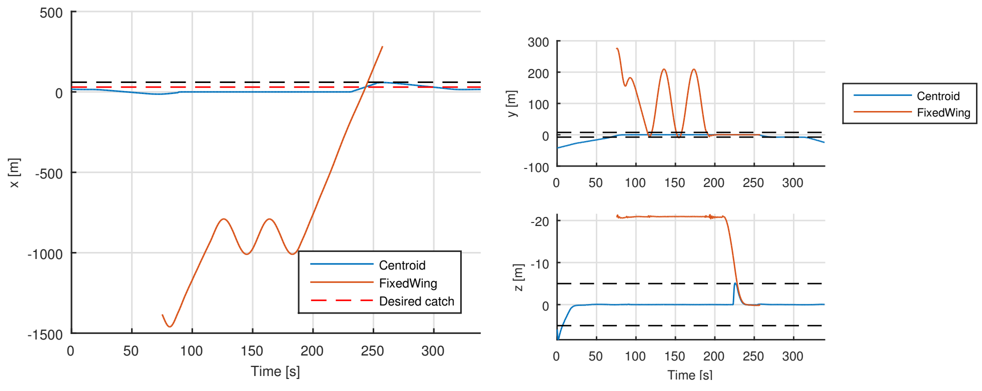


Figure 8.12: Recovery (SITL): North-East overview of the recovery, where the green and the blue line is the path of the fixed-wing and the multirotors respectively. The centroid path is also plotted as the center multirotor. The black dotted lines represents the boundaries of the virtual-runway and the blue stars is the desired waypoints, connected with the solid black line.

[5]A video of the SITL recovery maneuver can be seen in *SITL/Recovery.mp4* in *Digital Appendix*.

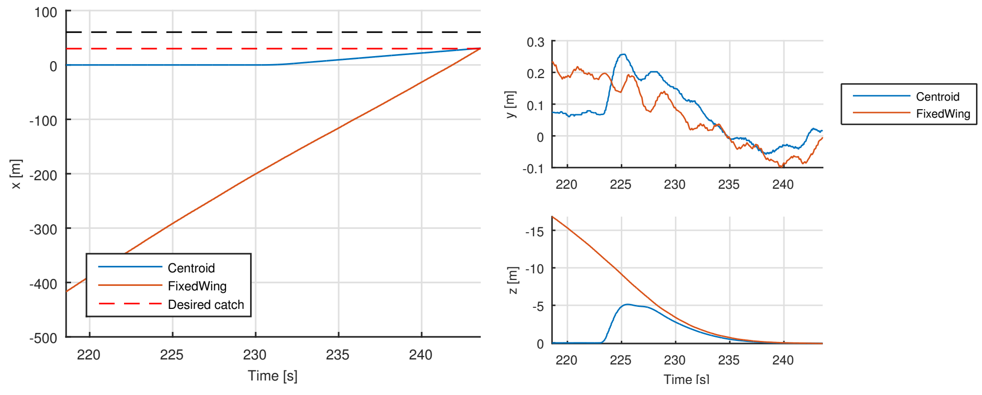
Unfortunately the SITL simulator given by ArduCopter does not provide wind disturbances, therefore, no wind was applied for the following results. Without wind the fixed-wing UAV was able to track the desired virtual runway direction in the North-East plane with very low error magnitude. Also, the multrotors were able to follow the fixed-wing, as illustrated in Figure 8.13 and Figure 8.14. For a more realistic experiment strong wind with turbulence should be applied, however, field experiments will also give such conditions.



(a) Along-track **(b)** Cross-track

Figure 8.13: Recovery (SITL): Along- and cross-track positions in the path-frame $\{p\}$, the boundaries of the virtual-runway is represented as the black-dotted line.

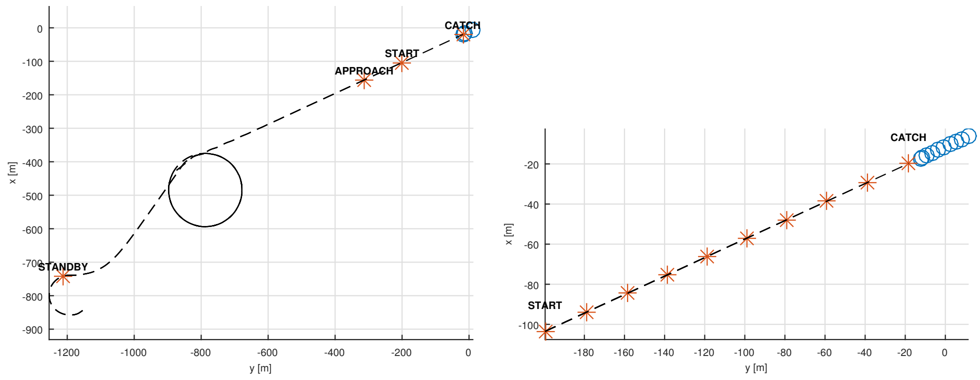
In Figure 8.13 a closer view is given on the along- and cross-track positions, revealing that the scheme are able to render the multirotors to the desired recovery point when the fixed-wing arrives and with the correct cross-track position.



(a) Along-track **(b)** Cross-track

Figure 8.14: Recovery (SITL): Along- and cross-track positions in the path-frame $\{p\}$ a short period before the impact.

The supervisor as discussed in Section 7.5 and Section 8.2.1 handles the different states during the maneuver, and in Figure 8.15 an overview of these are presented.



(a) All state changes during recovery

(b) Closer view on the states after the approach state

Figure 8.15: Recovery (SITL): The recovery state change timestamps during the operation symbolized as red stars along the fixed-wing path (black line), the blue circles is the position of the centroid at the specific timestamp.s

Lastly, the results from the velocity controller are presented in Figure 8.16, were the cross-track velocities are closely followed. However, as discussed in Section 8.4.1 the controller is not able to reach the desired surge velocity. Still, the recovery occurs within the virtual runway and close to the desired recovery point, proving the feasibility of this controller scheme.

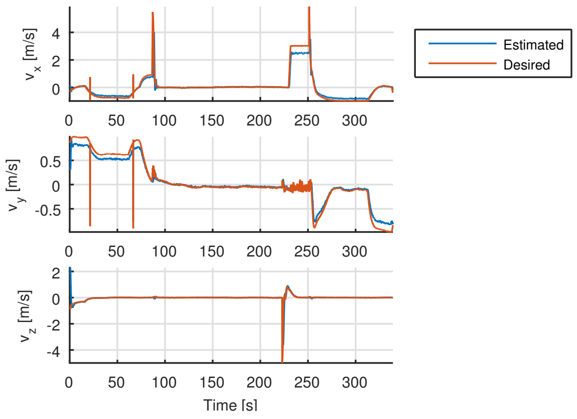


Figure 8.16: Recovery (SITL): Centroid velocities denoted in the virtual-runway frame $\{p\}$

Chapter 9

Operational aspects

This chapter will discuss some of the challenges regaring the practical usage of the recovery maneuver. This includes an example of how the overall maneuver can be conducted, as well as timing considerations in order to abort the operation. Lastly, the practical control and monitoring of the operation will be discussed.

9.1 Overall maneuver

The overall landing operation from a ship can be done as seen in Figure 9.1, note the safety radius r_s from the ship, which defines the minimum distance to the recovery maneuver. Given the desired fixed-wing landing path $\Pi(s)$ a recovery path $\Gamma(s)$ can be defined. The path contains crucial checkpoints and the legs between them are defined as in Table 9.1, at the different legs different controllers will be enabled.

$c_1 \rightarrow c_2$	Take-off	l_1
$c_2 \rightarrow c_3$	Move to the runway	l_2
$c_3 \rightarrow c_4$	Pre-recovery maneuver	l_4
$c_4 \rightarrow c_5$	Post-recovery maneuver	l_5
$c_5 \rightarrow c_6$	Transport to ship	l_6
$c_6 \rightarrow c_7$	Landing	l_7

Table 9.1: The different legs on the path $\Gamma(s)$

For the recovery maneuver discussion the relative along-track distances between the net and the fixed-wing r_a and r_{coll} are defined. The distance r_a from c_3 gives the desired position of the fixed-wing when the maneuver should start at checkpoint c_S as a function of the desired along-track impact position given by r_{coll} defining c_4. The transportation legs and the the legs noted as l_4 and l_5 are of most interest for this work and defines the recovery maneuver as discussed in Chapter 7.

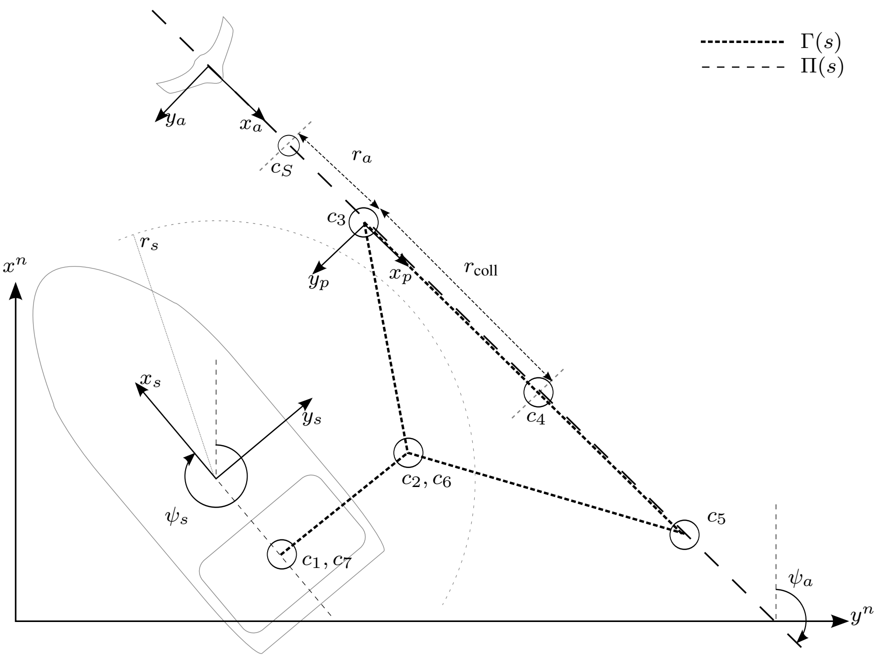


Figure 9.1: Overall plan in the North-East plane $\{n\}$ where $\{s\}$ presents body frame of the ship with heading ψ_s for illustration, and the approaching fixed-wing $\{a\}$ with heading ψ_a aligned with the path frame $\{p\}$

9.2 Abort

As the maximum operation duration is limited by the battery capacity of the multirotor the total operation time will be considered with respect to the battery time for the multirotors and a maximum waiting time at checkpoint c_4 will be given. Further if the multirotor is not able to reach the standby-checkpoint c_3 in time the fixed-wing should abort. Table 9.2 defines the different time stamps and durations to be consider in this analysis. Note that in the timing analysis the notation for minimum and maximum time duration is given as $t_{min} = \check{t}$ and $t_{max} = \hat{t}$ respectively.

[1]Time relative to the start of the overall plan

Time to start recovery mission[1]	t_0
Battery duration time (fixed-wing)	$t_{a,bat}$
Battery duration time (multirotors)	$t_{cop,bat}$
Time to do maneuver at leg l_i	t_i
Time to stand at checkpoint i	$t_{c,i}$

Table 9.2: The different time-parameters

Abort multirotor operation

During the transit from start to checkpoint c_3 the fixed-wing might abort landing due to multiple reasons and go for another try. As the total operation time is limited by the battery time of the multirotors (assuming $t_{cop,bat} << t_{a,bat}$), the maximum time the multirotor can wait at c_3 is $\hat{t}_{c,3} \geq t_{c,3}$, given that there exists an upper limit on the estimated time left of the operation.

$$t_{cop,bat} - \left(\sum_{i=1}^{3} t_i + \sum_{i=1}^{2} t_{c,i} + \sum_{i=4}^{7} \left(\hat{t}_i + \hat{t}_{c,i} \right) \right) = \hat{t}_{c,3} \tag{9.1}$$

Abort fixed-wing landing operation

The fixed-wing should abort if the multirotor is not capable of reaching checkpoint c_3 before the time to start the maneuver at t_0, therefore at each checkpoint c_j before c_3 the fixed-wing will abort if the current estimated maximum total time $\hat{t}_j$ to reach c_3 is longer than t_0, thus $t_j \geq t_0$, where t_j is given by

$$\hat{t}_j = \sum_{i=1}^{j} (t_i + t_{c,i}) + \left(\sum_{i=j+1}^{3} \hat{t}_i + \sum_{i=j+1}^{2} \hat{t}_{c,i} \right) \tag{9.2}$$

These timing abort mechanisms has not been investigated further, however, they should be considered when for practical usage of the recovery concept.

9.3 Recovery considerations

The proper size and mesh sizes of the net were considered. Based on experience from previous projects with net recovery of the Skywalker X-8 [Skulstad et al., 2015] a 5 meters wide and 3 meters tall was chosen. In order to recover the fixed-wing properly multiple hooks will be attached to the fixed-wing and a mesh size of 100x100 mm was chosen such that the hooks will attach to the net at impact. In Figure 9.2 a section of the net is illustrated. A light and stiff-rod will be attached to the top of the net in order to keep the net as stiff and stable during transit and impact, therefore a 5 meter long light tent-pole was chosen[1]. The reader should note that the choice of hooks and the feasibility of this recovery method is still to be addressed.

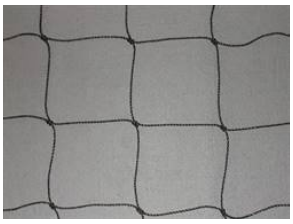


Figure 9.2: A section of the net

The net should be suspended to the multirotors using a rope between the multirotor and each corner of the net respectively. For safety reasons one should be able to quickly release the net, therefore a release mechanism was constructed[2]. A servo was mounted to a simple and robust release mechanism constructed to release sailplanes as illustrated in Figure 9.3. The ropes should not be too long in order to keep the magnitude of the oscillatory motions at a minimum. On the other hand, a too short rope reduces the relative operating area of the multirotor in the formation. This gives lower error margins with respect to the cooperative controller and the positioning system.

Furthermore a reasonable strong and light net is needed to catch the airplane, and a net made from the light and strong polyethylene (PE) was chosen. Based on experience from previous projects with net recovery of the Skywalker X-8 in [Skulstad et al., 2015] a 5 meters wide and 3 meters tall net seemed reasonable. Multiple hooks will be attached to the fixed-wing and with a mesh size of 100x100 mm the hooks should attach to the net at impact. To keep the net flat and stable during transit and impact, a light and stiff-rod will be attached to the top of the net. The reader should note that the choice of hooks and the feasibility of this recovery method are still to be addressed with physical experiments.

As the two octocopters should be able to lift the fixed-wing and a net, a mass budget was considered. The external payload gives the load the octocopters should be able to lift and consists of the airplane, the net and the rod as seen in Table 9.3a. Given the Maximum

[1]A light tent-pole as e.g. seen at `http://www.helsport.no/dac-featherlite-teltstang` fulfill these requirements.

[2]The reader should note that this mechanism is not constructed by the author, but as a part of the project summarized in [Klausen et al., Submitted 2016]

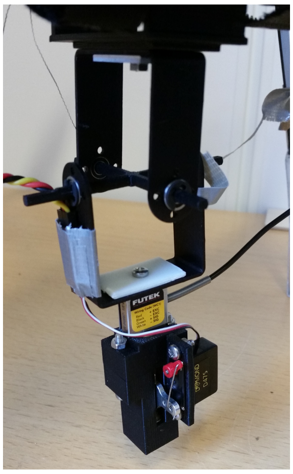


Figure 9.3: Release mechanism and weight cell attached to a gimbaled suspension

Take-Off Weight (MTOW) for each octocopter and the mass of all the external payloads as seen in the mass budget for two octocopters in Table 9.3b shows that there is a total of (MTOW $-$ Total weight) ≈ 2.6 kg left for both octocopters. This extra take-off weight should be filled by extra internal payload and the forces applied during the collision, and the choice of platforms seems feasible for the given operation, but is still to be addressed with physical experiments.

Component	Mass [kg]
Aircraft	4.0
Net	0.68
Rods	0.36
Total	5.03

(a) External payload to be lifted by the octocopters

	Mass [kg]
Frames	4.1×2
Battery[3]	1.765×2
External payload	≈ 5
Total	$\approx 10.3 \times 2$
MTOW	11×2
MTOW - Total	≈ 2.6

(b) Mass budget for two DJI S1000+

Table 9.3: Mass budget

[3]Mass of a 17 Ah 6 cells lithium polymer battery

9.4 Mission Control

The overall mission should be operated from the Neptus GCS as illustrated in Figure 9.4. Here the virtual runway is illustrated as the yellow box, defining the North-East plane boundaries. Furthermore, additional waypoints are added for the purpose of moving the net to the runway, and the transportation of the recovered fixed-wing UAV in the same manner as in Figure 9.1.

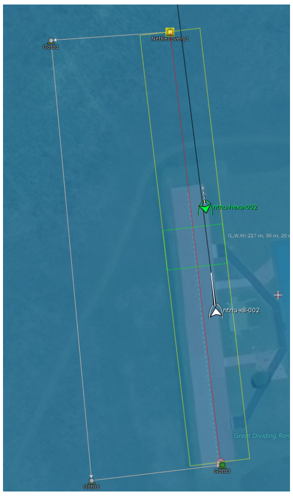


Figure 9.4: Screenshot from Neptus GCS during SITL, the fixed-wing UAV and the multirotor are illustrated as the green and white arrow respectively.

When the virtual runway is defined, the fixed-wing UAV will also receive the plan and generate a landing path according to the desired height and bearing of the runway [Nevstad, 2016]. Whilst waiting for the multirotor to reach the runway, the fixed-wing UAV will loiter around in a circle movement. On the operator signal, the fixed-wing will enter the landing trajectory and eventually trigger the multirotor recovery maneuver.

On the other hand, the take-off and landing phase has not been addressed, however, these phases should eventually be autonomous. For now, it is proposed to do these phases manually with two pilots. Moreover, two pilots should always be ready to take control of the multirotors, especially without any robust and safe abort maneuvers implemented.

Chapter 10

Experiments

The ultimate goal with this thesis is to demonstrate that the presented recovery concept can be conducted in the physical world. This chapter will demonstrate how the overall system as presented in Chapter 3 were used in order to do field experiments and proof the feasibility of the controller scheme implementation as presented in Chapter 8. Ultimately the DJI S1000+ octocopter will be used, however, for these initial experiments the 3DR hexacopter was used as the desired multirotor platform. Firstly, a brief presentation of the experiment location will be given, followed by the a selection of the experimental results. The different parts of the system were systematically tested to ensure that a successful demonstration of the overall recovery operation can eventually performed. As for the SITL experiments it has been focused on the two essential parts of the system; the transportation scheme using the reference simulator and the actual recovery maneuver at the virtual-runway. The rest of this chapter presents a selection of these field experiments.

10.1 Test facility - Agdenes airfield

The Agdenes airfield located about 90 km northwest of Trondheim is the primary test field for unmanned aerial systems (UAS) operations for the NTNU UAV-Lab. This airfield was also the test facility for all experiments conducted for this thesis. As the airfield has an actual airstrip as illustrated in Figure 10.1, the location was well qualified for performing the recovery maneuver. Therefore, in the recovery experiments the actual airstrip were treated as the *virtual-runway*.

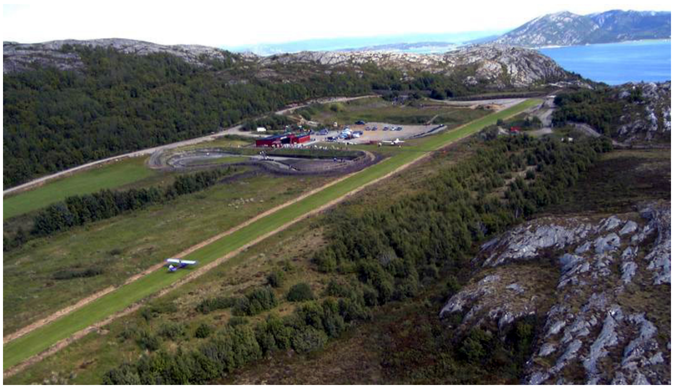


Figure 10.1: Agdenes airfield. *Image courtesy of norskeflyplasser.no*

10.2 Setup

In Table 10.1 the rates of the different system components are presented. Comparing with the SITL experiments Section 8.4 a lower feedback rate can be observed, as the feedback states now need to be estimated based on sensor measurements. Also, the feedback rates are now estimates of the measured rate into the controller systems. Concerning the parameters on the different algorithms it should be noted that the parameter presented for each part of the control architecture will be used for the all experiments unless something else is specified.

Regarding the navigation setup, the RTK solution is preferred due to its high precision. However, if a precise RTK solution is not available, the system implemented in [Sørbø, 2016] will use the Pixhawk solution instead, where the position is based on a single standalone GPS receiver. In the following experiments, the label *Estimated* will refer to the output from the navigation system. As the RTK solution presents both position and velocity solutions, these might change if a drop-out in RTK occurs. Orientation states and acceleration will always be given from the Pixhawk estimates based on the internal IMUs.

Feedback (Pixhawk)	20 Hz
Feedback (RTK)	10 Hz
Path control	10 Hz
Recovery coordinator	10 Hz
Velocity controller	30 Hz

Table 10.1: Experiments - Controller and feedback frequencies

10.3 Case 1: Gain scheduling

Initially experiments with the gain scheduling scheme as discussed in Section 5.2.2 was conducted. The purpose was to demonstrate that the scheduler were able to ensure a safe and smooth approach to the desired formation. Furthermore, the parameter was tuned to ensure that these requirements was met.

10.3.1 Setup

The experiment was conducted with only one multirotor airborne. Therefore, the slave multirotor was simulated on a external computer. The necessary data was transmitted using the wireless link between the simulated slave and multirotor as illustrated in Figure 10.2. The benefits with this setup is that only one pilot is required, furthermore, the risk factors are greatly reduced. On the other hand the simulation lack the possibility to simulate wind and other external factors. The viability of the gain scheduling was also shown in the following experiments in Section 10.5.

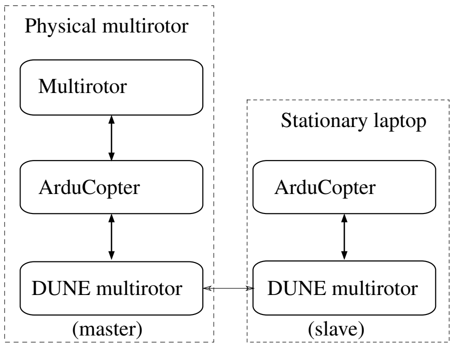


Figure 10.2: Simulated slave setup

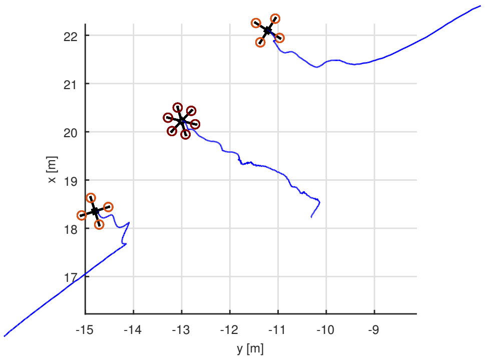
10.3.2 Results

First, both the slave and the multirotor were instructed to fly toward two separate locations 20-25 meters away from each other. Given the parameters in Table 10.2 this distance ensured that the initial gain was close to the far-distance gain K_f. Then a formation plan was conducted with zero reference speed ($u_{ref} = 0$ m/s) such that the multirotors flied towards each other and reached the desired link length l.

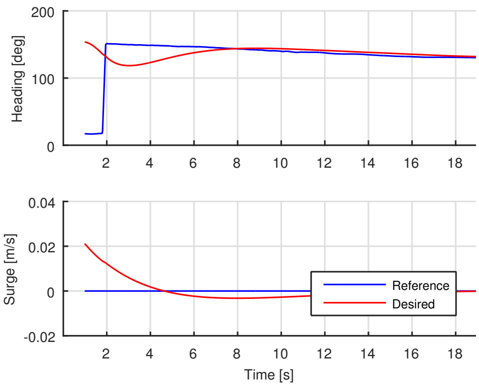
Gain close, K_c:	0.5
Gain far, K_c	0.05
$\tilde{z}_0$	7.0 m
$z_{0.9}$	4.5 m
Desired link length, l	5.0 m

Table 10.2: Case 1: Gain scheduling – Parameters

The resulting approach can be observed in Figure 10.3a, where the simulated slave multirotor are located west of the centroid[1]. Further as observed in Figure 10.3b a slight change in the desired heading can be observed due to the initial reference heading step, however, this did not affect the experiment greatly as the dynamics are slowly varying and of low magnitude.



(a) North-East multirotor positions, the dark red multirotor presents the centroid position-



(b) Comparing reference simulator inputs (reference) and output (desired); heading ψ and surge u.

Figure 10.3: Case 1: North-East positions, reference heading and surge

[1]A video showing the multirotors can be seen in *Experiments/Case 1 - Gain scheduling.mp4* in *Digital Appendix*. Here the telemetry logs are post-processed and illustrated using Matlab.

Then in Figure 10.4 the link distance $||\mathbf{z}||^2$ can be observed. The North-East plane link distance reveals a non-aggressive behavior towards the switching point at $\tilde{z}_0 = 7.0$ m/s where the sigmoid functions rises towards the in-formation link gain K_c resulting in a more aggressive final approach towards the desired link distance as desired.

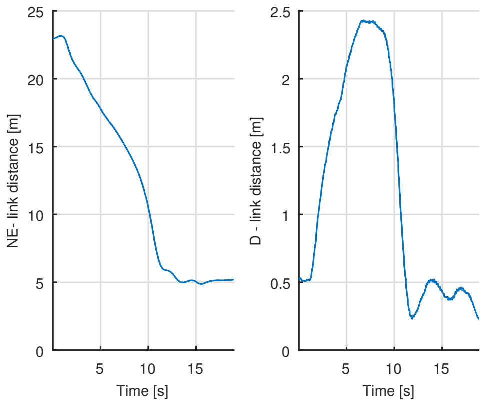


Figure 10.4: Case 1: Link distance expressed in the North-East plane $||\mathbf{z}_{1:2}||^2$ and the Down axis $|\mathbf{z}_3|$ respectively.

10.4 Case 2: Recovery

In the initial recovery experiments, a single multirotor without any suspended payload and a fixed-wing were used to test the recovery maneuver from Section 8.2.1. The purpose with this initial test was to see that the approaching fixed-wing triggered the multirotor maneuver. Two experimental results will be presented here, in the first one the fixed-wing were able to track the center of the virtual-runway, hence minor multirotor cross-track compensation was necessary. However, in the latter one, the fixed-wing had larger cross-track error, leading to major cross-track compensation from the multirotors.

10.4.1 Setup

For safety reasons the vehicles was instructed to be separated in height. Furthermore, the multirotor believed the fixed-wing to be at the same height as the virtual runway by adding an offset on the fixed-wing state message received on the multirotor, then the along-track and cross-track control scheme could be tested. The fixed-wing was instructed to fly in a predefined landing path with constant height when approaching the virtual runway. The cross-track controller gain matrices was set to $\mathbf{K}_{p,p} = \text{diag}(\begin{bmatrix}0.5 & 1\end{bmatrix})$, and $\mathbf{K}_{p,d} = 0.1\mathbf{K}_{p,p}$. For the position hold, the gain matrices were set to $\mathbf{K}_{p,ph} = \text{diag}(\begin{bmatrix}0.5 & 0.5 & 1\end{bmatrix})$ and $\mathbf{K}_{d,ph} = 0.1\mathbf{K}_{p,ph}$.

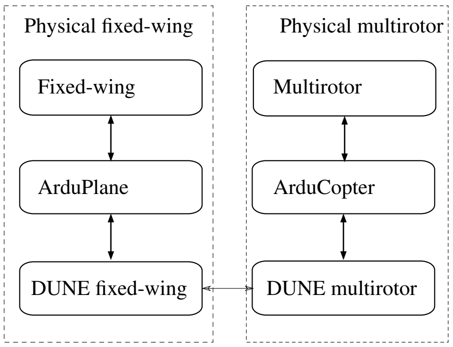


Figure 10.5: Case 2: Recovery setup

x_f	40 m
v_f	5 m/s
a_{max}	1.5 m/s^2
Virtual-runway volume: $w \times h \times l$	$10 \times 5 \times 60$ m^3
Height separation: z_L	40 m

Table 10.3: Case 2: Recovery – Parameters

10.4.2 Case 2a: Minor compensation

The following experiment required minor cross-track correction during the approach phase as seen in the overview in Figure 10.6[2].

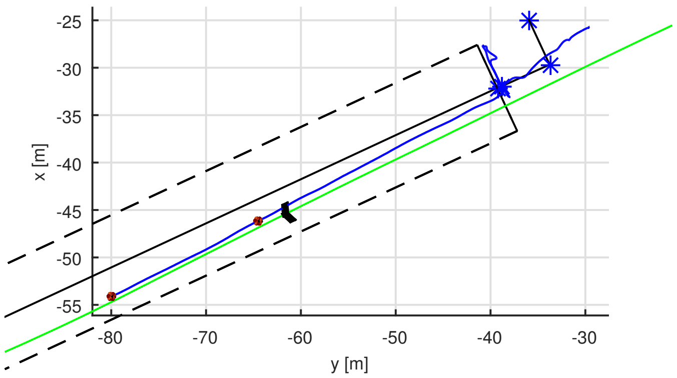
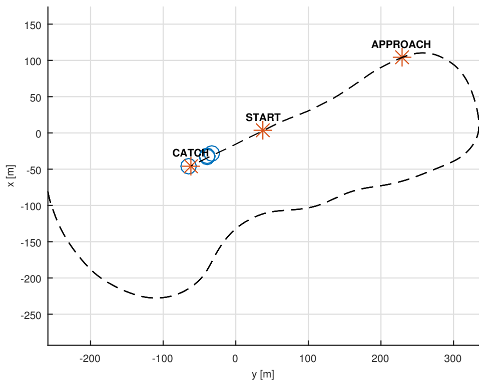


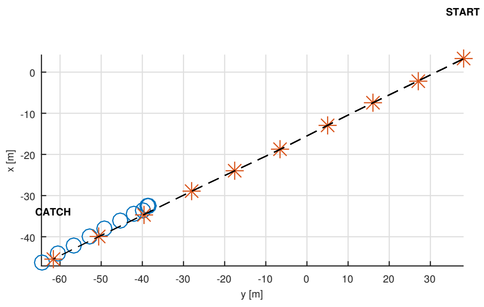
Figure 10.6: Case 2a: North-East overview over the recovery, where the green and the blue line is the path of the fixed-wing and the multirotor respectively. The black dotted lines represents the boundaries of the virtual-runway. The blue stars is the desired waypoints, connected with the solid black line.

During this experiment the fixed-wing followed a larger landing trajectory as seen in Figure 10.7a. However, as expected, the boundaries for the approach phase for the multirotors ensured that cross-track control was engaged after the fixed-wing had turned towards the runway.

[2]A video of the experimental recovery maneuver can be seen in *Experiments/Case 2a: - Recovery.mp4* in *Digital Appendix*generated by post-processing in Matlab.



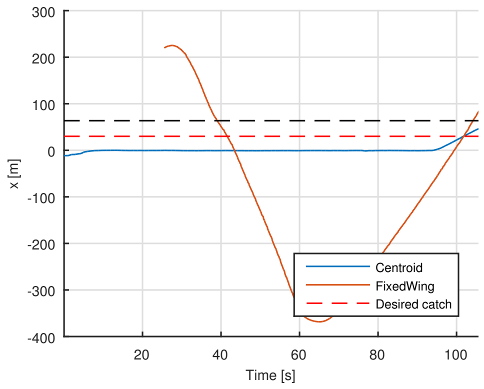
(a) All state changes during recovery



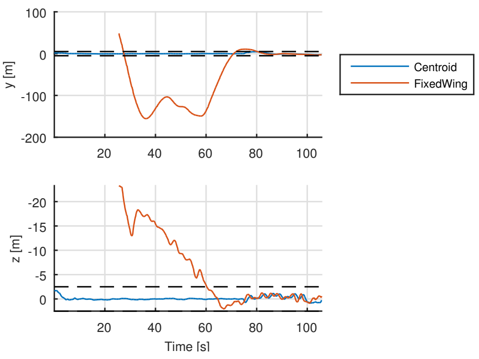
(b) Closer view on the states after the approach state

Figure 10.7: Case 2a: The recovery state change timestamps during the operation symbolized as red stars along the fixed-wing path (black line), the blue circles is the position of the centroid at the specific timestamps

The approach phase can be observed more closely in Figure 10.8 and Figure 10.9. Due to the estimation of the ETA for the fixed-wing the multirotor are able to reach the desired along-track catch point when the fixed-wing passes the same location within some reasonable boundaries. More importantly, the position of the multirotor in the cross-track plane are close to the fixed-wing at the rendezvous. For this run, the fixed-wing would have hit the net.

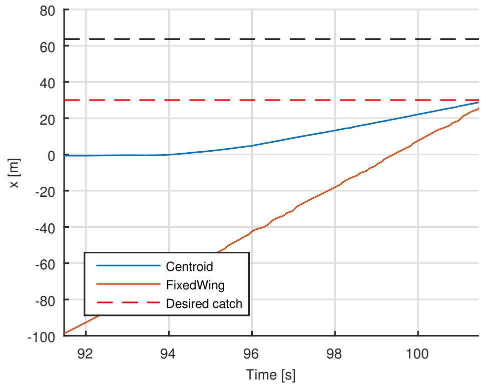


(a) Along-track

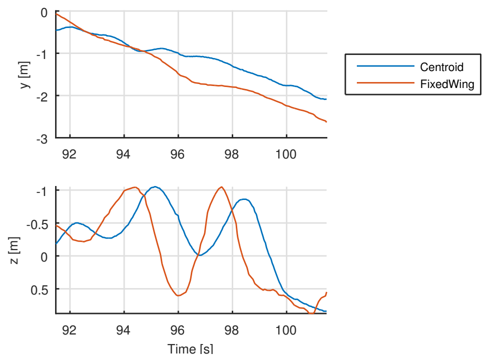


(b) Cross-track

Figure 10.8: Case 2a: Along- and cross-track positions in the path-frame $\{p\}$, the boundaries of the virtual-runway is represented as the black-dotted line



(a) Along-track



(b) Cross-track

Figure 10.9: Case 2a: Along- and cross-track positions in the path-frame $\{p\}$ a short period before the impact

Lastly, the performance of the velocity loop was investigated as seen in Figure 10.10. The controller was able to follow the reference, however, some time delay can be observed between the desired state and the state. This is as expected, as none of the controllers incorporate acceleration feedback during this maneuver.

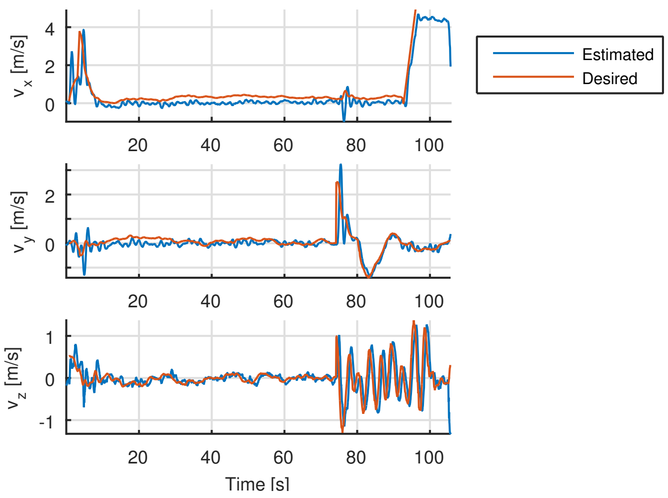


Figure 10.10: Case 2a: Multirotor velocities denoted in the virtual-runway frame $\{p\}$.

10.4.3 Case 2b: Major compensation

This case illustrate on of the flights where the fixed-wing struggled to minimize the cross-track error, hence, more multirotor cross-track movement was observed as seen in Figure 10.11[3]

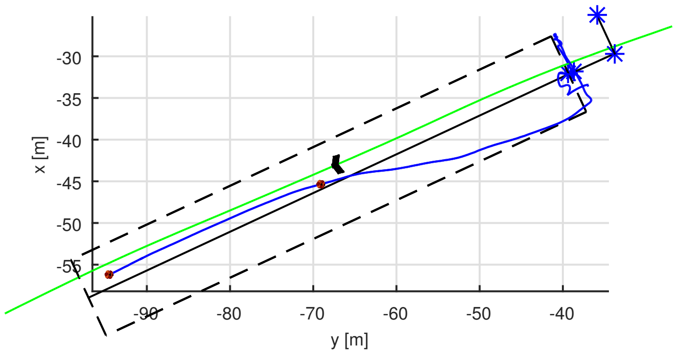
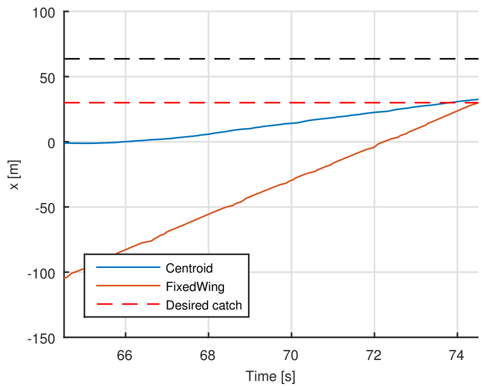


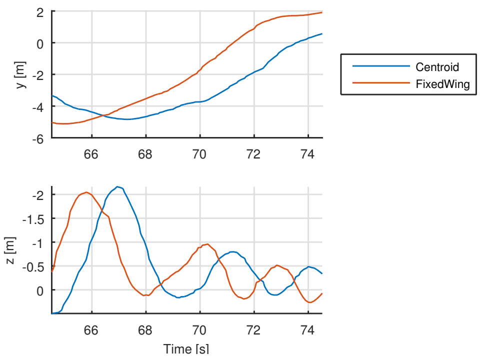
Figure 10.11: Case 2b: North-East overview over the recovery, where the green and the blue line is the path of the fixed-wing and the multirotor respectively. The black dotted lines represents the boundaries of the virtual-runway. The blue stars is the desired waypoints, connected with the solid black line.

Still, the multirotor was able to reach the rendezvous as illustrated in Figure 10.12a. On the other hand major time lag for the cross-track positions are seen in Figure 10.12b. This is not surprising, as the controller were not tuned aggressively as the controller might experience large initial errors when engaged at approach.

[3]A video of the experimental recovery maneuver can be seen in *Experiments/Case 2b: - Recovery.mp4* in *Digital Appendix*generated by post-processing in Matlab.



(a) Along-track



(b) Cross-track

Figure 10.12: Case 2b: Along- and cross-track positions in the path-frame $\{p\}$ a short period before the impact

To support the arguments above one can inspect the tracking of the desired velocity trajectory in Figure 10.13 is reasonably good. Still, some lag can be observed, however, the magnitude of this time difference can not explain the cross-track position errors.

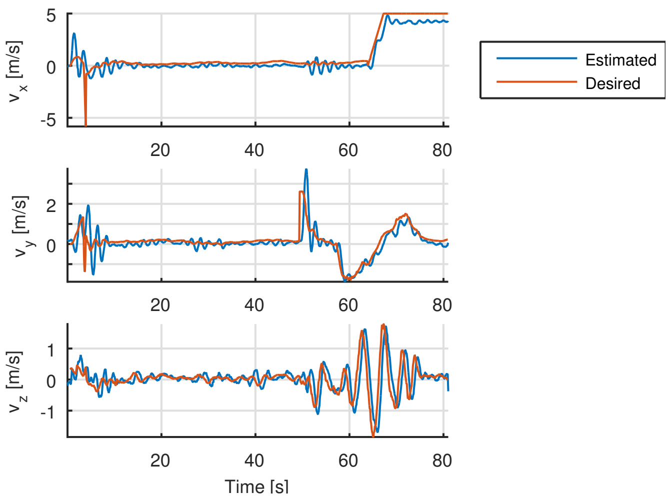


Figure 10.13: Case 2b: Multirotor velocities denoted in the virtual-runway frame $\{p\}$.

10.5 Case 3: Payload transport

In the initial transport experiments, one sought to test the path controller from Section 8.2.2 to verify a smooth movement along a predefined path controlling two multirotors. For the following experiments the path was formed as a square in order to challenge the guidance scheme through the whole maneuver and the possibility to do multiple laps.

10.5.1 Setup

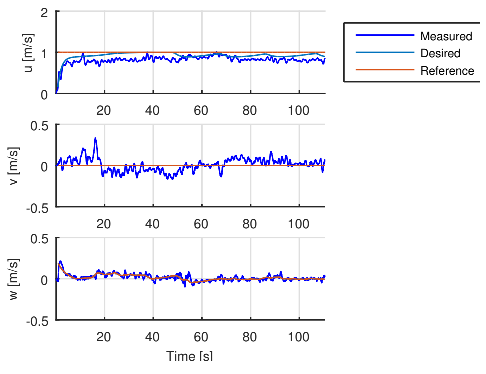
For operational and safety reasons the initial experiments was conducted with one multirotor simulated on a stationary computer and the other one being a physical multirotor platform as described in Figure 10.2.

10.5.2 Case 3a: Compare reference surge

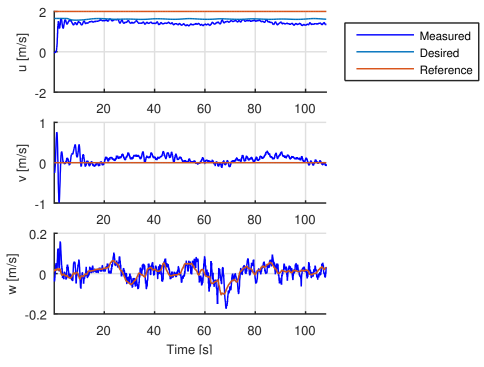
For this section, experiments with cooperative multirotor control was conducted adding a second simulated multirotor slave. The experimental setup as presented in Figure 10.2 was utilized in order to achieve such a scenario. Furthermore, as only one multirotor was airborne all results were obtained utilizing the navigation solution from the Pixhawk.

Two experiments altering the reference surge u_{ref} by 1 m/s and 2 m/s were conducted. Here the lookahead distances in the LOS controller were set to $\Delta_y = \Delta_z = 5$.

In Figure 10.14 one can observe that the controllers were able to reach the desired velocity, however, the reference simulator do not reach the reference surge when setting $u_{ref} =$ 2 m/s.



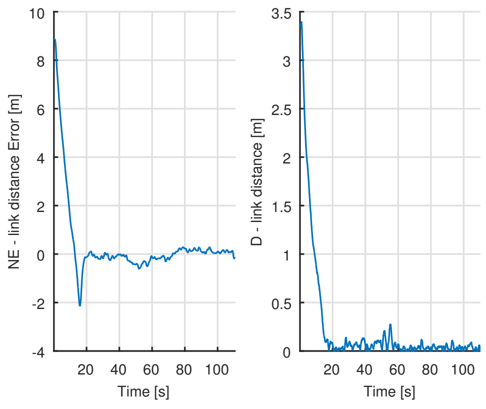
(a) $u_{ref} = 1$ m/s



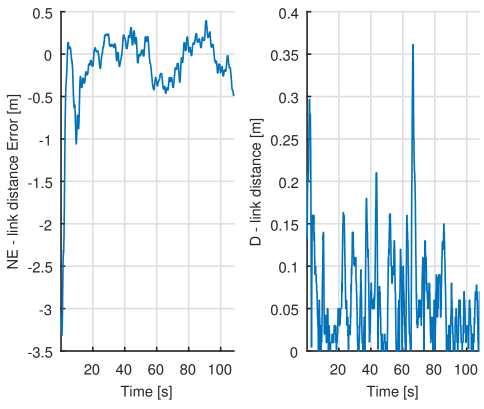
(b) $u_{ref} = 2$ m/s

Figure 10.14: Case 3a: Centroid body $\{\bar{c}\}$ velocities, comparing reference and desired values.

Further it seemed that the desired velocity did not have a great impact on the link distance as illustrated in Figure 10.15.



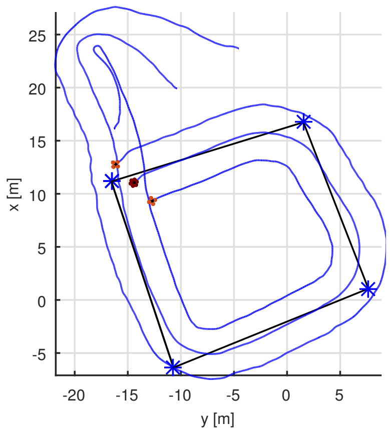
(a) $u_{ref} = 1$ m/s



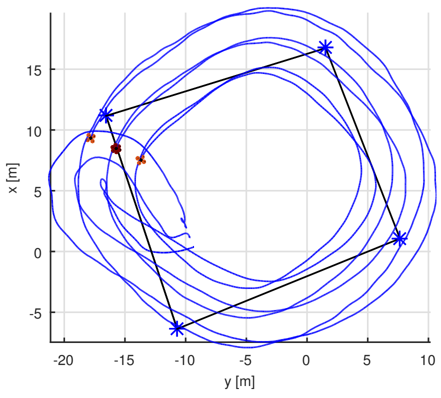
(b) $u_{ref} = 2$ m/s

Figure 10.15: Case 3a: Link distance error $\tilde{z}$, expressed in the North-East plane and and Down axis respectively

On the other hand, with greater speed, the centroid was not able to reach the path between the waypoints as seen in Figure 10.16[4]. The latter one pinpoint the fact that the desired waypoint switching radius is dependent on the current surge reference speed as well as the lookahead distance Δ_y. This coupling must be handled in order to achieve the desired performance for any desired path and speed.



(a) $u_{ref} = 1$ m/s



(b) $u_{ref} = 2$ m/s

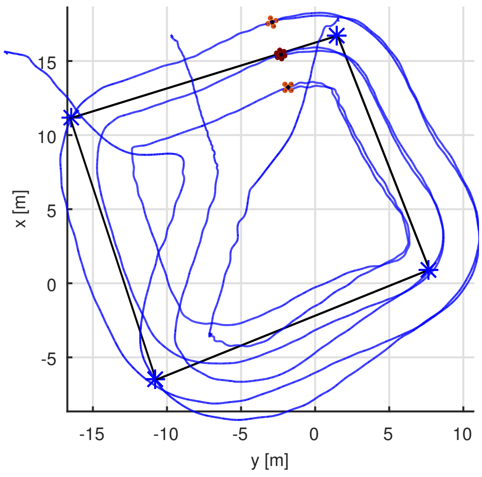
Figure 10.16: Case 3a: North-East multirotors and centroid positions.

[4]The simulated slave is the outermost multirotor

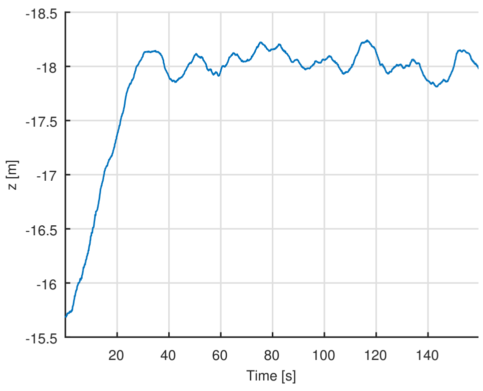
10.5.3 Case 3b: Cooperative

The ultimate transport experiment was conducted using two airborne multirotors. The takeoff was performed manually by the two pilots using a RC-controller. Then the two multirotors were instructed to move to two separate points located approximately 20 meters apart. Here, the RTK navigation solution was utilized to obtain centimeter accuracy. The LOS controller lookahead distances was set to $\Delta_y = \Delta_z = 5$. Furthermore, the centroid was instructed to move with a surge velocity $u_{ref} = 1$ m/s. Lastly the desired link distance was set to $l = 5$ m.

The resulting positions of the multirotors through the run are illustrated in Figure 10.17[5]. It can be observed that the multirotors move into formation slowly while gaining forward speed towards the first waypoint located southwest. However, on the last leg moving towards the waypoint located farthest north the right multirotor starts to oscillate towards the other multirotors, hence, the pilots decides to abort the mission by regaining manual control.



(a) North-East multirotor positions



(b) Down position in $\{n\}$

Figure 10.17: Case 3b: North-East positions and height

[5]A video showing the multirotors can be seen in *Experiments/Case 3b: Cooperative.mp4* in *Digital Appendix*. Here the telemetry logs was post-processed and illustrated using MatLab.

Furthermore, by observing the link distance in Figure 10.18 the cooperative controller shows satisfactory performance for this experiment.

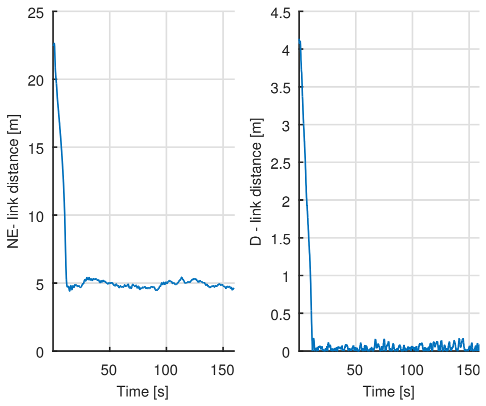
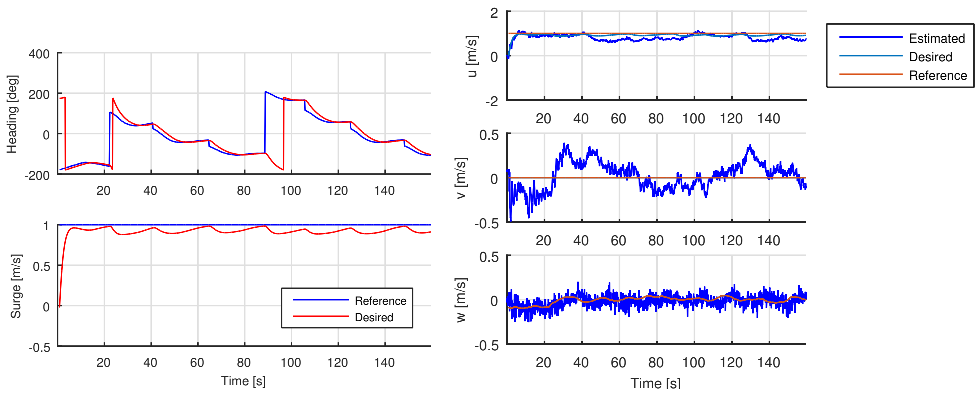


Figure 10.18: Case 3b: Link distance expressed in the North-East plane $||\mathbf{z}_{1:2}||^2$ and the Down axis $|\mathbf{z}_3|$ respectively

Also by inspecting the centroid velocities and the references in Figure 10.19 oscillatory movements can be observed along all axes at the end of the maneuver. However, the centroid seems to follow the desired velocity in the beginning of the maneuver.



(a) Comparing reference simulator inputs (reference) and output (desired); heading ψ and surge u.

(b) Centroid body $\{\bar{c}\}$ velocities, comparing reference and desired values.

Figure 10.19: Case 3b: Centroid velocity and reference simulator states.

Even so, it should be noted that there was windless conditions during this experiment, moreover, the controllers activated during this maneuver do not utilize integral action. Therefore, large deviations from the setpoint should be expected during more windy conditions.

10.6 Discussion

The field experiments proved that the proposed concept is able to both transport and recover the fixed-wing. However, the actual transportation and recovery of the fixed-wing UAV is yet to be verified. Still, the concept lack robustness. Among others need the cooperative controller to be revised in order to sustain the formation in strong wind, furthermore the response with a suspended payload has not been observed. It should also be noted, that integral action should be considered for the along-track velocity control in order to be able to fully reach and hold the along-track velocity. This will be more important when the is attached, as the net will introduce more drag to the system.

Moreover, the supervisor introduce potentially large steps in the references to the controllers. Taking the cross-track controller as an example one will observe aggressive behavior if the multirotors enters the approach state with large fixed-wing cross-track error. This was observed during the experiments and was temporarily solved by lowering the controller gains. However, this introduce a slow multirotor tracking during the approach state. Hence, the gain should adapt according to the magnitude of the error; large position errors should reduce the gain. For the cooperative link gain this was solved using a logistic function as explained in Section 5.2.2, but other methods may also be considered. On the other hand, the solution can be handled by filtering the reference signals during the state transitions to avoid this behavior.

The experiments also revealed that the system are not trivial to tune. Firstly, as addressed above, the parameters in the LOS steering-law and the reference simulator are coupled. Hence, relations between some of the parameters should be exposed to simplify the tuning. Moreover, the outer position and cross-track controller which dispatches a desired velocity into the velocity controller assembles a cascaded control system. These control loops can be combined to investigate pole placement of the closed loop system utilizing the simplified dynamics for control synthesis in Section 4.4. Still, the current approach is to keep the bandwidth of the inner loop higher than the outer loop. Then, the outer loop can neglect the inner loop dynamics and assume that the current velocity equals the desired. This approach is discussed in detail in Appendix D.

Chapter 11

Conclusion and Closing Discussions

This chapter will conclude and discuss the main results and see if they were as expected, based on the discussion some suggestions for future work will be presented.

11.1 Dynamic models

The multi-body dynamics presented in Chapter 4 gives the ability to investigate the different challenges emerging when suspending a load to a group of multirotors. Furthermore, the consequences of the impact when the fixed-wing are recovered into the net is of great interest for the feasibility of the concept. The analysis provided among others crucial data about the tension in Section 4.5 demonstrating that the multirotors must be able to withstand a vast amount of forces for some period during the impact.

However, the most important property of the collision dynamics is that a oscillatory movement is engaged as seen in Figure 4.6. These movements most be suppressed before the transportation of the fixed-wing can be continued. However, there exists multiple methods to withstand these fluctuations as discussed in e.g. [Klausen et al., 2015] and [Bisgaard et al., 2010] which has not been investigated further in this work.

Further, a partially linear model for control synthesis purposes as given in Section 4.4 are utilized. This model simplifies the net to be an external force disturbing the dynamics of the multirotor. As discussed in [Klausen et al., 2015] the suspended load and the multirotor can be modeled as a coupled non-linear dynamical system. However, for this work, the linear analysis has proven to be sufficient for proving the viability of the concept.

11.2 Transport and recovery concept

The payload transport scheme as discussed in Chapter 6 gives a more feasible path controller for the purpose of transporting a net. Even though more time and a longer path is required, the benefits is less side-way motion in the net. Furthermore, the lower-level formation controller is instructed with a less aggressive heading reference trajectory, ensuring that the desired link distance between the multirotors is maintained for all formation rotations. Still, the coupling between the tuning of the LOS guidance law should be investigated further.

Furthermore, the feasibility of the recovery scheme concept discussed in Chapter 7 has been shown in both simulation and practical experiments. However, the actual recovery has not been verified yet. On the other hand, common sense and the collision discussion has shown that reducing the relative velocity between the net and the fixed-wing will reduce the impact forces. Furthermore, the multirotors agility should be utilized to control the position of the net according to the fixed-wing UAV. Still, the transport scheme in Chapter 6 sought to ensure a smooth and non-aggressive movement of the suspended net. This approach has not been studied for the recovery maneuver. On the other hand, the fixed-wing act as the reference model in this case, and the fixed-wing controller scheme [Nevstad, 2016] instruct the fixed-wing to follow a slowly declining glide slope towards the virtual runway, ensuring a non-aggressive behavior. This argument has been illustrated through simulation and experiments.

Summing up, this approach is far from optimal, but for the purpose of investigating the concept it has proven to be sufficient.

11.3 Low-level controllers

The low-level controller as discussed Chapter 5 is responsible for the velocity and formation of the controllers has shown overall good performance through the analytic and simulation results. However, these environment did not incorporate external disturbances such as wind, therefore practical experiments revealed that the robustness of the controllers was not sufficient enough. Still, the recovery maneuver cross-track controller in Section 7.3 incorporate integral action, and the transport LOS guidance law in Section 6.1 can also be modified to obtain such effects as discussed in among others [Fossen, 2011]. Even so, the formation controller is still subjected to the same challenges during windy conditions. A MRAC velocity controller was conceptually analyzed and shows promising results as discussed in detail in both Section 6.3.2 and Section 7.6.3, it should be noted that the results are not optimized regarding the tuning of the controller. As the suspended load and multirotor dynamics are highly coupled and change during the impact, applying an adaptive controller seems to be an desirable optional solution that should be investigated further.

11.4 Further work

The analysis of the system dynamics, specifically the impact dynamics should be validated through experimental setups. This can performed by instructing the multirotors to hold the airborne net, then by instructing a fixed-wing UAV to fly into the net the suspended load sensors can measure the direction and magnitude of the force applied between the net and the multirotors. Then the model properties might be utilized for more advanced pre-/post-collision maneuvers. The fluctuation of the net during the maneuvers and specially after the impact should be reduced, here the methods as proposed by among others [Klausen et al., 2015] should be investigated.

Furthermore, the coupling between the LOS guidance law and the reference simulator should be investigated further, ultimately proposing a tuning scheme independent of the desired path geometry and surge. Utilizing the coupling between the turning radius of the multirotor formation and the desired waypoint acceptance radius might introduce parameter dependencies and simplify the tuning of the setup. On the other hand, a closed-loop surge synchronization with the fixed-wing should be investigated for the recovery maneuver, here it should be utilized that the fixed-wing follows a predefined path as discussed in [Nevstad, 2016]. Lastly, the MRAC controller can be investigated further for estimating and compensating for the load force, here it would be interesting to incorporate the impact dynamics for predicting the load during the impact.

The timing analysis in Section 9.2 has not been addressed in simulations, however, the discussions shows that optimization techniques might be applicable for the operation as the total operation time needs to be minimized since the flying time for the multirotor is finite. Also different abort mechanisms concerning both time and deviations from the desired recovery path should be stressed more thorough. Moreover, the robustness and repeatability of the operation of the overall operation is not pursued in this work, and is crucial for any practical usage of the recovery method.

Appendix A

Additional Dynamical Modeling Theory

A.1 Modeling multi-body constrained dynamics

The concept of multi-body constrained dynamics raises different modeling challenges, and the choice of method is crucial depending on the purpose of the mathematical modeling. When modeling with the purpose of simulation studies, a more detailed and realistic model is required compared to a model for control synthesis purposes, where the most important dynamics will be considered. Here, the first one will be addressed. Different techniques exists, but the major differences depends on whether the forces applied on each body is found explicitly using independent generalized forces or if a method utilizing the dependent generalized coordinates where the constraint forces can be found explicitly. The latter one will be emphasized in this analysis. A method proposed by Udwadia and Kalaba [Udwadia & Kalaba, 1992] gives the possibility to first find the unconstrained dynamics of the different bodies, and then find the constrained dynamics, as well as calculating the constrained forces directly.

A.1.1 Udwadia-Kalaba

The method is presented in [Udwadia & Kalaba, 1992], and is applied on suspended load systems by among others [Klausen et al., 2014] and [Bisgaard, 2008]. The following section will summarize the derivation presented in these papers.

The Udwadia-Kalaba (UK) approach considers the dynamics of the unconstrained and constrained system. Therefore the generalized coordinates $\mathbf{q} = \left[q_1, \cdots q_n\right]^T \in \mathbb{R}^n$ and $\mathbf{q}_u = \left[q_1, \cdots q_n\right]^T \in \mathbb{R}^n$ for the constrained and unconstrained system respectively are defined.

The unconstrained dynamical Newtonian system is given as

$$\mathbf{M}\ddot{\mathbf{q}}_u = \mathbf{Q} \tag{A.1}$$

where the mass matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ is symmetric and positive definite and $\mathbf{Q} \in \mathbb{R}^n$ is the generalized forces applied on the system.

Then let the system be subjected to m linearly independent constraints in the form

$$\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}, t) \tag{A.2}$$

where the constraint matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{b} \in \mathbb{R}^m$ are known.

By applying the constraints in Equation (A.2) the explicit constrained system is affected by the generalized forces of the constraints $\mathbf{Q}_c \in \mathbb{R}^m$ such that the constrained system can be written on the form

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{Q} + \mathbf{Q}_c \tag{A.3}$$

The constrained generalized forces are found explicitly by applying Gauss's principle of least Constraints. The principle states that the constrained acceleration $\ddot{\mathbf{q}}(t)$ follows the closest unconstrained acceleration $\ddot{\mathbf{q}}_u(t)$ which is fulfilled by Equation (A.2).

The Gaussian function $\mathfrak{G}$ is defined as

$$\mathfrak{G} = \left[\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_u\right]^T \mathbf{M} \left[\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_u\right] \tag{A.4}$$

and the solution is found by minimizing $\mathfrak{G}$ subject to Equation (A.2). The solution is found using the Moore-Penrose pseudoinverse denoted as $()^\dagger$ which gives the constrained acceleration $\ddot{\mathbf{q}}(t)$

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_u + \mathbf{M}^{-\frac{1}{2}} \left(\mathbf{A}\mathbf{M}^{-\frac{1}{2}}\right)^\dagger (\mathbf{b} - \mathbf{A}\ddot{\mathbf{q}}_u) \tag{A.5}$$

by combining the results the constrained general forces $\mathbf{Q}_c \in \mathbb{R}^m$ is found explicitly

$$\mathbf{Q}_c = \mathbf{M}^{\frac{1}{2}} \left(\mathbf{A}\mathbf{M}^{-\frac{1}{2}}\right)^\dagger (\mathbf{b} - \mathbf{A}\ddot{\mathbf{q}}_u) \tag{A.6}$$

A.2 Collision dynamics

For general collision models it is common to assume that the forces between the contact points are large compared to the other external forces and that the orientation and positions do not change in the actual collision as discussed in e.g. [Chatterjee & Ruina, 1998]. From this it follows that the impulse momentum of the total system is conserved, such that the linear and angular momentum of the system before and after the collision do not change as seen in Equation (A.7) and Equation (A.8) respectively.

$$\mathbf{P}_+ = \mathbf{P}_- \tag{A.7}$$

$$\mathbf{L}_+ = \mathbf{L}_- \tag{A.8}$$

here the subscript $()_+$ indicates momentum directly after the impact and $()_-$ immediately before.

Based on the assumption of conserved impulse momentum the average collision force and moment can be calculated. By following the derivation in Section 2.2 one can use Equations (2.13)–(2.15) and Equations (2.14)–(2.16) to find the average force and moment during the collision respectively. Such that for each body i in the collision the following equations can be derived.

$$\bar{\mathbf{f}}_i = \frac{\mathbf{P}_{i,+} - \mathbf{P}_{i,-}}{\Delta t} = \frac{(\sum_j m_j)\mathbf{v}_{i,+} - m_i \mathbf{v}_{i,-}}{\Delta t} \tag{A.9}$$

$$\bar{\mathbf{m}}_i = \frac{\mathbf{L}_{i,+} - \mathbf{L}_{i,-}}{\Delta t} = \frac{(\sum_j \mathbf{I}_j)\boldsymbol{\omega}_{i,+} - \mathbf{I}_i \boldsymbol{\omega}_{i,-}}{\Delta t} \tag{A.10}$$

where the total linear momentum and angular momentum of the system is conserved, that is by defining $\mathbf{P} = \sum_j \mathbf{P}_j$ and $\mathbf{L} = \sum_i \mathbf{L}_j$ by summing over the momentum of the bodies the following holds

$$\sum_j \mathbf{P}_{j,+} = \sum_j \mathbf{P}_{j,-} \tag{A.11}$$

$$\sum_j \mathbf{L}_{j,+} = \sum_j \mathbf{L}_{j,-} \tag{A.12}$$

Perfectly inelastic collision

The expressions can be simplified if the collision is assumed to be perfectly inelastic, that is, the bodies stick together after the collision. Then the bodies will have the same linear and angular velocity after the collision such that by denoting $\mathbf{v}_+ = \mathbf{v}_{j,+}$ and $\boldsymbol{\omega}_+ = \boldsymbol{\omega}_{j,+}$ one get

$$\sum_j \mathbf{P}_{j,+} = (\sum_j m_j)\mathbf{v}_+ \tag{A.13}$$

$$\sum_j \mathbf{L}_{j,+} = (\sum_j \mathbf{I}_j)\boldsymbol{\omega}_+ \tag{A.14}$$

by inserting the expressions into Equations (A.11)–(A.12) the linear and angular velocity after the collision is

$$\mathbf{v}_+ = \frac{\mathbf{P}_-}{\sum_j m_j} = \frac{\sum_j m_j \mathbf{v}_{j,-}}{\sum_j m_j} \tag{A.15}$$

$$\boldsymbol{\omega}_+ = \left(\sum_j \mathbf{I}_j\right)^{-1} \mathbf{L}_- = \left(\sum_j \mathbf{I}_j\right)^{-1} \sum_j \mathbf{I}_{j,-}\boldsymbol{\omega}_{j,-} \tag{A.16}$$

finally by inserting into Equations (A.9)–(A.10) the average force and moment during the collision is found as

$$\bar{\mathbf{f}}_i = \frac{\sum_j m_j \mathbf{v}_{j,-} - m_i \mathbf{v}_{i,-}}{\Delta t} \tag{A.17}$$

$$\bar{\mathbf{m}}_i = \frac{\sum_j \mathbf{I}_j \boldsymbol{\omega}_{j,-} - \mathbf{I}_i \boldsymbol{\omega}_{i,-}}{\Delta t} \tag{A.18}$$

A.3 Time dependent equation of motions

The time-dependent equation of motions for the net during the collision will be derived using the methods discussed in [Fossen, 2011, Ch.3] with support from the theorems in [Sagatun & Fossen, 1991]. It is assumed that the mass is the only time-dependent variable.

The method applies the Euler-Lagrangian equations, such that by defining the kinetic energy

$$T = \frac{1}{2}\boldsymbol{\nu}^\top \mathbf{M}\boldsymbol{\nu} \tag{A.19}$$

where the mass-matrix is diagonal on the following form

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{M}_{22} \end{bmatrix} \tag{A.20}$$

by setting $\boldsymbol{\nu} = \begin{bmatrix} \mathbf{v} & \boldsymbol{\omega} \end{bmatrix}^\top$ Equation (A.19) can be simplified to

$$T = \frac{1}{2}(\mathbf{v}^\top \mathbf{M}_{11}\mathbf{v} + \boldsymbol{\omega}^\top \mathbf{M}_{22}\boldsymbol{\omega}) \tag{A.21}$$

the method in [Sagatun & Fossen, 1991] applies the Kirchiff's equation

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \mathbf{v}}\right) + \mathbf{S}(\boldsymbol{\omega})\frac{\partial T}{\partial \mathbf{v}} = \boldsymbol{\tau}_1 \tag{A.22}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \boldsymbol{\omega}}\right) + \mathbf{S}(\boldsymbol{\omega})\frac{\partial T}{\partial \boldsymbol{\omega}} + \mathbf{S}(\mathbf{v})\frac{\partial T}{\partial \mathbf{v}} = \boldsymbol{\tau}_2 \tag{A.23}$$

where $\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}_1 & \boldsymbol{\tau}_2 \end{bmatrix}^\top$ is the generalized forces applied on the system further

$$\frac{\partial T}{\partial \mathbf{v}} = \mathbf{M}_{11}\mathbf{v} \tag{A.24}$$

$$\frac{\partial T}{\partial \boldsymbol{\omega}} = \mathbf{M}_{22}\boldsymbol{\omega} \tag{A.25}$$

and

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \mathbf{v}}\right) = \dot{\mathbf{M}}_{11}\mathbf{v} + \mathbf{M}_{11}\dot{\mathbf{v}} \tag{A.26}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \boldsymbol{\omega}}\right) = \dot{\mathbf{M}}_{22}\boldsymbol{\omega} + \mathbf{M}_{22}\dot{\boldsymbol{\omega}} \tag{A.27}$$

then Equation (A.22)–(A.23) can be restated as follows

$$\mathbf{M}_{11}\dot{\mathbf{v}} + \boldsymbol{S}(\boldsymbol{\omega})\mathbf{M}_{11}\mathbf{v} + \dot{\mathbf{M}}_{11}\mathbf{v} = \boldsymbol{\tau}_{1:3} \tag{A.28}$$

$$\mathbf{M}_{22}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\boldsymbol{\omega})\mathbf{M}_{22}\boldsymbol{\omega} + \boldsymbol{S}(\mathbf{v})\mathbf{M}_{11}\mathbf{v} + \dot{\mathbf{M}}_{22}\boldsymbol{\omega} = \boldsymbol{\tau}_{4:6} \tag{A.29}$$

Then the coreolis-matrix $\mathbf{C}(\boldsymbol{\nu}, t)$ and time differentiated mass-matrix $\dot{\mathbf{M}}$ can be given as

$$\mathbf{C}(\boldsymbol{\nu}, t) = \begin{bmatrix} \mathbf{S}(\mathbf{M}_{11}\mathbf{v}) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & -\mathbf{S}(\mathbf{M}_{22}\boldsymbol{\omega}) \end{bmatrix} \tag{A.30}$$

$$\dot{\mathbf{M}} = \begin{bmatrix} \dot{\mathbf{M}}_{11} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \dot{\mathbf{M}}_{22} \end{bmatrix} \tag{A.31}$$

such that the equations of motion finally is found

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \dot{\mathbf{M}}(t)\boldsymbol{\nu} + \mathbf{C}(\boldsymbol{\nu}, t)\boldsymbol{\nu} = \boldsymbol{\tau} \tag{A.32}$$

Appendix B

Polynomial Trajectory Theory

B.1 Polynomial trajectory generation

The field of trajectory planning is well established in the field of robot control, and the polynomial approach for creating feasible trajectories is introduced in among others [Spong et al., 2006, Ch. 5].

A trajectory is defined as a path with a desired set of states for any given time. In order to generate a trajectory one can define a set of constraints on the states at specific time stamps. With the general coordinate $q \in \mathbb{R}^1$ one can define a polynomial of order N for varying τ

$$q(\tau) = \sum_{k=0}^{N} a_k \tau^k \tag{B.1}$$

further the generalized trajectory state vector $\mathbf{q}(\tau) = [q_0,\ q_1,\ \cdots q_i,\ \cdots\ q_M]^\top \in \mathbb{R}^{M+1}$ gives the M derivatives of q with respect to τ

$$\mathbf{q}(\tau) = \left[q,\ \frac{d}{d\tau}q,\ \cdots \frac{d^i}{d\tau^i}q,\ \cdots\ \frac{d^M}{d\tau^M}q\right]^\top \tag{B.2}$$

where the ith derivative is given as

$$q_i(\tau) = \sum_{k=i}^{N} a_k \frac{k!}{(k-i)!}\tau^{k-i} \tag{B.3}$$

In general a trajectory can be defined by constraining m_i of the states $\mathbf{q}$ at a set of timestamps $\{\tau_0, \tau_1, \cdots \tau_j, \cdots \tau_n\}$, such that at timestamp τ_j the ith derivative is constrained by

$$q_i(\tau_j) = \sum_{k=i}^{N} a_k \frac{k!}{(k-i)!} \tau_i^{k-i} = q_{j,i} \tag{B.4}$$

and the set of constraints at τ_j stacked in $\mathbf{q}^*(\tau_j) = \mathbf{q}_j^* \in \mathbb{R}^{m_i}$ is subset of the state vector $\mathbf{q}$. This gives a total of $M^* = \sum_{i=0}^{n} m_i$ constraints. Then a polynomial of order $N = M^* - 1$ is required in order to determine the coefficients $\{a_0, a_1, \cdots, a_N\}$.

Expanding $\mathbf{q}^*(\tau_j) = \mathbf{q}_j^*$ by utilizing Equation (B.4) and stacking the coefficients of the polynomial $\mathbf{a} = [a_0, a_1, \cdots, a_N]^T \in \mathbb{R}^{N+1}$ gives

$$\mathbf{M}(\tau_j, m_i)\mathbf{a} = \mathbf{q}_j^* \tag{B.5}$$

given for a set of constraints $\mathbf{q}_j^*$ at time τ_j. Further by assuming that the constraint is on the form $\mathbf{q}_j^* = \begin{bmatrix} q_{j,0} & q_{j,1} & \cdots & q_{j,m_i} \end{bmatrix}^\top$ the matrix $\mathbf{M}(\tau_i, m_i) \in \mathbb{R}^{m_i \times M^*}$ can be found as

$$\mathbf{M}(\tau_j, m_i) = \begin{bmatrix} 1 & \tau_j & \tau_j^2 & \cdots & \tau^N \\ 0 & 1 & \tau_j & \cdots & \frac{N!}{(N-1)!}\tau^{N-1} \\ \vdots & & & & \vdots \\ 0 & \cdots & 1 & \cdots & \frac{N!}{(N-m_i-1)!}\tau^{N-m_i-1} \end{bmatrix} \tag{B.6}$$

By stacking the constraints into a vector
$\mathbf{b} = \left[(\mathbf{q}_0^*)^\top, (\mathbf{q}_1^*)^\top \cdots (\mathbf{q}_i^*)^\top, \cdots (\mathbf{q}_n^*)^\top\right]^\top \in \mathbb{R}^{M^*}$ and setting

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}(\tau_0, m_0) \\ \vdots \\ \mathbf{M}(\tau_n, m_n) \end{bmatrix} \in \mathbb{R}^{M^* \times M^*}$$

the total system can be given on the following form

$$\mathbf{M}\mathbf{a} = \mathbf{b} \tag{B.7}$$

given that $\mathbf{M}$ is invertible the coefficients can be found

$$\mathbf{a} = \mathbf{M}^{-1}\mathbf{b} \tag{B.8}$$

then the M^* coefficients $\mathbf{a}$ can be inserted into $\mathbf{q}(\tau)$ to generate the desired trajectory.

B.2 Polynomial generation matrices

Given a desired trajectory $\mathbf{q}(\tau) \in \mathbb{R}^4$ and polynomial of order $N = 7$. Then a set of coefficients stacked in a vector $\mathbf{a} \in \mathbb{R}^{8x1}$ such that the trajectory can be defined as

$$\mathbf{q}(\tau) = \mathbf{M}(\tau, 4)\mathbf{a} \tag{B.9}$$

following the notation from Appendix B.1, where

$$\mathbf{M}(\tau, 4) = \begin{bmatrix} 1 & \tau & \tau^2 & \tau^3 & \tau^4 & \tau^5 & \tau^6 & \tau^7 \\ 0 & 1 & 2\tau & 3\tau^2 & 4\tau^3 & 5\tau^4 & 6\tau^5 & 7\tau^6 \\ 0 & 0 & 2 & 6\tau & 12\tau^2 & 20\tau^3 & 30\tau^4 & 42\tau^5 \\ 0 & 0 & 0 & 6 & 24\tau & 60\tau^2 & 120\tau^3 & 210\tau^4 \end{bmatrix} \tag{B.10}$$

the matrix $\mathbf{M}$ defining the relation between the constraint vector $\mathbf{b} \in \mathbb{R}^{8x1}$ and the coefficients $\mathbf{a}$ can be defined as follows

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}(t_0, 4) \\ \mathbf{M}(t_f, 4) \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 & t_0^6 & t_0^7 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 & 6t_0^5 & 7t_0^6 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 & 30t_0^4 & 42t_0^5 \\ 0 & 0 & 0 & 6 & 24t_0 & 60t_0^2 & 120t_0^3 & 210t_0^4 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 & t_f^6 & t_f^7 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 & 6t_f^5 & 7t_f^6 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 & 30t_f^4 & 42t_f^5 \\ 0 & 0 & 0 & 6 & 24t_f & 60t_f^2 & 120t_f^3 & 210t_f^4 \end{bmatrix} \tag{B.11}$$

Appendix C

Adaptive Control Theory

C.1 Model Reference Adaptive Control

In [Ioannou & Sun, 2012] a simple MRAC scheme is derived for a linear Multiple-Input and Multiple-Output (MIMO) system on the following state space form with $\mathbf{x} \in \mathbb{R}^n$

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{C.1}$$

where in general $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times q}$ is unknown constant matrices. $(\mathbf{A}, \mathbf{B})$ is assumed to be controllable. The reference model is also defined as a n order MIMO Linear-Time-Invariant (LTI) system

$$\dot{\mathbf{x}}_r = \mathbf{A}_r\mathbf{x}_r + \mathbf{B}_r\mathbf{r} \tag{C.2}$$

if $\mathbf{A}$ and $\mathbf{B}$ is known, and the dynamics is truly the same as stated in Equation C.1 a controller with the optimal controller parameters $\mathbf{K}^*$ and $\mathbf{L}^*$ can be chosen

$$\mathbf{u} = -\mathbf{K}^*\mathbf{x} + \mathbf{L}^*\mathbf{r} \tag{C.3}$$

and by choosing $\mathbf{K}^* \in \mathbb{R}^{q\times n}$ and $\mathbf{L}^* \in \mathbb{R}^{q\times q}$ such that $\mathbf{A} - \mathbf{B}\mathbf{K}^* = \mathbf{A}_r$ and $\mathbf{B}\mathbf{L}^* = \mathbf{B}_r$ the system state $\mathbf{x}(t) \to \mathbf{x}_r(t)$ exponentially fast for any bounded reference input signal $\mathbf{r}(t)$.

By introducing the following adaptive control law and define $\tilde{\mathbf{K}} =: \mathbf{K} - \mathbf{K}^*$ and $\tilde{\mathbf{L}} =: \mathbf{L} - \mathbf{L}^*$

$$\mathbf{u} = -\mathbf{K}(t)\mathbf{x} + \mathbf{L}(t)\mathbf{r} \tag{C.4}$$

and further by defining the error state $\mathbf{e} = \mathbf{x} - \mathbf{x}_r$ the error dynamics can be found as

$$\dot{\mathbf{e}} = \mathbf{A}_r\mathbf{e} + \mathbf{B}(-\tilde{\mathbf{K}}\mathbf{x} + \tilde{\mathbf{L}}\mathbf{r}) \tag{C.5}$$

Then [Ioannou & Sun, 2012] propose a Lyuapunov function candidate

$$V(\mathbf{e}, \tilde{\mathbf{K}}, \tilde{\mathbf{L}}) = \mathbf{e}^T \mathbf{P} \mathbf{e} + \operatorname{tr}\left[\tilde{\mathbf{K}}^T \mathbf{\Gamma} \tilde{\mathbf{K}} + \tilde{\mathbf{L}}^T \mathbf{\Gamma} \tilde{\mathbf{L}}\right] \tag{C.6}$$

where $\mathbf{\Gamma}^{-1} = \mathbf{L}^* \operatorname{sgn}(l)$ and l is chosen based on whether $\mathbf{L}^*$ is positive definite or not, that is

$$l = \begin{cases} 1 & \text{if } \mathbf{L}^* > 0 \\ -1 & \text{if } \mathbf{L}^* < 0 \end{cases}$$

further $\mathbf{P} = \mathbf{P}^T > 0$ satisfies the Lyapunov equation

$$\mathbf{P}\mathbf{A}_r + \mathbf{A}_r^T \mathbf{P} = \mathbf{Q} \tag{C.7}$$

by choosing some $\mathbf{Q} = \mathbf{Q}^T > 0$, then by choosing the following adaption laws

$$\dot{\tilde{\mathbf{K}}} = \mathbf{B}_r^T \mathbf{P} \mathbf{e} \mathbf{x}^T \operatorname{sgn}(l) \tag{C.8}$$

$$\dot{\tilde{\mathbf{L}}} = -\mathbf{B}_r^T \mathbf{P} \mathbf{e} \mathbf{r}^T \operatorname{sgn}(l) \tag{C.9}$$

it can be shown that

$$\dot{V} = -\mathbf{e}^T \mathbf{Q} \mathbf{e} \tag{C.10}$$

Then [Ioannou & Sun, 2012] proofs that $\mathbf{K}(t)$,$\mathbf{L}(t)$,$\mathbf{e}(t)$ are bounded and that $\mathbf{e}(t) \to 0$.

C.2 Controllability

Given a continuous time LTI system with the state $\mathbf{x} \in \mathbb{R}^n$ and input $\mathbf{u} \in \mathbb{R}^k$ on the following state-space form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{C.11}$$

Then the concept of *controllability* determine if it is possible to reach any desired state $\mathbf{x}$ from an initial state $\mathbf{x}(t_0) = \mathbf{x}_0$ given an external input $\mathbf{u}$ in finite-time. In among other [Hespanha, 2009] a simple method for checking the controllability is presented. First, the *controllability matrix* for the LTI system in Equation (C.11) is defined as

$$\mathbb{C} := \begin{bmatrix} \mathbf{B} & \mathbf{A}\mathbf{B} & \mathbf{A}^2\mathbf{B} & \cdots & \mathbf{A}^{n-1}\mathbf{B} \end{bmatrix}_{n \times (kn)} \tag{C.12}$$

then the system is said to be controllable if $\operatorname{rank}(\mathbb{C}) = n$, that is, $\mathbb{C}$ has full row rank with n linearly independent columns.

Appendix D

Controllers – Tuning considerations

D.1 Alternative velocity control

The velocity controller in Section 5.2.3 controls the individual multirotor body velocities. For all simulations and experiments the controller gains were the same along all axes, hence, the effect of using the body frame did not appear. However, regarding the recovery maneuver, one might want to make the along-track controller more aggressive than the cross-track velocity controller. Therefore this section presents an optional controller in the centroid frame $\{\bar{c}\}$.

For each multirotor i a velocity controller is implemented which gives the desired force in the centroid frame $\{c_i\}$. For simplified notation, i is used to denote c_i in the following derivation.

The desired heading and velocities from the guidance controllers are given in the centroid frame $\{\bar{c}\}$ and must be rotated back to each individual multirotor frame $\{i\}$ by the following rotation

$$\mathbf{R}_{\bar{c}}^{i} = \mathbf{R}_{n}^{i}\mathbf{R}_{\bar{c}}^{n} = (\mathbf{R}_{i}^{n})^{\top}\mathbf{R}_{z}(\psi_{\bar{c}}) \tag{D.1}$$

where $\psi_{\bar{c}}$ is the actual heading of the centroid. Then a velocity controller is proposed which gives $\mathbf{f}_{i}^{\bar{c}}$ the desired force applied on multirotor i given the velocity feedback $\mathbf{v}_{i}^{i}$

$$\mathbf{f}_{i}^{\bar{c}} = \mathbf{K}_{p}(\mathbf{v}_{d}^{\bar{c}} + \mathbf{R}_{n}^{\bar{c}}\mathbf{v}_{i,df}^{n} - \mathbf{R}_{i}^{\bar{c}}\mathbf{v}_{i}^{i}) + m_{i}\dot{\mathbf{v}}_{d}^{\bar{c}} \tag{D.2}$$

where $\mathbf{K}_{p} \in \mathbb{R}^{3\times 3}$ is a positive definite tuning matrix. Further the desired velocity $\mathbf{v}_{d}^{\bar{c}}$ and acceleration $\dot{\mathbf{v}}_{d}^{\bar{c}}$ for the centroid is given from the guidance law, lastly $\mathbf{v}_{i,df}^{n}$ gives the desired velocity from the formation controller as seen in Section 5.2.1.

Then the desired force $\mathbf{f}^{i}$ can be rotated to $\{n\}$

$$\mathbf{f}_{i}^{n} = \mathbf{R}_{\bar{c}}^{n}\mathbf{f}_{i}^{\bar{c}} \tag{D.3}$$

D.2 Closed-loop dynamics

For the following analysis the inner velocity controller above in Appendix D.1 will be used together with the feed-forward term from Section 5.2.3 together with the model from Section 4.4. Then the closed-loop model for the inner-loop controller for multirotor i can be found as the following, where the formation controller term has been neglected.

$$m_i \dot{\mathbf{v}}_i^{\bar{c}} = \mathbf{K}_p(\mathbf{v}_d^{\bar{c}} - \mathbf{v}_i^{\bar{c}}) + m_i \dot{\mathbf{v}}_d^{\bar{c}} \tag{D.4}$$

setting $\mathbf{e}_i = \mathbf{v}_d^{\bar{c}} - \mathbf{v}_i^{\bar{c}}$ gives the following closed loop dynamics

$$\dot{\mathbf{e}}_i + \frac{1}{m_i}\mathbf{K}_p \mathbf{e}_i = 0 \tag{D.5}$$

which renders the error $\mathbf{e}_i$ exponentially fast to zero with the timeconstants $\frac{1}{m_i}\mathbf{K}_p$.

However, the inner control loop renders each individual multirotor i to the desired mission velocity $\mathbf{v}_d^{\bar{c}}$ and the desired formation. The outer position loops in Section 7.3 and Section 7.4 seeks to control the centroid velocity only. Hence, setting up the closed-loop dynamics from the desired position to the actual centroid position requires the closed loop dynamics of all individual multirotors giving the dynamics of the centroid.

The analysis can be simplified by utilizing successive loop closure as discussed in e.g. [Beard & McLain, 2012, Ch. 12]. If one assume that the inner loop is sufficiently fast, that is, the bandwidth of the inner loop is higher than the outer, the inner loop dynamics can be neglected for the outer control loop. Hence, when analyzing e.g. the outer position control loop for the centroid in Equation (7.37) one can assume that the inner control loop renders the desired velocity $\mathbf{v}_d^p$ to the actual centroid velocity $\mathbf{v}^p$ instantly, giving $\mathbf{v}^p \approx \mathbf{v}_d^p$. By setting $\mathbf{e}_p = \mathbf{p}_{d,\bar{c}}^p - \mathbf{p}_{\bar{c}}^p$, the position closed-loop dynamics can be found as follows

$$\ddot{\mathbf{e}}_p + \mathbf{K}_{p,d}\dot{\mathbf{e}}_p + \mathbf{K}_{p,p}\mathbf{e}_p = 0 \tag{D.6}$$

Comparing with the following second order system with the natural frequency ω_0 and relative damping factor ζ

$$\ddot{\mathbf{e}}_p + 2\zeta\omega_0\dot{\mathbf{e}}_p + \omega_0^2\ddot{\mathbf{e}}_p = 0 \tag{D.7}$$

the method as discussed in [Fossen, 2011, Ch. 12] and Section 6.2 can be used to tune the outer loop.

$$\mathbf{K}_{p,d} = \omega_0^2\mathbf{I}_{3\times3} \tag{D.8}$$

$$\mathbf{K}_{p,p} = 2\zeta\omega_0\mathbf{I}_{3\times3} \tag{D.9}$$

Here, only scalar elements are specified for ω_0 and ζ, however, they can be in general be specified for each axis.

Appendix E

Submitted conference paper for ICUAS'16

This appendix contains the conference paper submitted for *The 2016 International Conference on Unmanned Aircraft Systems* (ICUAS'16).

Coordinated Control Concept for Recovery of a Fixed-Wing UAV on a Ship using a Net Carried by Multirotor UAVs

Kristian Klausen*, Jostein Borgen Moe*, Jonathan Cornel van den Hoorn†, Alojz Gomola‡, Thor I. Fossen*, Tor Arne Johansen*
*Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics,
*NTNU, Norwegian University of Science and Technology, Norway
†TU Delft
‡Department of Electrical and Computer Engineering, University of Porto, Portugal
Email: *kristian.klausen@ntnu.no

***Abstract*—Ship-based Unmanned Aerial Vehicle (UAV) operations is an important field of research, and enables many mission types. Most of them, because of the endurance requirement, require the use of a fixed-wing UAV. Traditionally, a net located on the ship deck is used for recovering the fixed-wing UAV. There are, however, numerous challenges when attempting autonomous landings in such environments. Waves will induce heave motion, and turbulence near the ship will make approaches challenging. In this paper, we present a concept using multirotors to move the recovery operation off the ship deck. To recover the fixed-wing UAV, a net is suspended below two coordinated multirotor UAVs. By synchronizing the movement with the fixed-wing UAV, the multirotor UAVs can carry a net located away from the ship deck, and the approach trajectory can be optimized with respect to the wind direction and turbulence caused by the ship can be avoided. In addition, the multirotors can transport the net at a certain speed along the trajectory of the fixed-wing UAV, thus decreasing the relative velocity between the net and fixed wing UAV to reduce the forces of impact. This paper proves the concept through a simulation study and a preliminary control system architecture.**

I. Introduction

An increased effort has been made to enable autonomous operations with UAVs in marine environments. Here, UAVs are typically used for surveillance, data-acquisition or communication relaying.

Fixed-wing UAVs are often launched with a catapult-like device, powered by either pneumatics, springs or rubber bands. These devices have a reasonable footprint, and are quite popular. For recovery a fixed net can be used. To accommodate this however, a relatively large part of the ship needs to be set aside for UAV operations. This has a large footprint, and for safety reasons a large part of the deck needs to be emptied for both equipment and other personell. Depending on the construction of the net, the UAV has a risk for damage due to the impact with the fixed net. More importantly, waves will induce oscillatory heave motion on the ship, and the nets location and attitude may not always be optimized with respect to wind and turbulence near the ship due to the requirements of other ship operations.

This work was partially supported by the Research Council of Norway through its Centers of Excellence funding scheme, grant number 223254 (NTNU Centre for Autonomous Marine Operations and Systems).

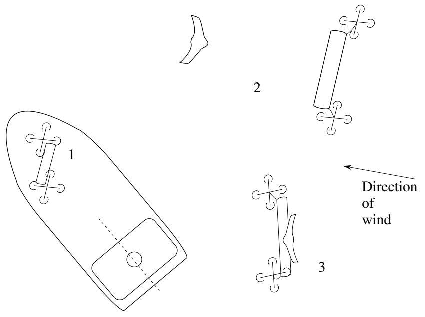


Fig. 1. The figure illustrates the recovery of a fixed-wing UAV. (1), the multirotors take off from the ship. (2), the fixed-wing UAV moves against the wind direction, while the multirotors position the net along its trajectory and accelerate to a prescribed velocity in order to catch the incoming fixed-wing UAV. (3), the multirotors are transporting the fixed-wing UAV back to the ship.

On the topic of recovering fixed-wing UAVs, autonomous landing based on e.g. GNSS [1] or visual servoing [2] have been studied in the literature. For ship-based operations, several methods including nets, hooks and wires have been pursued. Most notably is the SkyHook system developed by Insitu [3]. It consists of two components; a vertical wire attached to a mobile or fixed structure, and a hook on the wing tip of the fixed-wing UAV. When it is passing the wire, the wing gets hooked on tight. This system is commercially available today, with the specially designed ScanEagle UAV. This concept has been expanded by having the wire suspended in air by a heavy-duty multirotor [4]. Further, a concept for landing larger UAVs by using a horizontal wire and a hook, is presented in [5]. In stead of using nets or hooks, the high braking capabilities of a fixed-wing UAV in *deep stall* [6], [7] can be utilized for landing. In [8], the non-linear dynamics of a fixed-wing UAV during deep stall is analyzed and controlled using model predictive control. However, autonomous landing by a deep stall maneuver requires accurate models and high accuracy, and are especially prone to changing wind conditions.

In this paper, we present an approach for landing a small fixed-wing UAV in a net suspended by two powerful (> 10 kg *maximum takeoff weight*) multirotor UAVs. The fixed-wing UAV is equipped with hooks so that after impact with the net, it will be arrested by the net to be transported back to the ship, see Figure 1. Key benefits of such an approach include

- *Operational flexibility:* When recovering a fixed-wing UAV, it is crucial to travel against the wind to minimize the ground speed, and thus a fixed net needs to be aligned with this path. Even in vessels equipped with Dynamic Positioning (DP) systems, turning the ship can be undesired as it may interfere with operations. The multirotors can however quickly react to changing wind conditions, and align the net against the wind without interfering with other ship operations.
- *Not affected by waves and turbulence:* Since the net is suspended free from the ship, heave motion induced by waves on the ship will not affect the landing. Also, there is no impact from turbulence caused by the ship superstructure.
- *Safety:* By having the net suspended by two multirotor UAVs, the recovery operation can be moved off ship. Thus, no operators or staff risk coming in contact with the incoming UAV.
- *Smaller impact force:* By having the two multirotors move against the wind with the fixed-wing UAV, the relative speed difference between it and the net can be made smaller, thus decreasing the structural load on the fixed-wing body during impact.
- *Smaller footprint:* By moving the landing operation off ship, operations with UAVs can be conducted from smaller ships, not needing a large open deck with a net to support the mission. Launch and recovery of the multirotors are still required.

Recovery with nets suspended by multirotors have been attempted in various settings. Due to the popularity of consumer-type multirotor UAVs, there is an increased interest in the ability to safely remove such vehicles from restricted airspaces. In [9], a multirotor is equipped with a net gun, capable of incapacitating smaller multirotors by shooting a net at them to disable the rotors on the target. A similar experiment was conducted in [10], where the target multirotor stays attached to the larger multirotor after the net is fired, see also [11], [12]. But to the best of the authors knowledge, no attempts to recover fixed-wing UAVs in a net suspended between multirotor UAVs have yet been published.

The contributions of this paper is twofold. First, it presents a controller structure for the net recovery concept, where the main contribution consists of how to combine existing control methodologies to a complete system. Next, we present the results from numerical simulations, which gives insight into the dynamics during the recovery maneuver. In addition, we give an overview of an implementation of the controller.

A. Organization

This paper is organized as follows. Section II gives an overview of the maneuver and proposed control structure, followed by more details of each part of the controller in Section III. This section also introduces the necessary notation and dynamical models. In Section IV, the multi-body dynamics of the suspended net is discussed to create a simulator with 6 degrees of freedom (DOF) of all involved objects. A model of the impact dynamics is developed in Section IV-B. The results of the simulation is presented in Section V, which also shows the results of the proposed controller. Section VI gives an overview of the systems architecture of the proposed design, including necessary hardware. We also present the results of a Software-In-the-Loop (SIL) simulation, to verify the implementation. This is the same setup to be used in experiments. Section VIII gives a brief summary and concludes the paper.

II. Autonomous Net Recovery Concept

Autonomous recovery of a fixed-wing UAV in a suspended net is a complex task, so the functionality is split into several key components. The overall mission is executed in the following fashion:

- The fixed-wing UAV is instructed to follow a path against the wind, with the minimal airspeed required for safe flying. This is called the *virtual runway*, and the path is transmitted to the multirotor UAVs.
- Both multirotors are equipped with coordinated controllers that keep the inter-formation of the two intact, while lifting the suspended net.
- The current position and the velocity of the fixed-wing UAV is transmitted to a coordination controller in one of the multirotors, which sends desired setpoints to the formation controllers according to the phases of the mission, as to catch the fixed-wing UAV.

Although using two multirotor UAVs instead of one increases the complexity of the system, it has several practical advantages. First, by distributing the load, each multirotor can be physically smaller than a single with the combined lift capacity. Further, the two multirotors can spread the net without a support structure (top beam), giving reduced weight of the net.

Precise navigation is crucial for precision landing of UAVs. In this work, we utilize *Real-Time Kinematic* (RTK) *Global Navigation Satellite System* (GNSS). This is a navigation technique using the carrier wave of the incoming signals from the satellites, and comparing the signals to that received by a base station. By computing the phase shift between the signals at the UAV (rover) and the base, the location can be locked in at centimeter-level accuracy. Such a system was used in [13] for landing a fixed-wing UAV in a stationary net, which also contains more detailed information about RTK GNSS systems.

III. Control design

This section introduces the control design, and gives details about each of the different parts. The overall structure can be seen in Figure 2. As can be seen, there are three distinct control modules, which are detailed next. We also introduce the concept of the *virtual runway*.

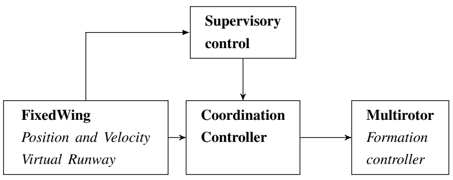


Fig. 2. Information flow in the controller structure. Based on the current position of the fixed-wing UAV, the supervisor starts the net-recovery maneuver. The coordination controller guides the two multirotors along a recovery path to intercept the fixed-wing UAV.

Note that the fixed-wing UAV acts as a reference generator (master) in the proposed control scheme, as it is not affected by the current position of the multirotors. Depending on the type of fixed-wing UAV used, it is preferred to keep a steady flight envelope, rather than correcting minor deviations from the net position. This is much better handled by the agility of the multirotors.

We assume that the fixed-wing UAV moves with a constant course and altitude along a virtual runway, and its position and velocity is communicated to the other vehicles. This is controlled by an on-board autopilot.

In the next sections, let $\mathbf{p}_i^n \in \mathbb{R}^3, i \in \{1, 2\}$ be the position of multirotor i in the inertial frame $\{n\}$. Further, we define the position $\bar{\mathbf{p}}^n$ as the centroid of the two multirotors plus an height offset to compensate for the position of the net. Further, the states of the fixed-wing UAV is denoted with subscript $\cdot_f$.

A. Virtual runway

Figure 3 illustrates the *virtual runway* (VR). The virtual runway defines a path frame $\{p\}$ at constant altitude, which is defined by an origin $\mathbf{p}_{p/n}^n$ and a rotation ψ around the $\{n\}$ z-axis such that $\mathbf{R}_p^n = \mathbf{R}_z(\psi)$. Then a position $\mathbf{p}^n$ can be decomposed in $\{p\}$ by the transformation $\mathbf{p}^p = (\mathbf{R}_p^n)^\top(\mathbf{p}^n - \mathbf{p}_{p/n}^n)$. By dividing the path frame into a *cross-track* plane and an *along-track* distance, we can design controllers for each part separately.

B. Supervisor

The supervisor monitors the position and velocity of the fixed-wing UAV relative to the virtual runway in order to switch between the different modes in the maneuver. Each mode enables a certain controller and reference which gives

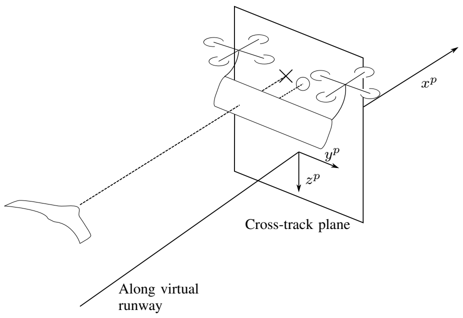


Fig. 3. Illustration of the *virtual runway*. The runway defines a *path frame* $\{p\}$, and can be divided into a cross-track plane ($y^p z^p$) and an along-track distance x^p. The position of the net on the cross-track plane is marked with a circle, while the intersection of the cross-track plane and the path of the fixed-wing UAV is marked with a cross.

a desired velocity setpoint. Figure 4 gives an overview of the different states of the supervisor. As can be seen in Figure 5, the supervisor also controls when to activate the two parts of the coordination controller.

In addition, the supervisor monitors the maneuver as it is progressing. If, because of wind or other factors, the fixed-wing UAV misses the net, it instructs the vehicles to try the maneuver again. Further, if the projected proximity of the fixed-wing UAV and multirotors are to small, the supervisor can abort the operation. Depending on the situation, an abort can involve the multirotors to climb and reposition for a retry, or releasing the net and abort the mission entirely.

C. Coordination - Cross-track

The position of the net is controlled according to the fixed-wing UAV position in the cross-track plane along the virtual runway. A cross-track frame $\{p_*\}$ is defined as the yz-plane in the path frame $\{p\}$, such that there exist a mapping from a position $\mathbf{p}^p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^\top$ to $\mathbf{p}^{p_*} = \mathbf{p}_{2:3}^p = \begin{bmatrix} p_y & p_z \end{bmatrix}^\top \in \mathbb{R}^2$. Then a modified pure-pursuit [14] scheme is introduced. Given a desired position $\mathbf{p}_d^{p_*}$ and the position error $\tilde{\mathbf{p}}^{p_*} := \mathbf{p}_d^{p_*} - \mathbf{p}^{p_*}$ the following controller is used

$$\mathbf{v}_d^{p_*} = \mathbf{K}_{p,p}\tilde{\mathbf{p}}^{p_*} + \mathbf{K}_{d,p}\dot{\tilde{\mathbf{p}}}^{p_*} + \mathbf{K}_{i,p}\int_0^t \tilde{\mathbf{p}}^{p_*}\, dt \tag{1}$$

where $\mathbf{K}_{j,p} \in \mathbb{R}^{2\times 2}$ for $j \in \{p, i, d\}$. The desired position $\mathbf{p}_d^{p_*} = \mathbf{p}_f^{p_*} = \mathbf{p}_{f,2:3}$ is defined as the current position of the fixed-wing UAV projected along the virtual runway to the cross-track plane.

It should be noted that the net position is not measured explicitly, and furthermore it is not a desirable control target as the net will swing during the transit. Therefore we seek to control the position $\bar{\mathbf{p}}$ as illustrated in Figure 3 as the circle in the cross-track plane. Hence, $\mathbf{p}^{p_*} = \bar{\mathbf{p}}_{2:3}$.

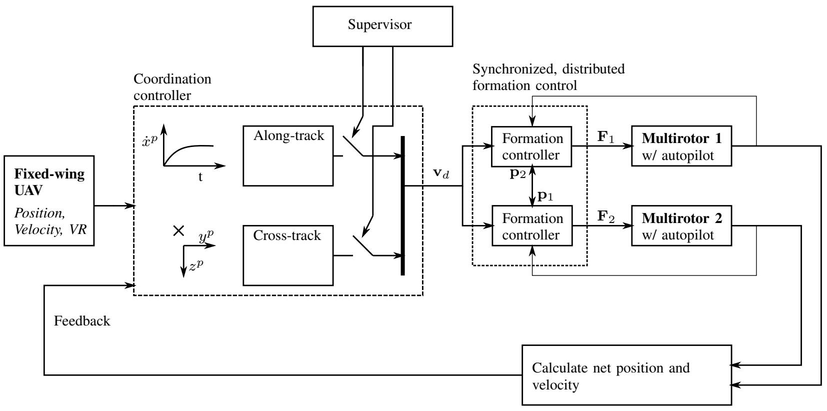


Fig. 5. Illustration of the control-structure. The cross-track controller gets feedback from the position of the fixed-wing UAV in a plane orthogonal to the virtual runway. The along-track controller is an open-loop controller, initiated by the supervisor when the fixed-wing UAV reaches the virtual runway.

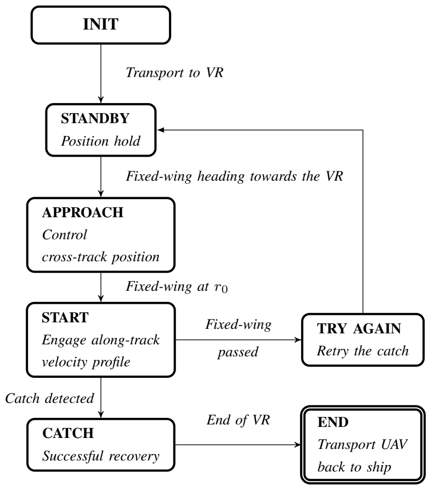


Fig. 4. Supervisor state-machine. The supervisor monitors the position of the fixed-wing UAV along the *virtual runway* (VR), and starts the coordination controller. When the along-track distance reaches r_0 (Figure 6), the START state is initiated.

D. Coordination - Along-track

The relative velocity between the net and the fixed-wing UAV is reduced by accelerating the net to a desired velocity. In order to control the point of impact an open loop scheme is proposed.

Firstly, the along-track velocity of the fixed-wing UAV is assumed to be constant, then a desired along-track velocity profile is defined. By integrating these the along-track position profile can be found analytically for both vehicles. Finally the desired catch point $r_c = r_c(t_f)$ is given as a function of the desired start point $r_0 = r_0(t_0)$ as illustrated in Figure 6.

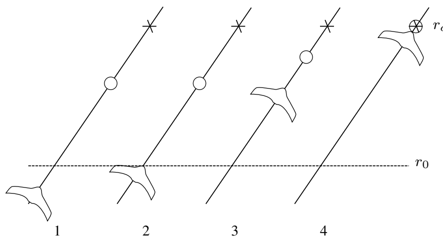


Fig. 6. Figure showing the timing of the along-track velocity, where the current position of the net is marked with a circle at different instances of time (1)-(4). When the fixed-wing UAV reaches r_0, the multirotors starts the velocity profile for forward flight as to intercept the fixed-wing UAV at r_c.

Different methods can be used to create a feasible velocity profile as the ultimate goal is to be able to calculate r_0.

These include, among others, ramps and reference models. Here a polynomial approach will be derived. First, we define the desired along-track velocity profile $v^p_{d,x}(t) := v(t)$ as an Nth-order polynomial

$$v(t) = \sum_{k=0}^{N} a_k (t - t_0)^k \tag{2}$$

Next, we seek to constrain the position, velocity, acceleration and jerk along that trajectory. This trajectory $\mathbf{q}(t) =$

$\begin{bmatrix} p(t) & v(t) & a(t) & j(t) \end{bmatrix}^\top \in \mathbb{R}^4$ can be found by integrating and differentiating (2). Then, by defining the constraints $\mathbf{q}(t_0)$ and $\mathbf{q}(t_f)$, the N coefficients a_k can be found as derived in e.g. [15]. With a total of 8 constraints a polynomial of order 7 is required. In Figure 7 an example of such an trajectory is shown.

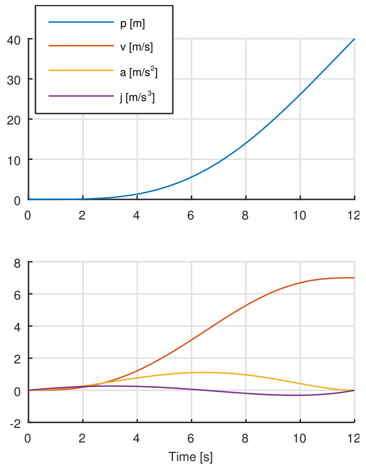


Fig. 7. Along-track polynomial velocity profile. Here the desired duration $t_f - t_0 = 12$, constraints $\mathbf{q}(t_0 = 0) = \mathbf{0}$ and $\mathbf{q}(t_f) = \begin{bmatrix} 40, & 7, & 0, & 0 \end{bmatrix}^\top$ are specified.

By combining the desired velocity from the cross-track and along-track control we get $\mathbf{v}_d^p = \begin{bmatrix} v_{d,x}^p(t) & (\mathbf{v}_d^{p_*})^\top \end{bmatrix}^\top$, and the resulting desired velocity in $\{n\}$ can be found by the following transformation

$$\mathbf{v}_d^n = (\mathbf{R}_n^p)^\top \mathbf{v}_d^p \tag{3}$$

which gives the desired velocity for the two multirotors, and is applied to the formation in the next section.

E. Multirotor Modeling

In this section, we start by presenting the dynamical model of a multirotor UAV, as in [16]. The model can be derived by Newtonian or Lagrangian methods, and readers are referred to [17] for details on its derivation. By further assuming the presence of an internal attitude controller, the relevant dynamics for the control design is extracted.

Let the dynamics of multirotor i be modeled by

$$\dot{\mathbf{p}}_i = \mathbf{v}_i \tag{4}$$
$$m_i \dot{\mathbf{v}}_i = m_i \mathbf{g} + \mathbf{R}_i f_i \tag{5}$$
$$\dot{\mathbf{R}}_i = \mathbf{R}_i \mathbf{S}(\boldsymbol{\omega}_i) \tag{6}$$
$$\mathbf{I}_i \dot{\boldsymbol{\omega}}_i = \mathbf{S}(\mathbf{I}_i \boldsymbol{\omega}_i)\boldsymbol{\omega}_i + \mathbf{M}_i \tag{7}$$

where $\mathbf{p}_i \in \mathbb{R}^3$ is the UAV position in the inertial frame $\{n\}$, $\mathbf{v}_i \in \mathbb{R}^3$ the translational velocity in $\{n\}$, $\mathbf{R}_i \in \mathcal{SO}^3$ a rotation matrix from the body-fixed frame $\{b_i\}$ to the inertial frame $\{n\}$, $\omega_i \in \mathbb{R}^3$ the angular velocity of the UAV, represented in $\{b_i\}$. Further, the operator $\mathbf{S}(\cdot)$ is the skew-symmetric transformation, such that $p \times q = \mathbf{S}(p)q$. m_i is the mass of the multirotor, and I_i the body-fixed inertia matrix. f_i is upwards thrust directed along the negative body-aligned z-axis, $\mathbf{M}_i$ are applied moment about the multirotor centre of gravity, and $\mathbf{g} = [0 \; 0 \; g]^\top$ where g is the gravitational constant.

Consider now the net being suspended in the centre of gravity of the UAV. This will affect the translational motion (5) by a force $\boldsymbol{\tau}_{L,i}$, given by the load dynamics, but the rotational motion (7) is unaffected. As control of the attitude of the multirotor is not considered in this paper, the model considering the translational motion is now

$$m_i \dot{\mathbf{v}}_i = m_i \mathbf{g} + \mathbf{R}_i f_i + \boldsymbol{\tau}_{L,i} \tag{5b}$$

Further, assume now that a sufficiently fast attitude controller is present. The direction of the applied force for translational motion (5) is given by $\mathbf{R}_i$, and by manipulating the roll and pitch of the UAV we can apply force in a desired direction. An example of such a controller is given in [17], and a similar controller is assumed present in the multirotor autopilot. Thus, the term $\mathbf{R}_i f_i$ can be replaced by an inertial control force $\mathbf{F}_i \in \mathbb{R}^3$, resulting in the linear dynamics

$$m_i \dot{\mathbf{v}} = m_i \mathbf{g} + \mathbf{F}_i + \boldsymbol{\tau}_{L,i} \tag{5c}$$

F. Formation Control of Two Multirotors

The formation controller is designed in two steps, following the procedure outlined in [18]. This is a passivity-based approach, where an inner loop controller takes a velocity setpoint from an outer controller, and the stability of the cascaded structure is proved by passivity theory. While the inner controller uses only its own measurements, the outer uses available information from the other multirotor to reach the desired formation.

First, let

$$\mathbf{F}_i = \boldsymbol{\tau}_{L,i} + m_i \mathbf{g} - \mathbf{K}_i(\mathbf{v}_i - \mathbf{v}_d) + m_i \dot{\mathbf{v}}_d + \mathbf{u}_i \tag{8}$$

where $\mathbf{v}_d$ is the desired common velocity from the coordination controller, known to both vehicles. $\mathbf{u}_i \in \mathbb{R}^3$ is the input from the outer loop formation controller, which acts as an injection to achieve a desired formation, to be specified later. Note that we assume we can measure the disturbance force $\boldsymbol{\tau}_{L,i}$ of the suspended net, so it can be compensated using feed-forward by the controller as discussed in Section VI-D.

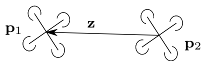


Fig. 8. Illustration of the vector between the two multirotors.

Next, let $\mathbf{z} := \mathbf{p}_1 - \mathbf{p}_2$ be the vector between the two multirotors in $\{n\}$, as in Figure 8. Let the desired value $\mathbf{z}_d$

be given by a length l and rotation ψ_n about $\vec{z}$ as

$$\mathbf{z}_d = \begin{bmatrix} l\cos(\psi_n) & l\sin(\psi_n) & 0 \end{bmatrix}^\top \tag{9}$$

The standard linear consensus protocol ([18]) can now be applied as

$$\mathbf{u}_i = d_i \mathbf{K}_p(\mathbf{z} - \mathbf{z}_d) \tag{10}$$

where $d_1 = 1, d_2 = -1$. We can now state the main result of this section:

Proposition 1. *The inner control loop* (8) *combined with the consensus protocol* (10) *ensures that the equilibrium point*

$$\mathbf{v}_1 = \mathbf{v}_2 = \mathbf{v}_d \tag{11}$$

$$\mathbf{z} = \mathbf{z}_d \tag{12}$$

is globally asymptotically stable.

Proof. Direct application of Corollary 2.2 in [18]. □

IV. Multi-body Simulation Model

This section discusses the dynamical models of the combined multirotor-net system. To provide a thorough understanding of the dynamical motion during the recovery maneuver, we developed a simulator that includes the full 6-DOF dynamics of the multirotors, fixed-wing UAV, and the net suspended under the multirotors. We also model the impact forces during collision.

A. Tension from the Suspended Net

By having the net attached to the multirotors, we have in effect a system of constrained motion where each wire connecting the net removes one degree of freedom. We consider the net being a rigid body. To model this behavior, one can reduce the state-space and use only *generalized coordinates* that cover the *configuration space* [19]. This, however, will hide the forces acting on the wires during impact. Instead, we chose to model the interconnected system with *constrained* coordinates. The Udwadia-Kalaba equation is presented in [20], which is a way to explicitly calculate the forces of constraints acting on each body. This methodology was used in [21] to develop equations of motions for helicopter slung load systems and [22] for multirotors. We follow a similar approach, described in this section. For textual brevity, we have omitted the torques resulting by the constraints due to attaching wires away from the centre of gravity of each body. They are, however, included in the numerical simulations conducted in Section V.

For generality, we consider the case with N multirotors connected to a common suspended load (in our case, a net). Let wire i, connecting body (multirotor) i to the load. A vector along the wire is given by

$$\mathbf{L}_i^n = \mathbf{p}_i - \mathbf{p}_n \tag{13}$$

where $\mathbf{p}_n$ is the position of the suspended load in $\{n\}$. A constraint g_i acting on body i and the load is given by

$$g_i = ||\mathbf{L}_i||^2 - d_i^2 = 0 \tag{14}$$

where d_i is the nominal length of wire i. (14) can be differentiated twice to obtain

$$\dot{g}_i = 2\dot{\mathbf{L}}_i^\top \mathbf{L}_i \tag{15}$$

$$\ddot{g}_i = 2\ddot{\mathbf{L}}_i^\top \mathbf{L}_i + 2\dot{\mathbf{L}}_i^\top \dot{\mathbf{L}}_i = 0 \tag{16}$$

By defining the concatenated position- and velocity vector $\mathbf{p} := [\mathbf{p}_1^\top, \ldots, \mathbf{p}_N^\top]^\top$, $\mathbf{v} := [\mathbf{v}_1^\top, \ldots, \mathbf{v}_N^\top]^\top$, the constraint can now be put on *standard form* [19]:

$$\mathbf{A}_i(\mathbf{p}, \mathbf{v})\dot{\mathbf{v}} = \mathbf{b}_i(\mathbf{p}, \mathbf{v}) \tag{17}$$

where

$$\mathbf{A}_i = 2\mathbf{L}_i^\top \begin{bmatrix} \mathbf{0}_{3\times3(i-1)} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3(N-i)} & -\mathbf{I}_{3\times3} \end{bmatrix} \tag{18}$$

and

$$\mathbf{b}_i = -\dot{\mathbf{L}}_i^\top \dot{\mathbf{L}}_i \tag{19}$$

According to [23], the constraint forces $\boldsymbol{\tau}_L$ acting on all the bodies are now given by:

$$\boldsymbol{\tau}_L = \mathbf{M}^{1/2}(\mathbf{A}\mathbf{M}^{-1/2})^+(\mathbf{b} - \mathbf{A}\dot{\mathbf{v}}) \tag{20}$$

where $\mathbf{A}$ and $\mathbf{b}$ are concatenations of $\mathbf{A}_i$ and $\mathbf{b}_i$, respectively, $(\cdot)^+$ denotes the MoOre-Penrose pseudo inverse, and $\mathbf{M}$ is a diagonal matrix with the masses of the involved bodies. To include the effects on the attitude dynamics by torques from attachment points, one can follow a similar procedure as listed above. These are more involved expressions, as they include states in the body-fixed coordinate systems. Readers are referred to [23] and [21] for details on derivation. Note that the results from the numerical simulations presented in the next section, includes the full attitude dynamics as well.

B. Modeling the Fixed-Wing UAV Impact

This section studies the dynamics during the impact, when the fixed-wing UAV gets arrested by the suspended net. The collision is assumed to be perfectly inelastic such that the bodies will stick together after the collision. In order to calculate the forces and moments on the suspended net, conservation of momentum is applied. The impact is assumed to affect the system in the timespan $t \in [t_-, t_+]$, where we define the duration $\Delta t = t_+ - t_-$.

Further it assumed that linear and angular momentum, denoted as $\mathbf{P} = m\mathbf{v}$ and $\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$ respectively, is conserved. Thus, the sum of momentum directly after $()_+$ and before $()_-$ is equal, with:

$$\mathbf{P}_+ = \mathbf{P}_- = m_f \mathbf{v}_{f,-} + m_n \mathbf{v}_{n,-} \tag{21}$$

$$\mathbf{L}_+ = \mathbf{L}_- = \mathbf{I}_f \boldsymbol{\omega}_{f,-} + \mathbf{I}_n \boldsymbol{\omega}_{n,-} \tag{22}$$

where n denotes the suspended load. Next, the common velocity $\mathbf{v}_+$ and angular velocity $\boldsymbol{\omega}_+$ after the collision can be found by

$$\mathbf{v}_+ = \frac{\mathbf{P}_-}{m_f + m_n} \tag{23}$$

$$\boldsymbol{\omega}_+ = (\mathbf{I}_f + \mathbf{I}_n)^{-1}\mathbf{L}_- \tag{24}$$

Finally, the average forces and moments applied to the suspended load can be found by utilizing the linear and angular impulse law. The linear and angular impulse $\mathbf{J} = \Delta\mathbf{P} = \mathbf{P}_+ - \mathbf{P}_-$ and $\mathbf{H} = \Delta\mathbf{L} = \mathbf{L}_+ - \mathbf{L}_-$ gives the change in momentum. The laws are coupled with forces and moments $\boldsymbol{\tau} = \begin{bmatrix}\mathbf{f}^\top & \mathbf{m}^\top\end{bmatrix}^\top$ over the timespan t_- to t_+ as follows

$$\begin{bmatrix}\mathbf{J}\\ \mathbf{H}\end{bmatrix} = \int_{t_-}^{t_+} \boldsymbol{\tau}\, dt = \bar{\boldsymbol{\tau}}\Delta t \tag{25}$$

Then, the average forces and moments on the suspended load $\bar{\boldsymbol{\tau}}_n$ is given as:

$$\bar{\boldsymbol{\tau}}_n = \frac{1}{\Delta t}\begin{bmatrix}(m_f + m_n)\mathbf{v}_+ - m_n\mathbf{v}_{n,-}\\ (\mathbf{I}_f + \mathbf{I}_n)\boldsymbol{\omega}_+ - \mathbf{I}_n\boldsymbol{\omega}_{n,-}\end{bmatrix} \tag{26}$$

V. Simulation and Results

In this section, we present the results from a numerical simulation using the controllers and models presented in the previous sections. In this case, we consider two multirotors, with a mass of $m_{1,2} = 6$ kg, recovering an incoming fixed-wing UAV at $m_f = 3$ kg. The fixed-wing UAV is approaching at a constant speed of 15 m/s, and the multirotors are set to reach an approach-speed of 7 m/s. Further, the multirotors are equipped with a basic autopilot that handles attitude setpoints, as discussed in Section III-E which is implemented as a PD control structure. This is a similar structure as in the autopilot to be used for experiments, discussed later in Section VI. Next, we consider a net with a width and height of 5 and 3 m, respectively. The numerical simulation is conducted in MATLAB, using *Runge Kutta 4* as integrator at 50 Hz. The total thrust of each multirotor is configured such that it uses half of the available power at hover. Due to the construction of the multirotor, the available torque is likewise limited so that each motor does not exceed its maximum. The multirotor has a motor-to-motor diameter of 1 m. Further, discrete time sampling is implemented with a *zero-order-hold*. The net is attached 10 cm below the Centre of Gravity of each multirotor.

In Figure 9, snapshots of the dynamics during the catch is shown. We can see that the multirotors successfully intercept the incoming fixed-wing UAV, and are able to handle the load during impact. The tension force on the left multirotor can be seen in Figure 10, where the oscillations of the load can be clearly seen. The steady value in the z axis conform to half the weight of combined net and fixed-wing UAV. Due to a slight twist in the net when it swings, a slight transient can be seen on the y-component of the tension force. Further, in Figure 11, the along-track velocity of the multirotors are shown. We can see that right after impact, some residual oscillations remains due to the swinging payload.

VI. Systems Architecture

This section gives an overview of the architecture of the experimental system, including the different vehicles used, autopilots and communication links. A detailed description of the base system architecture can be seen in [24].

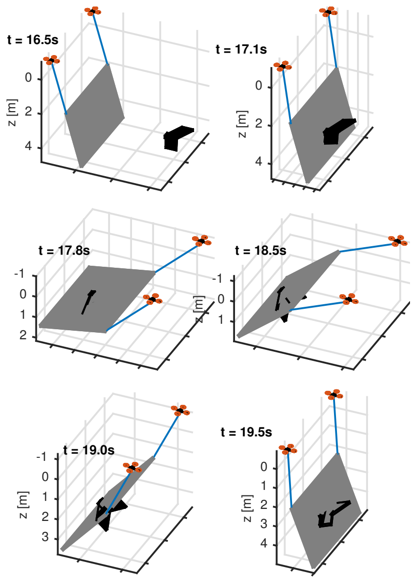


Fig. 9. Snapshots of a simulation run at right before and after the fixed-wing UAV is intercepted by the suspended net.

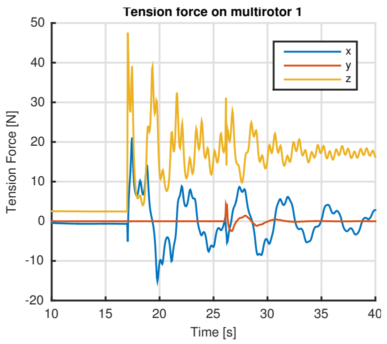


Fig. 10. The tension force on the wire connected to multirotor 1, in all three dimensions. The stationary value of z corresponds to half the total weight of the net and the arrested fixed-wing UAV.

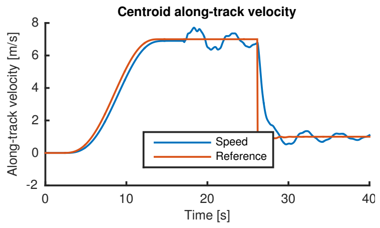


Fig. 11. Resulting velocity along the virtual runway.

TABLE I
SKYWALKER X8 DATA

Airframe weight	3 kg
Maximum takeoff weight	4.2 kg
Cruise speed	18 m/s
Flight time	45 - 60 minutes
Control Surfaces	Elevons (combined aileron and elevator)
Autpilot	Pixhawk w/ ArduPlane

TABLE II
DJI S1000+ DATA

Airframe weight	4 kg
Maximum takeoff weight	11 kg
Flight time	15-25 minutes
Autopilot	Pixhawk w/ ArduCopter

A. Airframes

In this research, we utilize the *Skywalker X8* as our fixed-wing UAV. This is a light-weight low-cost flying-wing type UAV made out of styrofoam, making it an ideal platform for research purposes. The main characteristics are summarized in Table I, and seen in Figure 12. For a multirotor platform, a versatile vehicle with plenty of lift capacity is needed. We are using the *S1000+*, made by DJI. This is a octocopter type design with eight arms and motors, as can be seen in Figure 13. The main data can be seen in Table II.

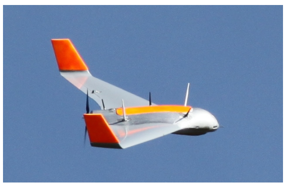


Fig. 12. The Skywalker X8 in flight.

B. Autopilot and Payload Computer

Both the fixed-wing and the two multirotors are controlled by the same basic hardware. We are using the Pixhawk [25] autopilot hardware, configured with Ardupilot [26] software

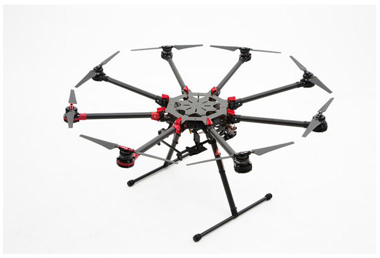


Fig. 13. DJI S1000+ frame. *Image courtesy of dji.com*

as the low-level autopilot. This setup can be configured for a multitude of different vehicle types, and handles basic navigation and control, such as sensor fusion from accelerometers and gyros, and attitude control. In addition, the vehicles are fitted with an onboard Linux Computer, running customized software. This sends commands and reference values to the autopilot. On the computer, we utilize the LSTS toolchain [27] as a framework for controller implementation. The toolchain is developed by the *Underwater Systems and Technology Laboratory* at University of Porto, Portugal, and is available open-source at Github [28]. The main component is DUNE, which is a modular software framework for autonomous control. The toolchain also contains a ground control segment (Neptus), messaging protocol (IMC), and a Linux distribution (Glued). The components communicate as illustrated in Figure 14.

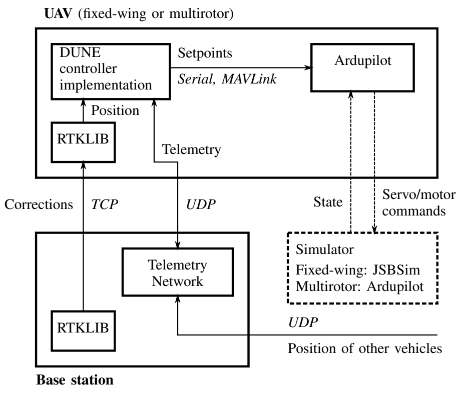


Fig. 14. Overview of the different elements involved in the UAV control system. A base station runs a server hosting corrections for RTK GNSS, and which is transmitted to each vehicle. The telemetry network is based on a 5.8 GHz ethernet/IP radio link, with *time division multiple access* as medium control. DUNE, running our control software in the vehicle on a *Beaglebone Black*, transmits setpoints to the autopilot. In the figure, *italic text* denotes a transport protocol.

C. Navigation

The autopilot internally fuses data from a MEMS-based IMU with a magnetometer and GNSS to provide a full state attitude and position reference solution using an Extended Kalman Filter. As described earlier, the position acquired from traditional GNSS is without the required accuracy to do precision landing and formation flight [13]. However, by using real-time kinematic (RTK) techniques with a differential correction, centimeter-level real-time positioning can be achieved due to the short signal wavelength (19 cm for GPS L1) of GNSS signals. This requires that integer carrier phase ambiguities are successfully resolved [13]. RTKLIB [29] is an open-source library for computing these ambiguities, and providing real-time position updates using raw data from a GNSS receiver, in our case the u-blox M8T [30]. The same receiver is used at both the base station and in the vehicles.

D. Network, Net and Custom Sensors

The vehicles communicate over a wireless 5.8 GHz network, using a radio from Ubiquity Networks. This is an ethernet/IP based radio, with *Time Division Multiple Access* medium control. This ensures constant transfer-delay in the network.

To keep the weight of the net low, but still have it strong enough to sustain multiple recovery missions, we chose a net made from *Polyethylene*. With a mask size of 12 cm, it weights only 45 $\mathrm{g/m^2}$. To have a reasonable margin for error, we chose a net size of 3 times 5 m. We will equip hooks along the wings and in the nose of the fixed-wing UAV. For safety, we have made a device to release the suspended net from the multirotors.

To measure the load of the net to the multirotors, we have constructed a device to measure the tension and angle of the attached wire. The angles are measured using two digital magnetic encoders (MTS 360 from *PIHER* [31]) in a gimbal-like structure, while the tension is measured using a light-weight load cell (LSB200 from *Futek* [32]).

VII. Software in the Loop Simulations

As a preparation for experiments, the control software is tested in a *Software In the Loop* (SIL) simulation procedure. Here, the software is the same as when running on the target vehicle, but where the dynamics and response are done by a simulator. Our control software is implemented in C++, and by using DUNE makes it straight forward to run the software in a SIL environment. Further, one of the benefits of using the open-source autopilot *ardupilot*, is that is has an option to run the autopilot code with an accompanied simulator in SIL as well. This makes all the software interfaces and communication channels identical, which makes it a good test of the implementation.

In this test, we investigate the performance of the control software with the focus on the coordination task. While the simulation presented in Section V included the complete dynamics of the suspended net, including the impact forces, there is no net in this test. We will however be able to detect that the fixed-wing UAV would have been arrested by the net given its position relative to the multirotors.

The results of the simulation can be seen in Figures 15–17. Figure 15 shows the trajectory of the fixed-wing UAV as it approaches the virtual runway. As it comes closer to the multirotors, the coordinated multirotors start the along-track velocity profile to intercept the fixed-wing UAV at the prescribed speed. Figure 16 gives a closer look at this phase. As noted above, this simulation does not contain any net dynamics. Thus, in Figure 17, we see the relative distance between the vehicles at the time the fixed-wing UAV would have hit the suspended net.

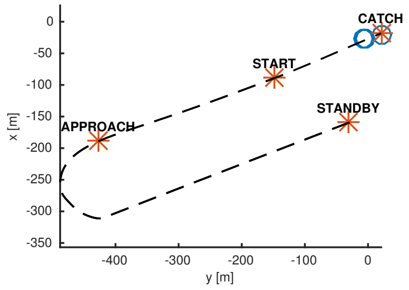


Fig. 15. An overview of the *software in the loop* simulation. The red cross marks the location of the fixed-wing UAV, while the blue circle represents the centroid location of the multirotors. The state of the supervisor at is marked at the corresponding position of the fixed-wing UAV.

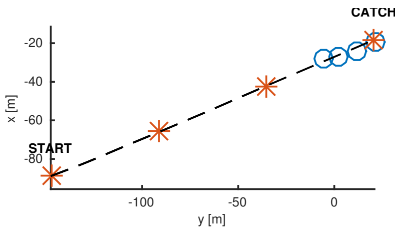


Fig. 16. A closer look at the simulation in the catch-phase. START marks the start of the multirotor along-track velocity, as to intercept the incoming fixed-wing UAV at the prescribed speed.

VIII. Conclusions

In this paper, we have presented a concept for recovery of a fixed-wing UAV in a net suspended by two multirotors. We have suggested a control design to comply with the concept. Further, numerical simulations which includes the full non-linear dynamics of the multirotors and the suspended net were conducted, which showed the feasibility of the controller. Further, we have suggested an implementation strategy, and the control system was implemented and tested in a SIL-setup which included the interface to the autopilots.

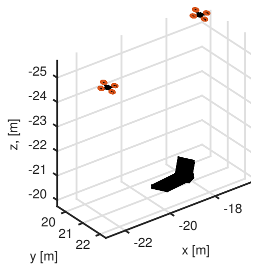


Fig. 17. A snapshot of the multirotors and fixed-wing UAV at the moment when the fixed-wing UAV would have hit the net.

References

[1] S. Pullen, P. Enge, and J. Lee, "Local-Area Differential GNSS Architectures Optimized to Support Unmanned Aerial Vehicles (UAVs)," in *Proceedings of the International Technical Meeting of The Institute of Navigation,*, San Diego, CA, 2013.

[2] S. Huh and D. H. Shim, "A Vision-Based Automatic Landing Method for Fixed-Wing UAVs," *Journal of Intelligent and Robotic Systems*, vol. 57, pp. 217–231, 2010.

[3] Insitu.com, "Insitu - ScanEagle," 2016. [Online]. Available: http://www.insitu.com/information-delivery/unmanned-systems/scaneagle

[4] Spectrum.ieee.org, "Watch This Massive Drone Launch and Recover Another Drone in Flight," 2015. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/drones/insitu-flares-drone-launch-and-recovery

[5] N. Sarigul-klijn and M. M. Sarigul-klijn, "A Novel Sea Launch and Recovery Concept for fixed wing UAVs," in *54th AIAA Aerospace Sciences Meeting*, no. 4-8 January, San Diego, California, USA, 2016, pp. 1–11.

[6] A. G. Sim, "Flight Characteristics of a Manned, Low-Speed, Controlled Deep Stall Vehicle," *NASA Technical Memorandum*, pp. 1–10, 1984.

[7] H. Taniguchi, "Analysis of deepstall landing for uav," *26Th International Congress of the Aeronautical Sciences*, pp. 1–6, 2008.

[8] S. H. Mathisen, K. Gryte, T. Johansen, and T. I. Fossen, "Non-linear Model Predictive Control for Longitudinal and Lateral Guidance of a Small Fixed-Wing UAV in Precision Deep Stall Landing," in *AIAA Guidance, Navigation, and Control Conference*, San Diego, 2016.

[9] Delftdynamics.nl, "DroneCatcher catches drone," 2015. [Online]. Available: http://www.delftdynamics.nl/index.php/en/news-en/117-dronecatcher-catches-drone

[10] Mtu.edu, "Michigan Tech Robotic Falconry," 2016. [Online]. Available: http://me.sites.mtu.edu/rastgaar/home/news/

[11] Theverge.com, "Tokyo police unveil net-wielding interceptor drone," 2015. [Online]. Available: http://www.theverge.com/2015/12/11/9891128/tokyo-interceptor-net-drone

[12] Nextgenerationvision.fr, "Drone interception," 2014. [Online]. Available: https://youtu.be/APWG3VEGbJw

[13] R. Skulstad, C. Syversen, M. Merz, N. Sokolova, T. Fossen, and T. Johansen, "Autonomous Net Recovery of Fixed-Wing UAV with Single-Frequency Carrier-Phase Differential GNSS," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 30, no. 5, pp. 18 – 27, 2015.

[14] T. I. Fossen, *Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011.

[15] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 3rd ed. Wiley New York, 2006.

[16] K. Klausen, T. I. Fossen, and T. A. Johansen, "Nonlinear Control of a Multirotor UAV with Suspended Load," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. Denver, CO: IEEE, 2015, pp. 176 – 184.

[17] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics and Automation magazine*, vol. 19, no. 3, pp. 20–32, sep 2012.

[18] H. Bai, M. Arcak, and J. T. Wen, *Cooperative control design: A systematic, passivity-based approach*. Springer-Verlag New York, 2011.

[19] J. Ginsberg, *Engineering Dynamics*. Cambridge University Press, 2008.

[20] F. Udwadia and R. Kalaba, "A new perspective on constrained motion," *Proceedings: Mathematical and Physical Sciences*, vol. 439, no. 2, pp. 407–410, 1992.

[21] M. Bisgaard, J. D. Bendtsen, and A. L. Cour-Harbo, "Modeling of Generic Slung Load System," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 2, pp. 573–585, mar 2009.

[22] K. Klausen, T. I. Fossen, T. A. Johansen, and A. P. Aguiar, "Cooperative path-following for multirotor UAVs with a suspended payload," *2015 IEEE Conference on Control Applications (CCA)*, pp. 1354–1360, 2015.

[23] F. E. Udwadia and P. Phohomsiri, "Explicit Poincaré equations of motion for general constrained systems. Part I. Analytical results," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2082, pp. 1421–1434, jun 2007.

[24] A. Zolich, T. A. Johansen, K. Cisek, and K. Klausen, "Unmanned Aerial System Architecture for Maritime Missions . Design & Hardware Description," in *IEEE RED-UAS Conference*, 2015.

[25] Pixhawk.ethz.ch, "PX4 Pixhawk," 2016. [Online]. Available: https://pixhawk.ethz.ch/

[26] Ardupilot.com, "Ardupilot - Open source autopilot," 2016. [Online]. Available: http://ardupilot.com

[27] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*, 2013.

[28] Github.com, "LSTS: Underwater Systems and Technology Laboratory," 2016. [Online]. Available: https://github.com/LSTS/

[29] T. Takasu and A. Yasuda, "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB," in *Proceedings of the International Symposium on GPS/GNSS*, Jeju, Korea, 2009.

[30] U-blox.com, "NEO/LEA-M8T — u-blox," 2016.

[31] Piher.net, "MTS-360," 2016. [Online]. Available: http://www.piher.net/

[32] Futek.com, "LSB200 S-Beam," 2016. [Online]. Available: http://www.futek.com/lsb200/overview.aspx

Bibliography

3DR (2015a). 3dr Pixhawk - 3drobotics Inc.
URL `https://store.3drobotics.com/products/3dr-pixhawk`

3DR (2015b). 3dr Radio Set - 3drobotics Inc.
URL `https://store.3drobotics.com/products/3dr-radio-set`

3DR (2015c). 3dr uBlox GPS with Compass Kit - 3drobotics Inc.
URL `https://store.3drobotics.com/products/3dr-gps-ublox-with-compass`

3DRobotics (2015). http://3drobotics.com.
URL `http://3drobotics.com/`

Ackerman, E. (2015). Watch This Massive Drone Launch and Recover Another Drone in Flight.
URL `http://spectrum.ieee.org/automaton/robotics/drones/insitu-flares-drone-launch-and-recovery`

Bai, H., Arcak, M., & Wen, J. (2011). *Cooperative Control Design: A Systematic, Passivity-Based Approach*. Springer Science & Business Media.

BeagleBoard (2015). BeagleBoard.org - black.
URL `http://beagleboard.org/black`

Beard, R. W., & McLain, T. W. (2012). *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press.

Bisgaard, M. (2008). *Modeling, Estimation, and Control of Helicopter Slung Load System*. Department of Control Engineering, Aalborg University.

Bisgaard, M., la Cour-Harbo, A., & Dimon Bendtsen, J. (2010). Adaptive control system for autonomous helicopter slung load operations. *Control Engineering Practice*, *18*(7), 800–811.

Caharija, W., Pettersen, K., Gravdahl, J., & Borhaug, E. (2012). Integral LOS guidance for horizontal path following of underactuated autonomous underwater vehicles in the presence of vertical ocean currents. In *American Control Conference (ACC), 2012*, (pp. 5427–5434). IEEE.

Chatterjee, A., & Ruina, A. (1998). A New Algebraic Rigid-Body Collision Law Based on Impulse Space Considerations. *Journal of Applied Mechanics*, *65*(4), 939–951.

Egeland, O., & Gravdahl, J. T. (2002). *Modeling and simulation for automatic control*, vol. 76. Marine Cybernetics Trondheim, Norway.

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

Futek (2015). LSB200 S-Beam Jr.
URL `https://www.futek.com/lsb200/overview.aspx`

Gryte, K. (2015). *High Angle of Attack Landing of an Unmanned Aerial Vehicle*. Master's thesis, Norwegian University of Science and Technology (NTNU).

Hespanha, J. P. (2009). *Linear Systems Theory*. Princeton, New Jersey: Princeton Press.

Hugh D. Young, & A. Freedman, R. (2011). *University Physics with Modern Physics*. San Francisco: Addison-Wesley, 13 edition ed.

Huh, S., & Shim, D. H. (2009). A Vision-Based Automatic Landing Method for Fixed-Wing UAVs. *Journal of Intelligent and Robotic Systems*, *57*(1-4), 217–231.

I.I: Argatov (2012). Mathematical modeling of linear viscoelastic impact: Application to drop impact testing of articular cartilage. *Institute of Mathematics and Physics, Aberystwyth University, Ceredigion SY23 3BZ, Wales, UK*.

Insitu.com (2016). Insitu - ScanEagle.
URL `https://insitu.com/information-delivery/unmanned-systems/scaneagle`

Ioannou, P. A., & Sun, J. (2012). *Robust Adaptive Control*. Courier Corporation.

JSBSim (2015). JSBSim Open Source Flight Dynamics Model.
URL `http://jsbsim.sourceforge.net/`

Kim, H. J., Kim, M., Lim, H., Park, C., Yoon, S., Lee, D., Choi, H., Oh, G., Park, J., & Kim, Y. (2013). Fully Autonomous Vision-Based Net-Recovery Landing System for a Fixed-Wing UAV. *IEEE/ASME Transactions on Mechatronics*, *18*(4), 1320–1333.

Klausen, K., Fossen, T., & Johansen, T. (2014). Suspended load motion control using multicopters. In *2014 22nd Mediterranean Conference of Control and Automation (MED)*, (pp. 1371–1376).

Klausen, K., Fossen, T., & Johansen, T. (2015). Nonlinear control of a multirotor UAV with suspended load. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, (pp. 176–184).

Klausen, K., Moe, J. B., van den Hoorn, J. C., Gomola, A., Fossen, T. I., & Johansen, T. A. (Submitted 2016). Coordinated Control Concept for Recovery of a Fixed-Wing UAV on a Ship using a Net Carried by Multirotor UAVs.

Lankarani, H. M., & Nikravesh, P. E. (1990). A Contact Force Model With Hysteresis Damping for Impact Analysis of Multibody Systems. *Journal of Mechanical Design*, *112*(3), 369–376.

Li, Y.-C., Zhao, Y.-P., Gui, F.-K., & Teng, B. (2006). Numerical simulation of the hydrodynamic behaviour of submerged plane nets in current. *Ocean Engineering*, *33*(17–18), 2352–2368.

LSTS (2015). The Laboratório de Sistemas e Tecnologia Subaquática-Toolchain.
URL `http://lsts.fe.up.pt/toolchain`

Mahony, R., Kumar, V., & Corke, P. (2012). Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor. *IEEE Robotics Automation Magazine*, *19*(3), 20–32.

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

Morais, F., Ramalho, T., Sinogas, P., Marques, M. M., Santos, N. P., & Lobo, V. (2015). Trajectory and guidance mode for autonomously landing an UAV on a naval platform using a vision approach. In *OCEANS 2015 - Genova*, (pp. 1–7).

Nevstad, S. O. (2016). *Autonomous landing of Fixed-Wing UAV in net suspended by Multirotor UAVs – A Fixed-Wing landing system*. Master's thesis, Norwegian University of Science and Technology (NTNU).

Phidgets (2015). PhidgetBridge.
URL `http://www.phidgets.com/products.php?category=34&product_id=1046_0`

Piher (2015). MTS-360 datasheet.
URL `http://piher.net/pdf/mts360_datasheet.pdf`

Ravn, P. (1998). A Continuous Analysis Method for Planar Multibody Systems with Joint Clearance. *Multibody System Dynamics*, *2*(1), 1–24.

Røli, J.-H. B. (2015). *Cooperative Control and RTK Navigation System for Multirotors*. Master's thesis, Norwegian University of Science and Technology (NTNU).

Sagatun, S. I., & Fossen, T. I. (1991). Lagrangian formulation of underwater vehicles' dynamics. In *, 1991 IEEE International Conference on Systems, Man, and Cybernetics, 1991. 'Decision Aiding for Complex Systems, Conference Proceedings*, (pp. 1029–1034 vol.2).

Skulstad, R., Syversen, C., Merz, M., Sokolova, N., Fossen, T., & Johansen, T. (2015). Net recovery of UAV with single-frequency RTK GPS. In *Aerospace Conference, 2015 IEEE*, (pp. 1–10). IEEE.

Skywalker Technology Co., L. (2015). Skywalker Technology Co.,Ltd.
URL `http://www.skywalker-model.com/`

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot modeling and control*, vol. 3. Wiley New York.

Swift, M. R., Fredriksson, D. W., Unrein, A., Fullerton, B., Patursson, O., & Baldwin, K. (2006). Drag force acting on biofouled net panels. *Aquacultural Engineering*, *35*(3), 292–299.

Sørbø, K. H. (2016). *Autonomous landing of fixed wing uav in a stationary net – Path and navigation system*. Master's thesis, Norwegian University of Science and Technology (NTNU).

The Society of Naval Architects and Marine Engineers (SNAME) (1950). Nomenclature for Treating the Motion of a Submerged Body Through a Fluid. *Technical and Research Bulletin No. 1-5*.

u-Blox (2016). NEO/LEA-M8T.
URL `https://www.u-blox.com/en/product/neolea-m8t`

Ubiquiti-Networks (2015). Rocket^{TM}M.
URL `https://www.ubnt.com/airmax/rocketm/`

Udwadia, F. E., & Kalaba, R. E. (1992). A New Perspective on Constrained Motion. *Proceedings: Mathematical and Physical Sciences*, *439*(1906), 407–410.