



Norwegian University of  
Science and Technology

# Advanced Mission Planner for Cooperative Underwater Vehicles

**Stephanie Buadu**

Master of Science in Engineering and ICT

Submission date: June 2018

Supervisor: Ingrid Schjøberg, IMT

Co-supervisor: Tore Mo-Bjørkelund, IMT

Norwegian University of Science and Technology  
Department of Marine Technology





## **MSC THESIS DESCRIPTION SHEET**

- Name of the candidate:** Stephanie Buadu
- Field of study:** Marine cybernetics
- Thesis title (Norwegian):** Avansert oppdragsplanlegger for samarbeidende undervannsfarkoster
- Thesis title (English):** Advanced mission planner for cooperative underwater vehicles

### **Background**

The increasing complexity of underwater missions in both military and civil applications has led to an increase in the importance of cooperative and coordinated behavior between single autonomous vehicles. It becomes beneficial to deploy multiple vehicles simultaneously to obtain fault tolerance and time, space, and functional distributions, flexibility, and adaptability that is unachievable with a common single vehicle approach.

However, with new applications and higher complexity, new questions are introduced including; how can a team of multiple vehicles be programmed to complete a set of tasks without colliding, while at the same time taking advantage of their simultaneous presence?

Mission planning and formation control in underwater environments entails many challenges. Research within formation control for autonomous vehicles is lagging behind multi-vehicle formation control in general, and most of the available research is purely theoretical.

The primary goal of this thesis is to develop an application for operating a cooperative autonomous vehicle system. The application should include mission planning capabilities and formation control for surface missions.

### **Work description**

1. Literature study on mission planners and cooperative underwater systems with emphasis on the following:
  - Basic concepts and challenges within mission planning for AUV.
  - State-of-the-art cooperating systems including AUVs.
  - Central approaches to formation control including their advantages and disadvantages.
2. Develop a formation control method for a cooperative AUV system based on existing approaches within formation control.
3. Develop an application for a cooperative AUV system with the following properties
  - Functionality for operating a cooperative system of up to three LAUVs.
  - A user interface that enables a user to define, monitor and control simple missions for cooperative systems.
  - Formation control of the cooperating system using the developed formation control method.
  - The application should work both standalone, and as an integrated part of the command and control software in the LSTS toolchain.

4. Use Neptus and application requirements to verify and validate the application.
5. Test and verify the software and formation control method in a simulated environment based on existing modules.
6. Test and verify the software and formation control method through field tests on the surface by combining LAUV Fridtjof and simulators in the Trondheim Fjord.
7. Discuss and compare the results obtained from simulation and field tests.
8. Discuss the experimental verification method of combining simulated and real vehicles.

### **Specifications**


The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulation results, experimental results, discussion, and a conclusion including a proposal for further work. The source code should be provided in the attachments with code listing enclosed in appendix. The Department of Marine Technology, NTNU, can use the results freely in its research by referring to the students work. The thesis should be submitted within June 11th, 2018.

**Start date:** January 15<sup>th</sup>, 2018

**Due date:** June 11<sup>th</sup>, 2018

**Supervisor:** Ingrid Schjølberg

**Co-advisor:** Tore Mo-Bjørkelund

  
\_\_\_\_\_  
**Ingrid Schjølberg**  
Supervisor

---

# Abstract

In later years, the complexity of underwater missions has increased, resulting in a growing interest in cooperative Autonomous Underwater Vehicle (AUV) systems. When executing a complex mission, cooperative systems are superior to a single vehicle, and mission planning and formation control are two components that contribute to this superiority. Despite the growing interest in the field, published literature on formation control of AUVs is for the most part based on theoretical research, and there is a lack of published experiments and practices.

A literature study on mission planning and cooperative underwater systems was conducted to gain knowledge about the fields. Basic concepts of mission planning and the challenges associated with mission planning for AUV were reviewed in addition to state-of-the-art applications for cooperative systems, and the formation control problem. The motivation for this thesis was to contribute to the research in the fields of cooperative AUV systems, and this was achieved by implementing a desktop application with mission planning and formation control functionality for a cooperative AUV system.

The application is named the MCS (Mission Control System) and provides an operator with a user interface for connecting to a system of Light Autonomous Underwater Vehicles (LAUVs), defining a mission plan, and monitoring and controlling mission execution. The application also functions as a framework for testing formation control methods. The MCS functions both standalone and as an integrated part of the command and control software offered in the Underwater System and Technology Laboratory (LSTS) toolchain. Verification of the software included evaluating system requirements and comparison to existing command and control software. Simulations and field tests demonstrated the application's performance and proved it to be robust in both environments. Based on observations made during verification and testing, suggestions were made to enhance the application.

An algorithmic formation control method with properties from leader-follower systems, virtual structures, and behavior-based formation control was designed. Each vehicle in the team is assigned as either master or slave, and the method further relies on the vehicles maneuvering predetermined paths simultaneously. Four cooperative strategies are in place to restore the formation if deviations occur. The method was verified through simulations and field experiments.

The experimental verification method entailed controlling a cooperative system consisting of LAUV Fridtjof in combination with LAUV simulators, and field tests were conducted on the surface in the Trondheim Fjord. Simulation and field test results proved the designed formation control method to be valid. The vehicle teams were able to complete missions and restore the formation when constraint violations occurred. Based on the observations made, suggestions for improving the method and making it more robust against external forces acting on the vehicles were given. Results imply that further testing should be carried out to validate the method's performance on a larger scale and with multiple physical vehicles. Before this can be done, collision avoidance should be implemented.



---

# Sammendrag

I de senere år har kompleksiteten til undervannsoperasjoner økt, noe som har resultert i en økende interesse for systemer av samarbeidende autonome undervannsfarkoster (*Autonomous Underwater Vehicles*, AUVer). Når det gjelder å utføre komplekse oppdrag er et samarbeidende system overlegent i forhold til en enkelt farkost. Oppdragsplanlegging og formasjonskontroll er to komponenter som bidrar til denne overlegenhet. Til tross for den økende interessen i feltet, er publisert litteratur om formasjonskontroll av AUVer for det meste basert på teoretisk forskning, og det er mangel på publiserte eksperimenter og praksis.

Et litteraturstudie om oppdragsplanlegging og samarbeidende undervannssystemer ble utført for å få kunnskap om fagfeltene. Grunnleggende begreper innen oppdragsplanlegging og utfordringene knyttet til oppdragsplanlegging for AUV ble gjennomgått i tillegg til dages løsninger for samarbeidende systemer, og formasjonskontrollproblemet. Motivasjonen for denne oppgaven var å bidra til forskningen innen samarbeidende AUV-systemer. Dette ble gjort ved å utvikle en skrivebords applikasjon med funksjonalitet for oppdragsplanlegging og formasjonskontroll for et samarbeidende AUV-system.

Applikasjonen heter MCS (Mission Control System) og tilbyr en operatør et brukergrensesnitt for å koble seg til et system av lette autonome undervannsfarkoster (*Light Autonomous Underwater Vehicles*, LAUVer), definere oppdrag, og overvåke og kontrollere oppdragsutførelsen. MCS fungerer også som et rammeverk for å teste formasjonskontroll. Applikasjonen kan kjøres som en frittstående applikasjon eller som en integrert del av kommando- og kontrollprogramvaren som tilbys av Underwater System and Technology Laboratory (LSTS) ved Universitetet i Porto. Verifisering av applikasjonen ble gjort ved evaluering av systemkrav og sammenligning med eksisterende kommando- og kontrollprogramvare. Applikasjonens ytelse ble demonstrert under simulasjoner og felttester, og den viste seg å være robust i begge miljøer. Basert på observasjoner gjort under verifisering og testing ble det gitt forslag til forbedring av applikasjonen.

En algoritmebasert formasjonskontrollmetode ble utformet med egenskaper fra leder-følger system, virtuelle strukturer og oppførsel-basert formasjonskontroll. Hver farkost i systemet defineres som mester eller slave, og videre bygger metoden på at farkostene manøvrerer forhåndsbestemte baner samtidig. Fire samarbeidsstrategier er på laget for å gjenopprette formasjonen dersom avvik oppstår. Metoden ble testet gjennom simuleringer og eksperimenter i felt.

Den eksperimentelle verifikasjonsmetoden innebar å kontrollere et samarbeidende system bestående av LAUV Fridtjof i kombinasjon med LAUV-simulatorer. Simuleringer og feltforsøk ble utført i overflaten, og resultatene viser at den konstruerte formasjonskontrollmetoden fungerer. Systemet av farkoster klarte å fullføre oppdrag og gjenopprette formasjonen når avvik oppsto. Basert på observasjoner ble det gitt forslag til forbedring av metoden for å gjøre den mer robust mot ytre krefter som virker på farkostene. Resultatene tilsier at ytterligere testing av formasjonskontrollen bør utføres for å validere metoden ytelse i en større skala og med flere fysiske farkoster. Før dette kan gjøres, bør kollisjonsunngåelse implementeres.





---

# Preface

This master thesis has been written to conclude a five year integrated masters program in Engineering and ICT, with specialization within marine cybernetics, at the Norwegian University of Science and Technology. The underlying research was partly conducted during the author's project thesis carried out in the fall of 2017. Development and remaining research was carried out during the spring semester of 2018. It is assumed that the reader of this report retains basic knowledge within engineering science.

## Acknowledgements

I would like to thank my supervisor Ingrid Schjølberg for guidance and encouragement during this work. Not to mention, for helping me make this thesis reflect my interests within both software engineering and control of underwater vehicles. To my co-supervisor Tore, thanks for great discussions, motivation when I was stuck, and assistance during field tests. I would also like to thank the Applied Underwater Robotics laboratory (AUR-lab) for letting me use LAUV Fridjof.

Finally, to my family and closest friends, thank you for all the support and motivation you have given me throughout my years of study.

*My mother always wanted a doctor and an engineer. I just picked the cooler subject.*

Stephanie Buadu  
Trondheim, June 2018



# Table of Contents

<b>MSc Thesis Description</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Sammendrag</b>	<b>v</b>
<b>Preface</b>	<b>vii</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xvi</b>
<b>Abbreviations</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.1.1 Cooperative versus Coordinated System . . . . .	2
1.2 Objectives and Scope . . . . .	2
1.3 Contributions . . . . .	3
1.4 Structure of Thesis . . . . .	3
<b>2 Literature Review</b>	<b>5</b>
2.1 UUVs . . . . .	5
2.1.1 AUVs . . . . .	5
2.1.2 ROVs . . . . .	6
2.2 Mission Planning . . . . .	6
2.2.1 Basic Concepts . . . . .	6
2.2.2 Challenges . . . . .	9
2.3 Cooperative Underwater Systems . . . . .	10
2.3.1 State-of-the-art . . . . .	10
2.3.2 Formation Control . . . . .	11
2.4 Summary . . . . .	15

---

<b>3</b>	<b>System Description</b>	<b>17</b>
3.1	LAUV . . . . .	18
3.1.1	LAUV Simulator . . . . .	19
3.2	The MCS . . . . .	19
3.3	Additional Hardware and Software . . . . .	20
3.3.1	DUNE . . . . .	20
3.3.2	Neptus . . . . .	21
3.3.3	Manta Communications Gateway . . . . .	22
3.4	Communication . . . . .	24
3.4.1	Inter-Module Communication Protocol . . . . .	24
3.5	Formation Control . . . . .	26
3.5.1	Formation . . . . .	26
3.5.2	Path Generation and Guidance . . . . .	28
3.5.3	Control Algorithm . . . . .	29
3.5.4	Verification . . . . .	32
<b>4</b>	<b>The MCS</b>	<b>35</b>
4.1	Previous Work . . . . .	35
4.2	Description . . . . .	36
4.2.1	Purpose . . . . .	36
4.2.2	Application Requirements. . . . .	36
4.2.3	Application Code . . . . .	37
4.2.4	Technology . . . . .	38
4.3	Application Components . . . . .	39
4.3.1	MCSlib . . . . .	39
4.3.2	View 1 . . . . .	42
4.3.3	View 2 . . . . .	44
4.3.4	View 3 . . . . .	47
4.4	Verification . . . . .	49
4.4.1	Evaluation of Application Requirements . . . . .	49
4.4.2	Comparison to Neptus . . . . .	51
4.4.3	Additional Requirements . . . . .	53
<b>5</b>	<b>Simulations</b>	<b>55</b>
5.1	Slave B Falls Behind Mission . . . . .	56
5.2	Master Falls Behind Mission . . . . .	59
5.3	Slave Catches up to Master Mission . . . . .	61
5.4	Master Catches up to Slave Mission . . . . .	63
5.5	Unsuccessful Mission . . . . .	65
5.6	Discussion . . . . .	67
<b>6</b>	<b>Field tests</b>	<b>71</b>
6.1	Organization . . . . .	71
6.1.1	Objectives . . . . .	71
6.1.2	Method of Approach . . . . .	72
6.2	L Mission . . . . .	73

---

---

6.3	Stop Master Fridtjof Mission . . . . .	75
6.4	U Mission . . . . .	77
6.5	Discussion . . . . .	80
<b>7</b>	<b>Discussion</b>	<b>83</b>
7.1	Uncertainties . . . . .	83
7.2	The Experimental Verification Procedure . . . . .	84
7.3	Formation Control . . . . .	85
7.4	Mission Planning and the MCS . . . . .	86
<b>8</b>	<b>Conclusions and recommendations for further work</b>	<b>87</b>
8.1	Conclusions . . . . .	87
8.2	Recommendations for Further Work . . . . .	88
	<b>Bibliography</b>	<b>89</b>
<b>A</b>	<b>Attachments</b>	<b>99</b>
A.1	LogFiles . . . . .	99
A.2	Poster . . . . .	99
A.3	Source . . . . .	99
A.3.1	dune . . . . .	99
A.3.2	MCS . . . . .	100
A.3.3	MCSlib . . . . .	100
<b>B</b>	<b>Abstract Submitted to 2018 IEEE OES Autonomous Underwater Vehicle Symposium</b>	<b>101</b>
<b>C</b>	<b>IMC Message Specifications</b>	<b>105</b>
C.1	Vehicle State . . . . .	105
C.2	Estimated State . . . . .	106
C.3	Plan Control State . . . . .	107
C.4	Plan Control . . . . .	107
C.5	Plan Specification . . . . .	108
<b>D</b>	<b>Additional information about the vehicle and simulators</b>	<b>109</b>
D.1	Simulator Modifications . . . . .	109
D.2	Vehicle Specification . . . . .	110
D.3	Connecting to LAUV Fridtjof and Simulators . . . . .	111
<b>E</b>	<b>Video from the First Day of Field Testing</b>	<b>113</b>

---



# List of Tables

3.1	Team constraints in the formation control method. . . . .	30
3.2	Cooperative strategies for handling TC1 violations. . . . .	30
4.1	Application requirements. . . . .	37
4.2	Evaluation of application requirements. . . . .	50
4.3	Additional requirements for the MCS comprised after verification and testing. . . . .	53
5.1	Overview of vehicle team and formation in Slave B falls behind mission. . . . .	56
5.2	Mission progress during execution of Slave B falls behind mission. . . . .	57
5.3	Overview of vehicle team and formation in Master falls behind mission. . . . .	59
5.4	Overview of vehicle team and formation in Slave catches up to master mission. . . . .	61
5.5	Mission progress during execution of Slave catches up to master mission. . . . .	61
5.6	Overview of vehicle team and formation in Master catches up to slave mission. . . . .	63
5.7	Mission progress during execution of Master catches up to slave mission. . . . .	64
6.1	Overview of vehicle team and formation in L mission. . . . .	73
6.2	Mission progress during execution of Stop master Fridtjof mission. . . . .	75
6.3	Overview of vehicle team and formation in U mission. . . . .	77
6.4	Mission progress during execution of U mission. . . . .	78
D.1	Overview over required duplications and modifications to enable several LAUV simulator on the same PC. . . . .	109
D.2	Configuration of LAUV Fridtjof during field tests. . . . .	110
D.3	Overview of server addresses and ports for LAUV Fridtjof and simulators. . . . .	111

---



# List of Figures

2.1	Visualization of centralized and decentralized architectures. . . . .	13
3.1	System overview including software and hardware components and communication links. . . . .	17
3.2	LAUV Fridtjof ready for deployment in the Trondheim Fjord. . . . .	18
3.3	Command for starting an LAUV simulator in the terminal. . . . .	19
3.4	Screenshot of Neptus' LAUV console during an active mission with three simulators. . . . .	22
3.5	The Manta prepared for mission execution. . . . .	23
3.6	The Manta up close. . . . .	23
3.7	The use of IMC messages between the MCS and LAUV, and within the LAUV system. . . . .	25
3.8	Illustration of displacements, distance, and constraint for a two-vehicle team in formation. . . . .	27
4.1	Class diagram depicting all public functions and variables in the LAUV class. . . . .	40
4.2	Screenshot of view 1. . . . .	42
4.3	Select vehicle module for a two-vehicle team showing dropdown menu for slave A. . . . .	43
4.4	Warning dialog prompted when the MCS is unable to connect to a vehicle. . . . .	43
4.5	Warning dialog prompted if the mission details are attempted confirmed before connections to the vehicles are established. . . . .	44
4.6	Screenshot of view 2. . . . .	45
4.7	Path modification when a waypoint is deleted from the mission plan. . . . .	46
4.8	Notification dialogs prompted after a path collision check. . . . .	46
4.9	Screenshot of view 3. . . . .	48
4.10	Vehicle overview for a single vehicle depicting a connected vehicle in a ready state. . . . .	48
4.11	Warning dialog prompted when the connection to a vehicle is lost. . . . .	48
4.12	Verification of marker placement in the MCS by comparison to Neptus. . . . .	51
4.13	Verification of mission details for a mission defined in the MCS by comparison to Neptus. . . . .	52
4.14	Visualization of the implemented and suggested maximum deviation functionality. . . . .	54

---

5.1	Vehicle trajectories in Slave B falls behind mission. . . . .	56
5.2	Speed and distance data from Slave B falls behind mission. . . . .	58
5.3	Speed, distance, and progress data from Master falls behind mission. . . . .	60
5.4	Vehicle trajectories in Slave catches up to master mission. . . . .	62
5.5	Speed and distance data from Slave catches up to master mission. . . . .	62
5.6	Vehicle trajectories in Master catches up to slave mission. . . . .	64
5.7	Speed and distance data from Master catches up to slave mission. . . . .	64
5.8	Speed, distance, and progress data from an unsuccessful mission. . . . .	66
6.1	Vehicle trajectories in L Mission. . . . .	74
6.2	Distance and progress data from L mission. . . . .	74
6.3	Vehicle trajectories in Stop master Fridtjof mission. . . . .	76
6.4	Speed and distance data from Stop master Fridtjof mission. . . . .	76
6.5	Vehicle trajectories in U mission. . . . .	77
6.6	Speed and distance data from U mission. . . . .	79
C.1	The structure of a vehicle state message. . . . .	105
C.2	The structure of an estimated state message. . . . .	106
C.3	The structure of a plan control state message. . . . .	107
C.4	The structure of a plan control message. . . . .	107
C.5	The structure of a plan specification message containing two maneuvers and a transition. . . . .	108
D.1	A successful communication link between a simulator and the MCS visualized in the terminal running the simulator. . . . .	112

---

# Abbreviations

AUR-lab	=	Applied Underwater Robotics Laboratory
AUV	=	Autonomous Underwater Vehicle
CADRE	=	Cooperative Autonomy for Distributed Reconnaissance and Exploration
CCU	=	Central Control Unit
IDE	=	Integrated Development Environment
IMC	=	Inter-Module Communication
LAUV	=	Light Autonomous Underwater Vehicle
LSTS	=	Underwater System and Technology Laboratory (Laboratório de Sistemas e Tecnologia Subaquática)
MCS	=	Mission Control System
ML	=	Mission Language
MORPH	=	Marine Robotic System of Self-Organizing, Logically Linked Physical Nodes
MRA	=	Mission Review and Analysis
ROV	=	Remotely Operated Vehicle
TCP	=	Transmission Control Protocol
UUV	=	Unmanned Underwater Vehicle
XML	=	Extensible Markup Language



# Introduction

## 1.1 Background

The increasing complexity of underwater missions, in both military and civil applications, has led to an increase in the importance of cooperative and coordinated behavior between single autonomous vehicles. By simultaneously deploying multiple vehicles, fault tolerance, flexibility, adaptability, and time, space, and functional distribution that is unachievable with a common single vehicle approach, is obtained, and new applications emerge.

However, with new applications and higher complexity, new questions are introduced such as; *how can a team of multiple vehicles be programmed to complete a set of tasks without colliding, while at the same time taking advantage of their simultaneous presence?*

The question is answered by defining a high-level mission plan that each vehicle is subject to, rather than defining individual mission plans. Defining individual mission plans may result in a messy overall mission which is difficult to maintain, and more importantly, they are error-prone (Madureira et al., 2013). Instead, a high-level mission plan is defined, broken down, and specified to compute lower-level mission plans for the individual team members.

If the mission objective allows, a way to ensure that the vehicles do not collide is to arrange them in a formation. Formation control then becomes a central component of the high-level mission plan as it ensures that the vehicles complete the mission objective while preserving the formation.

This approach to organizing cooperative systems is not only seen in subsea applications, but also within aerial vehicles and ground mobile robots. However, due to the complexity of underwater environments, research within formation control of Autonomous Underwater Vehicles (AUVs) is lagging behind. Li et al. (2014) stated that many of the existing results within formation

control of multi-AUV systems were based on pure theoretical research without sufficient experiments and practices. Now, years have passed since the survey was conducted, and although the amount of literature referring to experiments and practices in relation to formation control of AUVs has increased, the number fades in comparison to the number of literature available within comparable fields.

### **1.1.1 Cooperative versus Coordinated System**

Cooperative and coordinated are two words used about a team of vehicles working together to complete a mission, and often they are used inconsistently. In this thesis, definitions adapted from Sariel (2007) are applied.

*A cooperative system refers to a team of vehicles working in common with commonly agreed-upon goals instead of working separately in competition.*

*Coordination refers to the process by which a vehicle reasons about its local actions and the (anticipated) actions of the other vehicles to try and ensure that the team acts coherently.*

Please note that this thesis focuses on cooperative systems.

## **1.2 Objectives and Scope**

The primary goal of this thesis is to develop an application with mission planning and formation control capabilities for operating cooperative AUV systems. The motivation behind this thesis is to contribute to the research in the fields of cooperative AUV systems and formation control by documenting the design, implementation, and results of simulations and experiments.

In order to reach this goal, the following tasks should be performed.

1. Literature study on mission planners and cooperative underwater systems with emphasis on the following:
  - Basic concepts and challenges within mission planning for AUV.
  - State-of-the-art cooperating systems including AUVs.
  - Central approaches to formation control including their advantages and disadvantages.
2. Develop a formation control method for a cooperative AUV system based on existing approaches within formation control.

3. Develop an application for a cooperative AUV system with the following properties
  - Functionality for operating a cooperative system of up to three LAUVs.
  - A user interface that enables a user to define, monitor and control simple missions for cooperative systems.
  - Formation control of the cooperating system using the developed formation control method.
  - The application should work both standalone, and as an integrated part of the command and control software in the LSTS toolchain.
4. Use Neptus and application requirements to verify and validate the application.
5. Test and verify the software and formation control method in a simulated environment based on existing modules.
6. Test and verify the software and formation control method through field tests on the surface by combining LAUV Fridtjof and simulators in the Trondheim Fjord.
7. Discuss and compare the results obtained from simulation and field tests.
8. Discuss the experimental verification method of combining simulated and real vehicles.

## 1.3 Contributions

The main contributions made in this thesis is an application for operating cooperative LAUV systems that provides a framework for testing formation control and a formation control method. The application provides an operator with a user interface for planning, monitoring and controlling missions for cooperative LAUV systems, and the thesis demonstrates the applications functionality and how the application in addition to working standalone may be integrated with the command and control software in the LSTS-toolchain. The designed formation control method based on properties from leader-follower systems, behavior-based formation control, and virtual structures is defined and later verified through extensive testing. Results from simulations and field tests are presented.

## 1.4 Structure of Thesis

**Chapter 2** presents a literature study on mission planning and cooperative underwater systems. Basic concepts of mission planning and the challenges of mission planning for AUVs are dis-

cussed. Further, state-of-the-art cooperative systems are presented, before the formation control problem and three central approaches for formation control are discussed.

**Chapter 3** gives an overview of the hardware and software components required to operate the cooperative AUV system and the designed formation control methods is presented.

**Chapter 4** gives an overview of the developed application, the MCS (Mission Control System). The requirements and application modules are presented, and the chapter concludes with verification of the application.

**Chapter 5** covers the simulations conducted in this thesis. Simulation results are presented, and the chapter concludes with a discussion of the results.

**Chapter 6** covers the field tests conducted in this thesis. The organization of the field tests and the results are presented before the chapter concludes with a discussion of the results.

**Chapter 7** analyses and discusses the overall results obtained in this thesis.

**Chapter 8** concludes the thesis and suggests further work that could improve and extend the work proposed in this thesis.

**Appendix A** presents the attachments to this thesis, and gives an overview of the code files.

**Appendix B** contains a two-page abstract submitted to the 2018 IEEE OES Autonomous Underwater Vehicle Symposium.

**Appendix C** includes the structure of the IMC messages presented in this thesis through examples of messages sent and received during testing.

**Appendix D** provides addition information bout LAUV Fridtjof and the simulators. The information includes the modifications done in the simulator files, vehicle specifications, and how to connect to the vehicles.

**Appendix E** includes a link to a video from the first day of field testing.



## Literature Review

In this chapter, findings from a literature study are presented. Parts of the study was conducted during the fall of 2017 for the author's project thesis; Buadu (2017). Section 2.1 gives an introduction to Unmanned Underwater Vehicles (UUVs) and their applications. Mission planning with the main focus on AUVs and vehicle teams are discussed in Section 2.2, and section 2.3 presents cooperative underwater systems and the formation control problem.

### 2.1 UUVs

UUVs are often divided into two categories: Remotely Operated Vehicles (ROVs) and AUVs, the main difference being a physical link to a surface vessel, i.e., an umbilical that delivers power and control commands. ROVs have an umbilical while AUVs do not (Ruud, 2016). A description of the vehicle categories and their applications is presented in the following sections.

#### 2.1.1 AUVs

AUVs often have a hydrodynamical shape which combined with the absence of an umbilical, allows the vehicles to be flexible, reach high speed and operate in complex environments. AUVs carry their power supply on board and are often underactuated, i.e., not controllable in all degrees of freedom. These properties limit the duration and geographical extension of operations and exclude operations that require high precision, i.e., control of all degrees of freedom. However, the vehicle may be equipped with a variety of sensors making it suitable for environmental monitoring, hydrography, search, and recovery (Kongsberg, 2016). Current applications for AUVs include tracking marine life (Lin et al., 2017), monitoring water quality in and around oil and gas facilities (Karimanzira et al., 2014), structural inspections (Jacobi, 2015), and countering the threat from sea mines (FFI, 2013).

## 2.1.2 ROVs

The umbilical connected to an ROV restricts the spatial movement of the vehicle and increases the drag and resistance, making the vehicle less flexible compared to an AUV. An advantage of the umbilical is that there is no restriction on the duration of operation as the vehicle, in an ideal setting, has an unlimited power supply. In addition to delivering power, the umbilical may transfer video and data signals, which creates a link between the operator on the surface and the vehicle (Rist-Christensen, 2016). ROVs may be fully actuated and equipped with sensors, lights and a variety of manipulators, making the vehicles well suited for high precision and heavy operations including dynamic positioning operations. Current applications for ROVs include sea bottom and pipeline surveys, cable maintenance as well as installation, maintenance and mooring of subsea structures and equipment (Henriksen, 2014).

## 2.2 Mission Planning

A definition of a mission and mission plan is given by Kothari et al. (2012):

*”A mission for a fleet of vehicles is a set of objectives to be achieved. A mission plan is a schema for achieving the objectives of the corresponding mission.”*

Mission objectives in marine applications can be divided into five groups; data transportation and communication, mapping, tracking, monitoring and search (Kothari et al., 2012). Each objective can be accomplished in several manners, and for it to be clear which is the correct one, a mission plan must be in place. The mission plan states what the vehicle should do at different stages of a mission. There are several approaches to mission planning depending on the type of mission and vehicles involved. In the following section, basic concepts of mission planning and challenges associated with mission planning for a single and multiple AUVs are introduced.

### 2.2.1 Basic Concepts

Vehicle primitives<sup>1</sup> are often the basis of a mission plan, and a vehicle primitive is a simple task that the vehicle can perform autonomously. If a mission is executed by multiple vehicles team constraints and cooperative strategies might be set onto the vehicles during mission execution. A mission language (ML) is used to state and communicate the mission plan, and the approach to mission planning says something about how the mission plan was generated. Vehicle primitives, team constraints and cooperative strategies, mission languages, and approaches

---

<sup>1</sup>Vehicle primitives are often referred to as vehicle commands or vehicle behavior.

to mission planning are four central concepts of mission planning that were studied, and this section discusses each of these concepts.

### **Vehicle primitives**

Vehicle primitives for UUVs range from basic sensor enabling to complex behavior as navigation towards a 3D waypoint. Which primitives a vehicle provides depends on its autonomous abilities, meaning that also ROVs may have primitives allowing them to operate at a semi-autonomous level. Kothari et al. (2012) presents four categories of primitives essential to mission planning: motion, communication, sensor, and payload primitives.

Motion primitives are called maneuvers and include dynamic controllers and control strategies. Path following, trajectory tracking and go to waypoint are three examples of motion primitives. There are also motion primitives for vehicle teams, and these are referred to as team maneuvers, where the team, as a whole, maneuvers autonomously. Communication primitives entail the vehicle being able to broadcast, accept, and decode messages. The sensor and payload primitives turn on and off hardware and set the sensor parameters.

### **Team constraints and cooperative strategies**

The cooperative strategy (Glotzbach et al., 2015; Lin et al., 2017) for a mission tackles the cooperative behavior which is not covered in the individual vehicle's mission plan and must ensure two things:

1. That vehicles do not collide.
2. That the vehicles' simultaneous presence is exploited.

Among other things, the cooperative strategy can contain team constraints, which are rules of how the vehicles should relate to one another. A team constraint could be that vehicle A must be within a distance  $r$  of vehicle B at all times. Given a path following maneuver, this would mean that the team follows the path subject to this constraint. Say vehicle A has a thruster malfunction and cannot keep the desired velocity. Vehicle B then has two choices; slow down in order not to violate the constraint or continue at the assigned velocity. Which choice is the right one is defined in the cooperative strategy and varies depending on the nature of the mission. A team constraint can apply to the team as a whole or only parts of the team.

### **Mission languages**

To communicate the mission plan and cooperative strategies between the mission control system and one or more vehicles the mission plan must be written down or specified in some way. The

solution is to use an ML to describe the sequence of tasks to undertake when carrying out the mission. Some languages introduce verification capabilities, others introduce planning skills, and the right choice when picking an ML for a system will vary (Palomeras et al., 2012). No single ML fits all types of missions, and often one will see that different actors have tailored MLs for their vehicles and missions. In the following section, some examples of MLs are presented.

A mission can be specified in a regular language<sup>2</sup> by abstracting the mission elements. The benefits of regular languages as MLs are discussed in McMahon and Plaku (2016). When using a regular language, the risk of error on a human operator's part is significantly reduced due to the rules imposed by the regular language. There are three additional properties making regular languages suitable as MLs. The first being that regular languages have logical operators which enable definition of hierarchical tasks. Secondly, regular expressions may be expressed via several computational models, and finally, the variety of regular languages and the possibility of converting from one language to another gives the user an increased when defining a mission.

Using a human-readable and writable ML makes it easier for the operator to understand the mission plan. Eckstein et al. (2013) evaluated Python<sup>3</sup> as an ML, and concluded that Python was not suited as an ML. The most significant shortcoming of Python was the complexity when checking the mission plan. A check to identify deadlocks must be run whenever a mission plan is received or updated. Python as an ML introduces the cumbersome process of checking every "if" and "switch" statement to verify all possible outcomes.

Fortunately, there are other options when a human readable ML is to prefer. Description languages, as Extensible Markup Language (XML)<sup>4</sup> are good candidates. The extensive use of XML is an advantage because it means there are parsers available in most programming languages for the mission handler to translate the mission plan. However, there is an overhead with XML when wanting to update a mission, which has to be handled, despite this, Eckstein et al. (2013) deemed that XML is a suitable ML.

Another option could be to use a high-level ML and compile the plan into a Petri Net<sup>5</sup>. This process is described in detail and discussed in (Palomeras et al., 2008).

---

<sup>2</sup>A regular language is a formal language used in formal language theory and theoretical computer science. For a more detailed description of regular languages see <https://goo.gl/f4SLQY>

<sup>3</sup>Python is a widely used high-level programming language

<sup>4</sup>XML is a markup language used to describe data. The format of the XML file is quite flexible and is both human-readable and machine-readable.

<sup>5</sup>A Petri net is a mathematical modeling language. See <https://goo.gl/Hmvrpg> for a detailed description.

### **Approaches to generating a mission plan**

Kothari et al. (2012) define three ways that a mission plan for specified objectives and a fixed set of maneuvers can be generated:

1. Directly by a human operator.
2. By solving a control problem on a fixed discrete/hybrid graph containing the set of possible environment and vehicle actions.
3. By using constraint problem-solving algorithms.

Which approach is the most suitable will vary with the mission objective, computational power available, and environmental conditions. This thesis gives an example of a mission plan generated by a human operator, and Bloem et al. (2012) and Py et al. (2010) presents two mission planners based on the second and third approach, respectively.

#### **2.2.2 Challenges**

There are several challenges associated with mission planning for AUVs, and the number of issues grows when several vehicles are added to the mix. A common denominator for many of the challenges is unreliable communication. On the surface, some vehicles use cellular and satellite communication, but this has high latency and limits the size of the data being transferred. When subsea, not all types of signals can be conducted, e.g., GPS signals, limiting the communication abilities even further. Acoustic communication is utilized for passing messages subsea, but this is prone to interference, disruption and unpredictable delays (Madureira et al., 2013).

As a consequence of the unreliable and restricted communication, missions with AUVs are prone to lacking reliable position measurements subsea, spatial restrictions between vehicles to preserve communication links, and challenges in how to handle large volumes of data. Also, mission planners and cooperative algorithms, in these types of missions, must be robust for communication faults.

Changing dynamics in mission environments pose a challenge if a vehicle's dynamics are altered as a result. McMahon and Plaku (2016) discuss how not considering this possibility in the initial planning phase may render certain tasks infeasible if a vehicle's dynamics are changed. In the case of unknown or uncertain environment dynamics, a solution could be to estimate the next state of the environment based on the current state and use this to conclude on the impact on the vehicle's dynamics in-situ. However, this approach is not without its challenges, and the following discussion is made in MahmoudZadeh et al. (2016):

”In large-scale operations, accurate estimation of the next state of the operating field (e.g., obstacles’ behavior, etc.), far beyond a vehicle’s current position and sensor coverage, is computationally hard and experimentally impractical.”

Obstacle avoidance is essential during mission execution as the mission environment may have both stationary and moving obstacles. The challenge is deciding where in the architecture the obstacle avoidance should be placed. It can be integrated into the maneuvers or placed in the mission planner. Some obstacle avoidance is necessary for the maneuvers, but placing too much obstacle avoidance here would mean increasing the level of control in the maneuvers. Placing the majority of the obstacle avoidance in the mission planner could lead to an overly complicated plan. Examples of how this has been solved are presented in Kothari et al. (2012).

## 2.3 Cooperative Underwater Systems

The introduction to this thesis sheds light on the need for cooperative systems to conduct the complex applications that are subsea, and which benefits deploying multiple vehicles rather than a single vehicle entails. Research on the problem of cooperative and coordinated multi-vehicle systems began in the nineteen-eighties, and the interest in the field has grown as the multi-agent technology and complexity and diversity of undersea tasks have evolved (Yao, 2013).

The following sections present state-of-the-art cooperating and coordinated underwater systems, successful applications, and the formation control problem.

### 2.3.1 State-of-the-art

At the beginning of the 2000’s the *Iraqi Freedom* mine countermeasure action was conducted. A task that would have taken 21 days was conducted in just 16 hours by a team of REMUS small UUVs (Yao, 2013). This action exemplifies the superiority of a system of multiple UUVs when it comes to time and space resolution.

A large part of the research in the field has been driven by the military’s interest and needs. The US Navy’s UUV Master Plan was completed in the year 2000 and detailed the need, vision, and approach for the US Navy’s development of UUVs (Fletcher, 2000). As a result, the Cooperative Autonomy for Distributed Reconnaissance and Exploration (CADRE) system was developed. The CADRE system is a framework for the coordination of heterogeneous collections of unmanned vehicles for autonomous execution of goal-oriented missions. Among the needs specified in the master plan were; communication and navigation aids, and undersea search and survey, and the CADRE system addressed both of these needs (Willcox et al., 2006).

The CADRE system was developed within the context of undersea mine countermeasures missions and consists of a network of AUVs and unmanned surface vessels that conduct undersea mine countermeasures surveys while remaining in constant contact with each other via a multi-modal communication architecture (Bibuli et al., 2015; Willcox et al., 2006).

Another approach for networked vehicles was proposed by the GREX project, a European research and development project, which studied coordination and control of cooperating heterogeneous unmanned systems in uncertain environments. The result of the project was a conceptual framework and middleware systems for coordinating a swarm of pre-existing heterogeneous vehicles cooperating to achieve a well defined practical goal (Kalwa, 2010). The final sea trial was conducted in 2009, where a network of two AUVs and two catamarans completed a mission using the GREX-system.

A framework with a similar purpose as the GREX system is the MORPH (Marine Robotic System of Self-Organising, Logically Linked Physical Nodes) concept (Kalwa et al., 2012). In addition to the existing abilities of the GREX framework, the vehicles executing a mission based on the MORPH concept maneuver in predefined formations. Karimanzira et al. (2014) present simulation results based on the MORPH concept where several vehicles equipped with various sensors and cameras are deployed in a formation to map an ocean structure.

Sujit and Saripalli (2013) present a theoretical approach to coordinating an AUV and Unmanned Aerial Vehicle (UAV) in the tracking of visible ocean features, e.g., oil spills. By exploiting the UAVs ability to cover large areas in a short period and the AUVs local coverage and capability to take measurements with high accuracy. The mission is planned in-situ based on the values of the AUVs measurements.

Successful applications of cooperative underwater systems in the field include 3D modeling of a water lagoon's salinity by executing four AUVs simultaneously (González et al., 2012) and tracking fish using a team of homogeneous AUVs to improve the estimated position of the target and pursuing it (Lin et al., 2017).

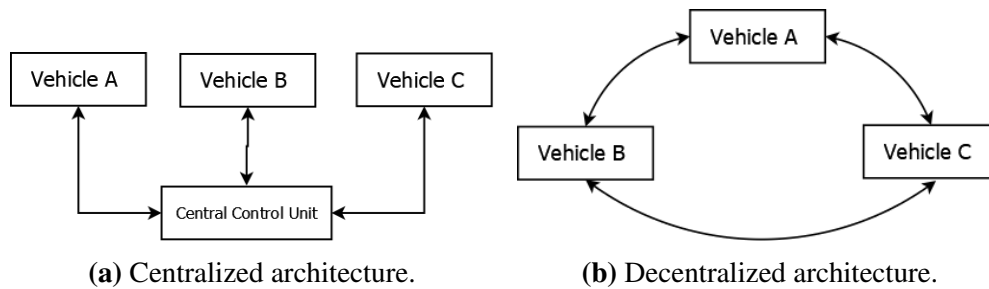
### 2.3.2 Formation Control

Formation control describes the task of controlling a group of robots to complete a mission and at the same time keep a desired shape. It is considered a fundamental problem in the field of cooperating systems in the air, on the ground, and at sea (Cui et al., 2010; Essaouari and Turetta, 2016). A solution to the problem should manage the robots in the team to ensure that they all are spaced appropriately and do not collide with their neighbors. Compared with aerial vehicles and ground mobile robots, studies on multi-AUV formation control are lagging behind.

However, as some of the approaches to formation control on the ground and in the air can be modified for subsea applications, it is a natural starting point for a study on formation control of AUVs (Li et al., 2014). Existing methods within formation control can be divided into two main categories; algorithmic and analytic. Methods that fall into the analytic category can be verified using mathematical tools, while algorithmic methods are studied through simulation. There are three central approaches to formation control; behavior-based formation control, virtual structures and leader-follower systems. The first one falls into the algorithmic category, and the remaining are defined as analytic methods (Breivik et al., 2008).

In addition to an approach for formation control, an architecture which decides where the formation control is placed in the system architecture must be in place. Two architecture types are commonly used in formation control; centralized and decentralized. A description of both based on Essaouari and Turetta (2016) and Pantelimon et al. (2018) follows. In a centralized architecture (illustrated in Figure 2.1a), decisions are made by a Central Control Unit (CCU) and communicated to the individual vehicles to be executed. The CCU can communicate with each vehicle and exercise control over it. Advantages of this architecture are that a vehicle does not need to be aware of the other agents in the network, control algorithms in this architecture can often be simplified, and relieving the individual agent of control increases their capacity for sensory infrastructure and payload. A disadvantage is that a failure in the CCU disables the whole system. A limitation of the centralized architecture is that the maximum number of agents is decided by the processing and communication capabilities of the CCU. Also, the spatial extension of the mission is limited by the communication range of the CCU. In a decentralized architecture, the control is placed with the individual agents and the agent share information as illustrated in Figure 2.1b. All decisions are made based on local information, i.e., feedback from the vehicle and its neighbors. By decentralizing the control system, fault tolerance is introduced into the system, and a fault in a single vehicle does not disable the whole system, only the vehicle in question. Another advantage of the decentralized architecture is that the number of vehicles is widely scalable. What limits the architecture is the communication capabilities of the whole network.





**Figure 2.1:** Visualization of centralized and decentralized architectures. The arrows represent communication links.

### Behavior-based formation control

Behavior-based methods are often implemented in a decentralized architecture (Cosic et al., 2013), and the idea is to prescribe a set of desired behaviors for each vehicle based on different factors during mission execution. Behavior in this context is synonym with vehicle primitive, and basic behaviors include formation maintenance, obstacle avoidance, collision avoidance, and goal seeking. The basic behaviors are given a relative priority, and a resulting control action for the vehicle is obtained as a weighted average of the desired behaviors in each iteration (Cosic et al., 2013; Beard et al., 2001; Li et al., 2014). Formation behavior is naturally found in nature, like flocking of birds and schooling of fish, and early research in the field of behavior-based control was in the context of simulating flocks of birds for computer graphics (Balch and Arkin, 1998). Since then, several approaches to formation control using the behavioral approach have been developed. A brief review of a few of these follows.

In McColgan and McGookin (2016), the schooling behavior of fish is used to coordinate multiple biomimetic AUVs <sup>6</sup>. In Jia and Li (2007), a behavioral approach is combined with potential functions to control multiple AUVs in a predefined formation while avoiding obstacles. The application of the behavioral approach in a mission planner for target searching and recognition in formation with AUVs is described in Sorbi et al. (2012).

Behavior-based methods inherit advantages such as explicit feedback on control actions and fault tolerance from the decentralized architecture. Another advantage is that it is natural to derive a control strategy despite competing objectives. Disadvantages include the lack of a way to explicitly define group behavior and the difficulty of mathematically guaranteeing the stability of a solution. When competing objectives are combined, occasionally strange and unpredictable things might occur. Given these realities, behavior-based approaches are often associated with other formation control methods. (Beard et al., 2001; Li et al., 2014)

<sup>6</sup> A biomimetic AUV is an AUV that employs similar propulsion and steering principles as real fish.

### **Virtual structures**

Another solution to the formation control problem is to organize the vehicle team as a virtual structure. A virtual structure consists of a collection of elements, e.g., AUVs that make up nodes in a rigid virtual structure. The elements maintain their shape through constraints preserving the geometric relationship between the individual elements and to a frame of reference (Lewis and Tan, 1997; Cosic et al., 2013; Essaouari and Turetta, 2016). The method entails three repeating phases. First, the desired dynamics of the virtual structure is established, then the dynamics are translated to the individual vehicles, and finally, a control action is calculated for the individual vehicles (Beard et al., 2001; Lewis and Tan, 1997). The virtual structure approach can be structured in both a centralized and decentralized architectures (Pantelimon et al., 2018).

There are many examples of virtual structures in formation control, but few related to AUVs and only in two dimensions. Cooperating AUVs in formation observing and predicting ocean processes are reported in Fiorelli et al. (2006). Reports of virtual structures being combined with other methods for formation control, such as potential fields (Pan et al., 2017; McIntyre et al., 2016) and leader-follower methods (Yan et al., 2016; Rout and Subudhi, 2016; Zhang et al., 2016) are more frequent.

Advantages and disadvantages of the architecture will be inherited by the virtual structure method. Other advantages include that feedback to the virtual structure is naturally defined and that it is fairly easy to prescribe a coordinated behavior for a group. A disadvantage is that the rigid constraints of a virtual structure may limit the potential applications for the vehicle team. (Beard et al., 2001)

### **Leader-follower**

In leader-follower formation control, a vehicle in the group is designated leader, and the vehicles designated as followers track the leader's movement in a given geometric formation or with a time offset (Cosic et al., 2013; Beard et al., 2001). Variations of leader-follower methods include multiple leaders in a team, chaining the vehicles, i.e., each vehicle follows the vehicles in front and leads the vehicle behind, and many more. Leader-follower methods can be organized in both a centralized and decentralized architecture (Pantelimon et al., 2018).

Systems using leader-follower methods involving AUVs are described in Yao (2013), where the vehicle with the highest accuracy navigation system in a heterogeneous team is set as the leader and the rest as followers; in Lapierre et al. (2003), where a team of two AUVs organized as leader and follower maneuver parallel paths and the follower adapts its speed based on the leader; in Paliotta et al. (2015), where a constant bearing controller is developed, and the followers successfully track the leader in simulation using the leaders velocity data; and

in Soares et al. (2013), where a three-vehicle team of AUVs has two designated leaders and a single follower.

Comparing virtual structures and leader-follower methods, a clear advantage of the latter is the possibility of time-varying formations, allowing for maneuvering in complex environments (Cosic et al., 2013). Other advantages include that it is relatively straightforward to implement and understand, easy communication structures due to only one way broadcast being required, and scalability (this is conditioned of a decentralized architecture). Disadvantages of a leader-follower method are that a failure in the leader means the mission fails and that there is no explicit feedback from the followers to the leader. (Pantelimon et al., 2018; Li et al., 2014)

## 2.4 Summary

The following section outlines which of the concepts and approaches discussed in the literature review have been used in the implementation of the MCS and formation control method. In the case of several options, an argument is made to support the choice.

To avoid developing a translator for translating the mission plan generated by the mission planner, the mission language implemented on board the vehicle has been used in the MCS. Inter-Module Communication (IMC) messages are XML based, and a closer presentation of IMC messages and the communication protocol is given in Section 3.4.

The mission plan is generated directly by a human operator. This approach was selected due to the low complexity of the missions and the limited amount of maneuvers and constraints required to reach the goal. The author deemed all three approaches to generating a mission plan satisfactory, but this as the most straightforward to implement.

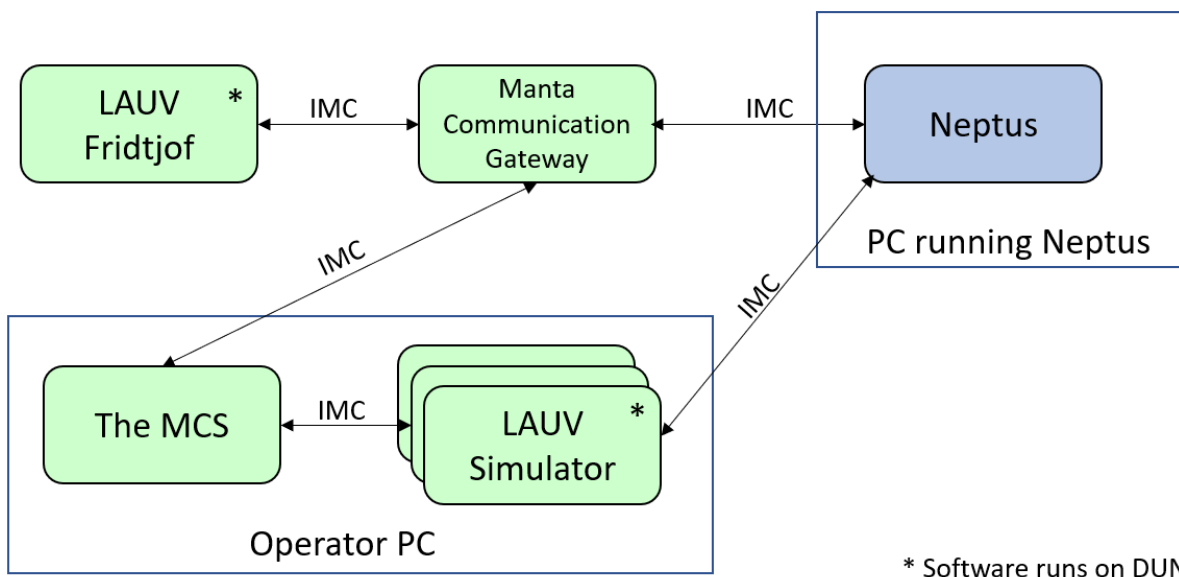
A centralized architecture was chosen first and foremost for its decoupled properties. The architecture makes it trivial to exchange a simulated vehicle for a real one without affecting the performance of the remaining team members. Also, the vehicles' payload and sensory capacities are preserved to allow for later expansions of the mission planner.

Finally, the formation control method was designed based on various properties from both algorithmic and analytic methods, with the goal of creating a method that would work based on the vehicle, field conditions, and time at the author's disposal. The behavioral properties were deciding, causing the formation control to be categorized as an algorithmic method.



## System Description

This chapter is dedicated to presenting the software and hardware in the system used to operate the cooperative AUVs. The central components and their purpose in the greater system are discussed. Figure 3.1 gives an overview of the system components. Green components are essential for the system functionality, while the blue may be removed without loss of system performance. The arrows represent communication links.



**Figure 3.1:** System overview including software and hardware components, and communication links. The arrows represent the communication links, and the blue component may be excluded without loss of system functionality, while the green components may not.

## 3.1 LAUV

The vehicle used for testing the MCS and formation control is Fridtjof, an LAUV owned by NTNU. The LAUV is developed by the Underwater System and Technology Laboratory (LSTS) at Porto University in cooperation with OceanScan-MST. The LAUV was developed for security and surveillance, oceanography, and hydrography purposes, and is easily launched, operated, and recovered with minimal operational setup (Madureira et al., 2013). The vehicle is operated with software developed by LSTS; DUNE for on board operations and IMC messages for communication, and a presentation of these is given in Section 3.3. The vehicle is operated with LAUV Remote<sup>1</sup> close to shore, and a desktop command and control software like Neptus (Section 3.3.2) or the MCS (Section 3.2) once it is a safe distance from shore. Figure 3.2 shows the vehicle ready for deployment.



**Figure 3.2:** LAUV Fridtjof ready for deployment in the Trondheim Fjord.

LAUV Fridtjof supports twelve motion primitives, but the system only requires one; the *go to* maneuver. The motion primitive maneuvers the vehicle to a waypoint defined by latitude, longitude, and depth or altitude, at a specified speed. The desired attitude at the destination can also be defined, but this property is not enabled by the system.

There is little documentation on how the maneuver is implemented. However, based on the author's observations and code review it is assumed that the *go to* maneuver generates a straight path between the vehicles current position and the desired waypoint. The vehicle then uses line-of-sight guidance to maneuver onto the path with the right heading and continues straight forward.

The vehicle uses GPS for positioning on the surface, the propeller's revolutions per minute

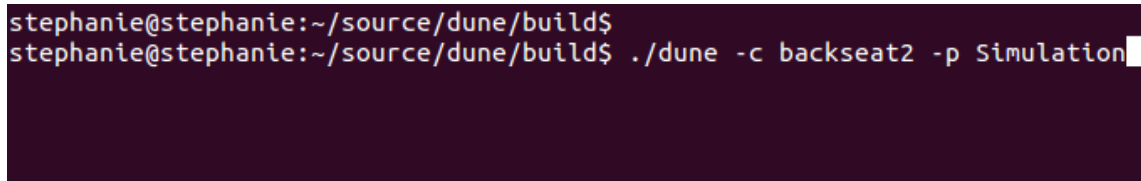
---

<sup>1</sup>LAUV Remote is an application developed by OceanScan-MST for steering an LAUV from a smartphone. The application is free and available for download in Google Play.

to estimate its velocity, and AHRS (Attitude and Heading Reference System) to measure its attitude. The communication system on LAUV Fridtjof consists of a Wi-Fi system, a global system for mobile communication, and an acoustic modem. The vehicle was on the surface for the duration of the field tests in this thesis, and only the Wi-Fi system was used. The vehicle acts as a wireless client and interfaces with other clients via a router, such as the Manta Gateway (Section 3.3.3). In optimal conditions, the LAUVs Wi-Fi range may exceed 1000 meters. (OceanScan-MST, 2016).

### 3.1.1 LAUV Simulator

The LAUV simulator runs on DUNE and is also developed by LSTS. It is open source, and available for download in the DUNE repository on GitHub<sup>2</sup>. An LAUV simulator is accessed in the *build* folder in the DUNE directory and started using the command shown in Figure 3.3. All simulations in this thesis have been performed with this simulator. To enable simulation of several vehicles simultaneously the simulator files were duplicated and renamed, and unique variables were modified. An overview of the modifications is given in Appendix D.1.



```
stephanie@stephanie:~/source/dune/build$  
stephanie@stephanie:~/source/dune/build$ ./dune -c backseat2 -p Simulation
```

**Figure 3.3:** Command for starting an LAUV simulator in the terminal.

## 3.2 The MCS

The MCS is an application for mission planning, control, and monitoring for a fleet of up to three LAUVs. The application may support other vehicles running on DUNE, such as autonomous surface vehicles, but this has not been tested. Through the user interface an operator may generate a mission and send it to the vehicles. The MCS handles formation control and allows the operator to monitor and intervene during mission execution. The application runs on Linux and is written in C++. A detailed description of the MCS' user interface, functionality and implementation is given in Chapter 4.

---

<sup>2</sup>The DUNE repository can be accessed via <https://goo.gl/EkJ4jZ>.

## 3.3 Additional Hardware and Software

### 3.3.1 DUNE

DUNE (LSTS, 2017; OceanScan-MST, 2015; LSTS, 2013a) is an embedded software for unmanned vehicles with modules for control, navigation, simulation, networking, sensing, and actuation. The software is developed by LSTS, and in addition to running on AUVs, DUNE supports UAVs, ROVs, and autonomous surface vehicles. DUNE is written in C++ and compatible with many operating systems including, but not limited to, Linux, Microsoft Windows, and Mac OS X.

#### Control logic

DUNE tasks make up the control logic for any running instance of DUNE. There are seven categories of DUNE tasks; actuator, sensor, estimator, controller, monitor, supervisor, and transport tasks. These tasks handle the logic for separate parts of the vehicle and are separated into threads that communicate by passing IMC messages.

For each category, there is a set of predefined methods that can be executed. The implementation of each method is empty and should be overridden to suit the vehicle in question. For example, the implementation of a sensor task calling the method *onActivation()* should differ for an LAUV and a UAV. For this thesis, the default implementation for LAUVs has been used.

#### Configuration

There are three profiles available in DUNE; simulation, hardware, and hardware-in-the-loop. Simulation means the system is not connected to any hardware and data is produced by simulated hardware components. The hardware profile requires there to be a connection to the vehicle. Hardware-in-the-loop will have a connection to the hardware, but some, or all of the sensors and actuators will have simulated input and output.

All DUNE instances share the same code base but have different configurations. An LAUV running with the hardware profile shares a code base with a UAV in a simulation, and this versatile application of DUNE is enabled by its modularity. A DUNE instance can run any of the predefined profiles; the profile need only be specified by the user in the configuration file. When initializing a new instance of DUNE a configuration file is run. The configuration file enables and initializes or disables DUNE tasks for the vehicle and profile in question. Configuring a vehicle is very simple because all that is needed is a single file and a specification of which profile it is to run.



### 3.3.2 Neptus

Neptus (LSTS, 2013*b*) is a software in the LSTS toolchain for command and control of unmanned vehicles. Neptus supports all the phases of a typical mission life cycle; planning, execution, and review and analysis (Madureira et al., 2013). It is possible to exclude Neptus from the presented system architecture without loss of the command and control functionality demonstrated in this thesis. There are two primary interfaces for Neptus; the operator console and Neptus Mission Review and Analysis (MRA).

#### Operator console

The operator console is the Neptus application an operator would use to plan and execute a mission. The needs of an operator differ when working with a UUV or UAV, and Neptus allows for these variations by having a configuration file for each console. The configuration file states which add-ons, both custom and open-source, should be active and not in the various profiles. In Neptus, a profile specifies the layout of a console and each console can have several profiles. Which profile to run is specified during configuration.

Figure 3.4 shows the operator console for an LAUV during mission execution for a team of three LAUVs. It is not possible to say whether or not the vehicles are simulated or real by looking at the interface. Neptus does not have functionality for planning team missions but supports monitoring all the vehicles simultaneously and send them commands individually. The LAUV operator console has a toolbar for mission planning and map interactions to the left. Mission commands can also be generated by clicking directly on the map. To the right, there are buttons for plan control, an overview of the current plan, and a list of the previous mission sent to the vehicle.

#### Neptus MRA

Neptus MRA is an application for inspecting and analyzing mission data (Holsen, 2015). All vehicles running on DUNE create a log file during execution of a mission containing all sent and received IMC messages. Neptus MRA has functionality for loading and decompressing the logs into text files for further visualization, processing and export (Pinto et al., 2012).

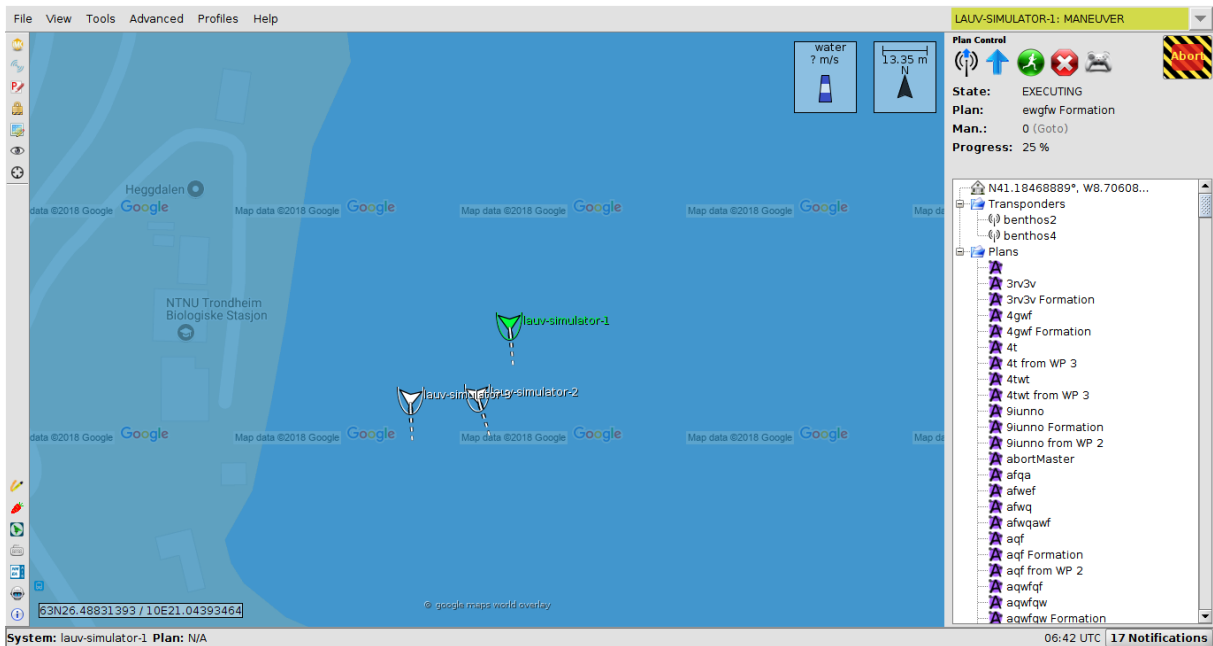


Figure 3.4: Screenshot of Neptus' LAUV console during an active mission with three simulators.

### 3.3.3 Manta Communications Gateway

The Manta Communications Gateway, commonly referred to as "the Manta", is developed by LSTS and integrates several capabilities including Wi-Fi, acoustic modem, and GPS (OceanScan-MST, 2016). Only Wi-Fi is required for the presented system to function, but the acoustic modem was enabled in case of emergency during field tests. The Manta's Wi-Fi signal has a range of up to 4.5 km and the acoustic modem a range of up to 1 km (LSTS, n.d.b). The Manta routes signals interfacing a vehicle with clients, e.g., a PC running the MCS or Neptus, or a mobile telephone running LAUV Remote. Figure 3.5 shows the Manta prepared for mission execution.

The Manta can connect to different vehicles, but only one at a time and this is set by the operator before mission execution. During a field test, one Manta would be required for each vehicle. Figure 3.6 shows the Manta close up. The information screen gives the operator information about which system the Manta is connected to and displays short messages from the system. A switch and three buttons are placed beneath the information screen. The switch to the far left is used to power the system on and off. The black button next to the switch is used to specify the vehicle. The green button pings the vehicle and the distance to the vehicle is displayed on the information screen when an answer is received. This functionality was used during field tests to ensure that the vehicle was still within the Wi-Fi-range of the Manta. The red button sends an abort message via all available channels to the vehicle which then stops all current actions and goes into standby mode.



Figure 3.5: The Manta prepared for mission execution.



Figure 3.6: The Manta up close depicting the information screen and switched and buttons available.

## 3.4 Communication

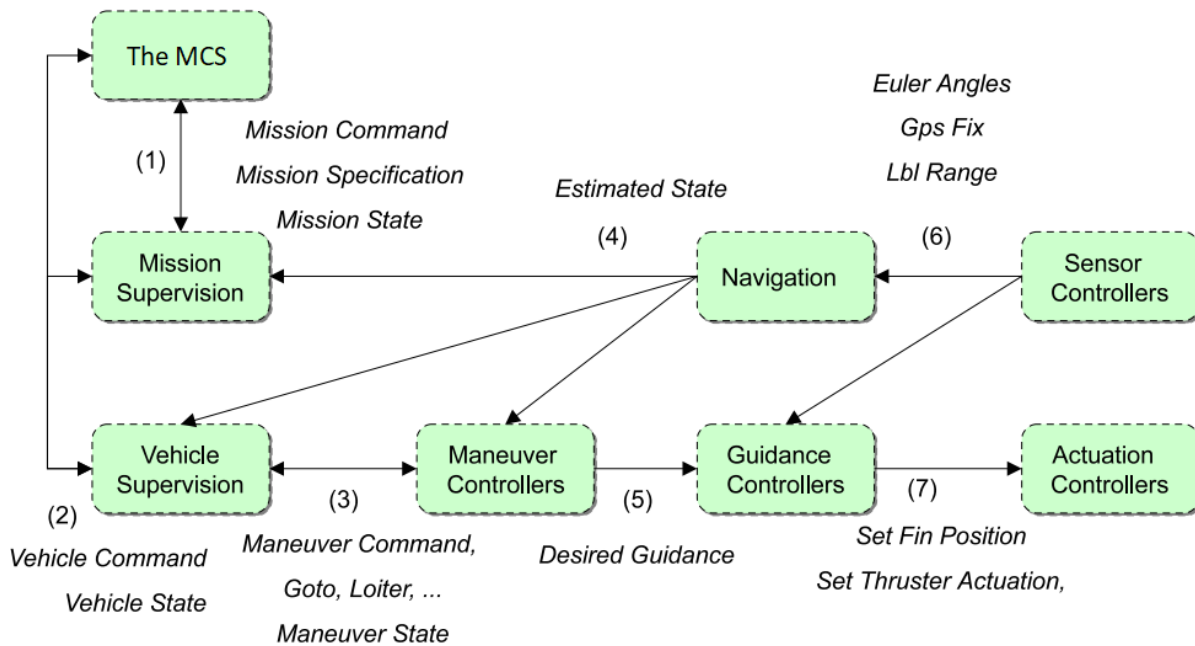
Communication in the system is enabled by IMC messages, and the Transmission Control Protocol (TCP)<sup>3</sup> is used to establish the communication links. Depending on which components are communicating with the MCS, the messages are routed via Wi-Fi or through local ports on the PC. In which manner an individual message is routed is handled by the TCP protocol, as the MCS cannot distinguish between simulated and real vehicles. The following sections describe the IMC protocol and some essential message types passed within the system.

### 3.4.1 Inter-Module Communication Protocol

The IMC protocol (Martins et al., 2009; LSTS, n.da) is an XML-based message-oriented communication protocol developed by LSTS that defines a common control message set understood by all types of vehicles and computers running DUNE. The protocol comprises seven logical message groups; (1) mission control, (2) vehicle control, (3) maneuver, (4) guidance, (5) navigation, (6) sensor and (7) actuator. Messages in groups (1) and (2) are passed between the MCS and the vehicle's units for mission and vehicle supervision. The remaining message groups are utilized when messages are passed between the various on board vehicle components. Figure 3.7 shows the components on board the LAUV and how the communication is structured between the components and the MCS.

---

<sup>3</sup>TCP is a standard protocol which defines how to establish and maintain a connection and allows for sending and receiving data over a network in a reliable manner without loss of information.



**Figure 3.7:** The use of IMC messages between the MCS and LAUV, and within the LAUV system. Adapted from Martins et al. (2009).

### IMC Messages

All IMC messages begin and end with non-empty headers and footers that have the same structure for all message packets. The data fields in the message convey relevant information about a given subject and may be in the format of an IMC message, integer, floating point number, or variable length bytes. An extensive description of the IMC protocol is published on <https://goo.gl/ANUUBw> and the essential IMC messages passed between the MCS are discussed in the following. IMC messages sent within the system demonstrating the message structures are available in Appendix C.

*Vehicle state* is a message periodically sent from the vehicle to the MCS that summarizes the overall state of the vehicle. Data fields of interest are; operation mode, current maneuver, and any field conveying information about errors.

*Estimated state* is a message periodically sent from the vehicle to the MCS containing the estimated states of the vehicle. Data fields of interest are; lat, lon, x, y, depth, and psi.

*Plan control state* is a message periodically sent from the vehicle to the MCS containing the overall state and progress of the current and last mission. If no plan is being executed, default values are used in the data fields concerning the current mission.

*Plan control* is a message sent from the MCS to the vehicle with a request regarding a plan. The request could be to *start, load, stop, or get* a mission plan. The message does not contain

the mission plan, just an identifier so it can be linked to a mission plan received in a *plan specification* message.

Plan Specification is a message sent from the MCS to the vehicle containing a mission plan. The mission plan is described by some general parameters, a list of maneuvers, and a list of transitions to be executed between the maneuvers.

## 3.5 Formation Control

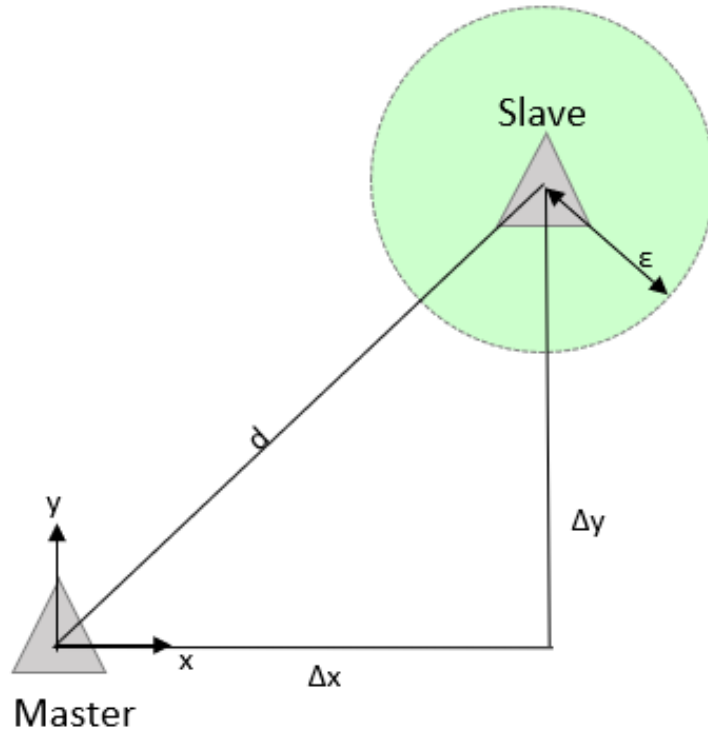
The formation control method designed in this thesis has properties from existing formation control methods and is defined as an algorithmic method. Team members are categorized as either *master* or *slave*, and only one master is assigned per vehicle team. The method is described in three parts; formation, path generation and guidance, and maneuvering, and control algorithm. All equations in this section relating WGS 84 coordinates<sup>4</sup> to a displacement in meters are based on the assumption that the earth is spherical.

### 3.5.1 Formation

The formation of the team is inspired by virtual structures, and the vehicles are placed in a coordinate system where the master's position defines origin. The position of each slave  $i$  is defined as a displacement  $\delta x_i$  on the x-axis and  $\delta y_i$  on the y-axis, from the master's position. The structure does not rotate, and the attitude of the individual vehicle is not taken into account. Figure 3.8 visualizes the displacements and constraints for a team consisting of a master and slave. A constraint (3.1) is set onto the distance  $d$  between the master and slave to preserve the geometrical relationship between the vehicles. Some deviation  $\epsilon$  is allowed before the slave is considered to violate the constraints and counteractive measures are taken.

---

<sup>4</sup>The coordinate origin of WGS 84 is meant to be located at the Earth's center of mass and is used as a standard in navigation.



**Figure 3.8:** Illustration of displacements, distance, and constraint for a two-vehicle team in formation.

$$d \in \left[ \sqrt{\Delta x^2 + \Delta y^2} - \epsilon, \sqrt{\Delta x^2 + \Delta y^2} + \epsilon \right]. \quad (3.1)$$

The vehicles' positions are defined in WGS 84 coordinates in the MCS and require conversion before the constraint can be checked. The Haversine formula (3.2), given in *Calculate distance, bearing and more between Latitude/Longitude points* (n.d.), is used to relate two latitude/longitude coordinates to a displacement in meters.

$$\begin{aligned} a &= \sin(\Delta\phi/2)^2 + \cos(\phi_S) \cdot \cos(\phi_M) \cdot \sin(\Delta\lambda/2)^2, \\ c &= 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}), \\ d &= R \cdot c, \end{aligned} \quad (3.2)$$

where  $\Delta\phi$  is difference latitude,  $\Delta\lambda$  is difference longitude,  $R$  is the earth's radius, and  $d$  is the distance between the two coordinates in meters.

### 3.5.2 Path Generation and Guidance

The master's path is generated by the operator selecting waypoints, and these being connected by straight lines. The waypoints are later displaced to generate paths with the same shape and length for the slaves. The equation for displacing the waypoints is given in *Calculate distance, bearing and more between Latitude/Longitude points* (n.d.) and stated in (3.3).

$$\begin{aligned}\phi_S &= \text{asin}(\sin \phi_M \cdot \cos \delta + \cos \phi_M \cdot \sin \delta \cdot \cos \theta), \\ \lambda_S &= \lambda_M + \text{atan2}(\sin \theta \cdot \sin \delta \cdot \cos \phi_M, \cos \delta - \sin \phi_M \cdot \sin \phi_S),\end{aligned}$$

where

$$\begin{aligned}\theta &= \text{atan2}(\Delta x, \Delta y), \\ d &= \sqrt{\Delta x^2 + \Delta y^2}, \\ \delta &= d/R,\end{aligned}\tag{3.3}$$

and  $\phi$  is latitude,  $\lambda$  is longitude,  $\Delta x$  and  $\Delta y$  are displacements in meters, and  $R$  is the earth's radius.

Before the MCS accepts the paths, they are checked for potential collisions. The paths are parameterized, and the distance between points on the parameterized lines are checked against a threshold set by the operator. The piecewise parametrization of the path is given in (3.4), where A, B are two waypoints. The distance  $d$  between points on the path is given by (3.2) and checked to satisfy  $d > d_{min}$ .  $d_{min}$  is set by the operator and is the minimum distance allowed between two vehicles in the team.

$$\begin{aligned}\phi(t) &= \phi_A + \overrightarrow{AB}_\phi \cdot t, \\ \lambda(t) &= \lambda_A + \overrightarrow{AB}_\lambda \cdot t,\end{aligned}\quad t \in [0, 1], \quad \overrightarrow{AB} = [\phi_B - \phi_A, \lambda_B - \lambda_A],\tag{3.4}$$

where  $\phi$  is latitude,  $\lambda$  is longitude,  $t$  is the parameterization variable.

The guidance in the formation control method stands out from other leader-follower approaches because there is no explicit synchronization of path variables or direct feedback of the master's position to the slaves during guidance. Instead, all vehicles start execution of their predetermined path at the same time, with the same desired speed set onto the speed controller, and no external actions are taken unless team constraints are violated. The team constraint are presented in Section 3.5.3.



A prerequisite for this strategy to work is that the vehicles are in formation before mission execution begins, and this is achieved by commanding the vehicles to the first waypoint before mission execution. To ensure that the vehicles have the right heading before starting the mission their path includes a waypoint  $X$  before reaching the first waypoint.  $X$  is generated based on the two first waypoints on the path  $A$  and  $B$ . Starting in  $A$ ,  $X$  extends the line between  $A$  and  $B$  by 20 meters. The coordinates of  $X$  are given by (3.5).

$$\begin{aligned}\phi_X &= \phi_A - \alpha \cdot \Delta\phi, \\ \lambda_X &= \lambda_A - \alpha \cdot \Delta\lambda,\end{aligned}$$

where

$$\begin{aligned}\Delta\phi &= \phi_B - \phi_A, \\ \Delta\lambda &= \lambda_B - \lambda_A, \\ \alpha &= \frac{20}{l},\end{aligned}\tag{3.5}$$

and  $\phi$  is latitude,  $\lambda$  is longitude, and  $l$  is the straight line distance between  $A$  and  $B$ .

### 3.5.3 Control Algorithm

The formation control is placed in the MCS, resulting in a centralized control architecture. A control algorithm that checks the team constraints, and takes action if any are violated, is looped during mission execution. The algorithm falls into the category of behavior-based control, and three behaviors are available for the vehicles; *path following*, *collision avoidance*, and *formation preservation*. However, the vehicle's resulting behavior is not a weighted combination, as the regular practice, but rather a single behavior. Path following is the default behavior, and the cooperative strategies activate the two remaining behaviors.

#### Team constraints and cooperative strategy

Two team constraints have been defined in the formation control method, and they are presented in Table 3.1. TC2 is in place to ensure that the vehicles do not collide, and violation of this constraint should result in one or more vehicle switching behaviors from path following or formation preservation to collision avoidance. Cooperative strategies for handling the violations have not been implemented and is left as further work. To preserve the formation, TC1 is set onto each vehicle, and five cooperative strategies, presented in Table 3.2, are designed to handle violations of this constraint. The cooperative strategies define which vehicle should switch

behaviors in different situations. CS1 and CS4 state that the master waits, i.e., takes on formation preservation behavior, once for each slave if it falls behind, and once if the master catches up to it. A consequence of this is that a master will stop considering a slave as a team member if it has previously taken action not to violate the constraint. The master switches back to path following behavior once the constraint is satisfied or when the waiting period, currently set to 30 seconds, is over, whichever comes first. CS2 and CS3 state that a slave must always wait if the master falls behind or the slave catches up to the master. There is no waiting period, and therefore the slave must wait until the constraint is satisfied before switching to path following behavior. To avoid producing additional constraint violations, C5 ensures that all slaves satisfying TS1 wait while the master waits.

**Table 3.1:** Team constraints in the formation control method.

ID	Description
TC1	A slave must always be within a radius of $\epsilon$ from its desired position.
TC2	The distance between two vehicles must always be greater than $d_{min}$ .

**Table 3.2:** Cooperative strategies for handling TC1 violations.

ID	Situation	Action
CS1	A slave falls behind during mission execution.	Master must wait once.
CS2	Master falls behind during mission execution.	A slave must always wait.
CS3	A slave catches up to master during mission execution.	A slave must always wait.
CS4	Master catches up to a slave during mission execution.	Master must wait once.
CS5	Master waits for a slave.	All remaining slaves must always wait.

### Control loop

The control algorithm is looped five times per second for each slave and has conditions in place to catch and identify constraint violations. The conditions are given in (3.6).

$$\begin{aligned}
 & \textbf{Condition 1:} & d \notin [d_{MS} \pm \epsilon], \\
 & \textbf{Condition 1.1:} & d > (d_{MS} + \epsilon) \quad \& \quad \%_S < \%_M, \\
 & \textbf{Condition 1.2:} & d > (d_{MS} + \epsilon) \quad \& \quad \%_S > \%_M, \\
 & \textbf{Condition 1.3:} & d < (d_{MS} - \epsilon) \quad \& \quad \%_S > \%_M, \\
 & \textbf{Condition 1.4:} & d < (d_{MS} - \epsilon) \quad \& \quad \%_S < \%_M, \\
 & \textbf{Condition 2:} & d < d_{min},
 \end{aligned} \tag{3.6}$$

where  $d$  is the distance between two vehicles,  $d_{MS}$  is desired distance between a master and slave,  $\%$  is mission progress,  $d_{min}$  is minimum allowed distance between two vehicles, and  $\epsilon$  is the maximum allowed deviation for a slave. Subscripts  $M$  and  $S$  represent master and slave.

Condition 1 is true if a slave deviates more than allowed from its desired position, violating TC1. Conditions 1.1 – 1.4 are used to categorize the deviation even further and identify which cooperative strategy should be activated. Condition 1.1 is true if the slave is too far away from the master, and the slave's progress is lower than the master's, indicating that the slave is falling behind. Condition 1.2 is true if the slave is too far away from the master, but has higher mission progress, indicating that the master is falling begin. Conditions 1.3 – 1.4 cover the situations where a slave and a master are too close to each other due to lower mission progress in one of the parties. Condition 2 is true if the distance between two vehicles is less than the minimum allowed distance, violating TC2. This condition must be checked for each slave in relations to the master, but also for each slave in relation to the other slaves.

Algorithm 1 shows how the team constraints, cooperative strategies, and conditions are placed in relation to each other in a control loop for a team consisting of a single master and slave. Note that CS5 is only activated if the number of slaves in the team exceeds one. The outer if-loop must be repeated for each slave included in the team. Condition 2 is checked before condition 1 because collision avoidance has higher priority than keeping the formation.

```
while formation control active do  
  if Condition 2 / TC2 violated then  
    | //not implemented  
  else if Condition 1 / TC1 violated then  
    if Condition 1.1 then  
      | CS1, CS5  
    else if Condition 1.2 then  
      | CS3  
    else if Condition 1.3 then  
      | CS4  
    else if Condition 1.4 then  
      | CS2, CS5  
    end  
  end  
end
```

**Algorithm 1:** Formation control algorithm loop.

### 3.5.4 Verification

The vehicles switching behavior based on what happens in real time makes it difficult to prove the validity of the formation control method analytically. Therefore, simulations and field tests have been conducted to test the method, and the results are presented in Chapters 5 and 6. The logical reasoning behind why the implemented cooperative strategies should yield successful formation control is presented in this section.

First, an argument is made for the validity of the formation control method in the case where the vehicle team is exposed to equal current, wind, and wave forces. It has been established that the path generated for each vehicle has the same length and shape and that the desired speed is identical. Further assumptions are:

1. The vehicles are homogeneous with similar vehicle dynamics.
2. The motion controllers on the vehicles are equal.
3. The vehicles are in formation and have the same heading before mission execution.
4. The generated mission is feasible.

The conclusion to be drawn from this is; if the vehicle's start mission execution at the same point in time and respond equally to the forces they are exposed to, their mission progress will be identical, and the vehicles will complete the mission while staying in formation.

The second case to consider is the case of the vehicle team being exposed to non-uniform external forces causing a violation of TC1, and the following assumptions are made in addition to the assumptions stated in the previous case:

1. The mission progress sent by the vehicle is continuously updated and gives an accurate representation of the vehicle's mission progress.
2. The master's wait period is sufficient for a slave to get back into the formation in the absence of extreme external forces or a fault with the vehicle.

Noting that the conditions stated in (3.6) catch all violations of TC1 and that a cooperative strategy is in place to restore the vehicle formation the following conclusion is drawn; if a vehicle violates TC1, it is given a chance to get back into formation, if it is within the cooperative strategy, before mission execution resumes.



## The MCS

In this chapter the application developed as a part of this thesis, the MCS, is presented. The previous version of the MCS and the most significant modifications are discussed in Section 4.1, and section 4.2 gives an overall description of the system including requirements and the technology used. System components are discussed in Section 4.3 and the MCS is verified in Section 4.4.

### 4.1 Previous Work

The MCS was first developed as a web application supporting mission planning, execution, and monitoring for a single LAUV during the fall of 2017. Then, the purpose of the MCS was to create a high-level mission planner interfacing ROS, a framework for writing robot software, and the on board control system on LAUV Fridtjof; DUNE. The MCS was a proof of concept supporting ROS as a framework for integrating heterogeneous vehicle teams with different on board systems. For a more detailed description see (Buadu, 2017).

The MCS has been significantly changed for this thesis, the main difference being that the MCS is now based on the DUNE framework. When designing the new version of the MCS a high priority was to ensure that the shortcomings of the previous version were dealt with appropriately. The issue of communication failures that occurred due to waypoints in a mission plan being sent gradually during mission execution was therefore addressed. In the current version, the complete mission plan is sent to the vehicle before mission execution. The map in the current version does not require an internet connection to render, and functionality for removing a single action from a mission plan without having to replan the whole mission has been implemented. Other changes from the previous to current version include:

- The application platform being changed from web browser to desktop.

- Additional views being added.
- The implementation of support for mission planning for up to three vehicles.
- The implementation of formation control.
- The mission language being changed from ROS to IMC.
- The programming language being changed from JavaScript to C++.

The way an operator defines the mission plan, how to visualize a mission plan, and some module components are preserved from the previous versions, in addition to relevant functional and non-functional requirements. Note that none of the code from the previous version has been utilized in the application presented in this thesis. All development has taken place during the spring of 2018.

## 4.2 Description

The MCS is a desktop application for mission planning, execution, and monitoring for fleets of up to three LAUVs. The following sections give a more detailed description of the purpose of the application, functional and non-functional requirements, and the technology used to build the application.

### 4.2.1 Purpose

The project background sheds light on the challenges of applying formation control methods to underwater vehicles and the limited amount of research available. The MCS provides a framework for testing the formation control method proposed in this thesis, enriching the research within formation control for underwater vehicles. The purpose of the MCS is to provide an operator with a user interface for defining and connecting to a vehicle team, defining a simple mission, setting constraints, and monitoring and controlling ongoing missions. All this while ensuring the right balance of abstraction and level of detail to limit the risk of human errors, and allowing the operator to make informed decisions.

### 4.2.2 Application Requirements.

The requirements for the MCS were derived based on the review of the previous version of the MCS and the thesis objectives. The requirements are specified in Table 4.1, and the priority ranging from 'High' to 'Low' was determined by the author.



**Table 4.1:** Application requirements.

Requirement ID	Requirement	Priority
R1	Communication between the system and the LAUV should be enabled by DUNE and IMC messages.	High
R2	All system functionality should be available offline.	Medium
R3	A user should be informed if the communication link to a vehicle is lost.	High
R4	The system should automatically reconnect to a vehicle if the communication link is lost.	High
R5	Functionality that requires communication with a vehicle should not be displayed if the communication link is not established.	Medium
R6	The system should automatically generate paths for the team members based on a path generated by an operator.	High
R7	It should be possible to send a mission plan to the vehicle and start the execution at a later point in time.	Medium
R8	A user should be able to review the mission plan at all times.	Medium
R9	The system should enable the user to monitor an ongoing mission.	High
R10	A user should be able to abort an ongoing mission.	Medium
R11	A user should be able to replan the most recent mission.	Medium
R12	It should be possible to generate a path by marking the way-points.	High
R13	The system should resend a message until a callback is received.	Medium

### 4.2.3 Application Code

A large part of this thesis has been to design and implement the MCS. To simplify interfacing DUNE with the MCS the backend components of the application were written in C++. Keeping the MCS as a web application was considered, but linking the backend code in JavaScript proved challenging. Several programming languages were unsuccessfully tested for the front-end development before C++ proved successful. Qt Creator (discussed in Section 4.2.4) was used as the Integrated Development Environment (IDE) for building the application.

The development has mainly taken place in two files; `gui.cpp` and `mcslib.cpp`. The user interface was designed using drag-and-drop functionality in Qt Creator, and a `.ui` file was auto-generated. The code for handling the TCP-connections in the application is developed by OceanScan-MST and open source, and the code for handling incoming messages is developed by NTNU. All copied code is marked in the code files. All code is attached to this thesis, and a description of the code is given in Appendix A.3.

## 4.2.4 Technology

### C++

C++ is one of the most popular programming languages primarily utilized with system/application software, drivers, client-server applications and embedded firmware (Inc., n.d.). The language embodies both high and low-level language features allowing the programmer a large amount of control, and at the same time access to well defined high-level features.

There is a wide range of libraries available for C++, simplifying development as functions for solving defined problems are already implemented. A quick search for "C++ library" on SurgeForce, an open source software platform, yields more than 6000 hits. One of the open source libraries included in the MCS is QCustomPlot, which defines the functions used to handle the map in the MCS.

### Qt

Qt is a cross-platform application development framework for desktop, embedded and mobile. In addition to offering Qt Creator, an IDE for C++, modules such as QtGui and QtWidgets, provide graphical components for designing an application. The C++ library QCustomPlot previously mentioned is developed in Qt using among other things QtWidgets. Qt is a fully functioning ecosystem for software development and has functionality for building and deploying applications. However, applications developed in the environment can be compiled by a standard C++ compiler and deployed outside of the Qt environment as well.

Development of Qt started in 1990 by the Norwegian company Trolltech. Since then, it has been widely expanded, and currently, it is known as the Qt Project. The Qt Project is based on a community principle and contributions to the development of Qt, such as writing code, documentation for the framework, and reporting bugs, are made by the Qt Community consisting of both individuals and companies. (Community, n.d.)

### QCustomPlot

QCustomPlot is a Qt C++ widget for plotting and data visualization. The library offers functionality for making 2D plots and graphs, visualizing real-time data, and handling user interaction on the plots or graphs. (Eichhammer, n.d.)

**mapz**

<https://www.mapz.com/> is a web page that offers city, regional and country maps of the entire world for immediate download based on open source geographical data (*About mapz*, n.d.). The map in the MCS was generated in mapz and downloaded as a PNG-file. Other formats are also available for download, and mapz also offers integration of interactive maps in web applications.

## 4.3 Application Components

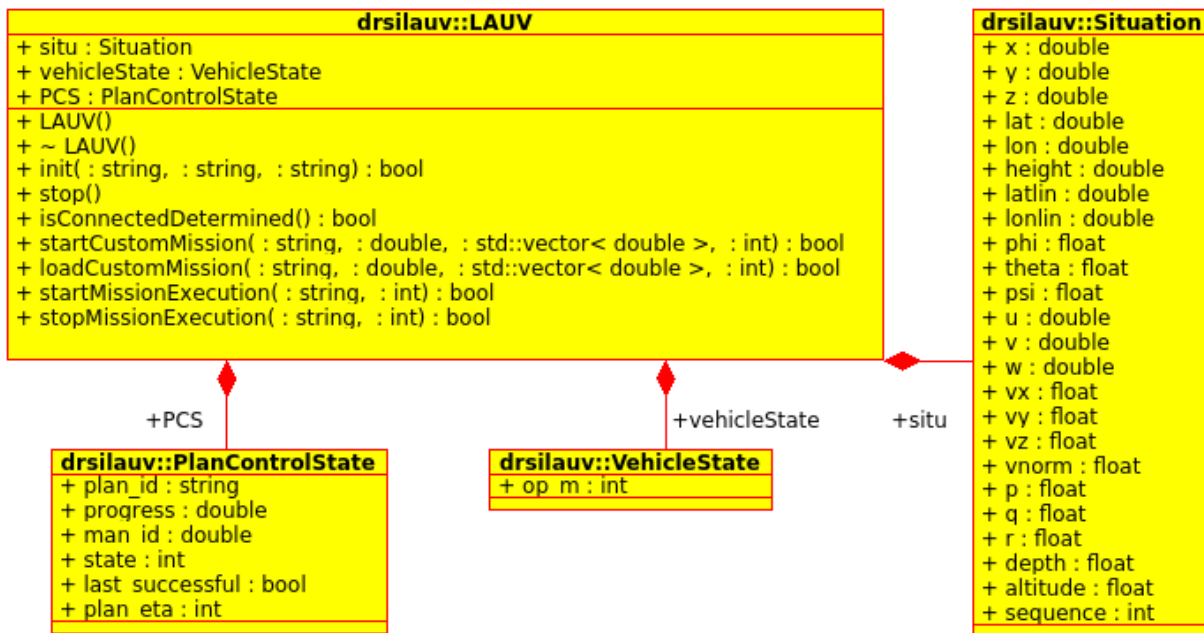
The MCS consists of four views and a library called MCSlib. MCSlib implements an LAUV class with functionality for communicating with a vehicle. The first three views are dedicated to a phase of mission planning or execution. In view 1, the user selects the vehicle team and sets details for the mission. View 2 allows the user to plan a path and load it onto the vehicles. View 3 enables the user to control mission execution and monitor the vehicles. The final view enables the user to induce test scenarios for testing the formation control method. The following sections describe the library and three first views in detail.

### 4.3.1 MCSlib

The LAUV class has functionality for handling three tasks; (1) connecting to a vehicle, simulated or real; (2) receiving and handling messages from the vehicle; and (3) sending mission plans and control commands. The library is categorized as a backend component because none of the functions are directly accessible by the user but rather executed on behalf of another part of the system. A brief description of the class and how the three tasks are handled follows.

### LAUV

Each vehicle object in the MCS is an instance of the LAUV class, and Figure 4.1 shows a class diagram with all the public functions and variables. The object holds three structs that summarize the vehicle and mission state; *Situation* holds information about the vehicle's position and pose; *VehicleState* holds the mode the vehicle is operating in, and *PlanControlState* holds information about mission execution.



**Figure 4.1:** Class diagram depicting all public functions and variables in the LAUV class.

### Task 1: Connecting to an LAUV

The application connects to a vehicle through a TCP-link by specifying the server address and port during initialization of the LAUV object. Fifteen attempts are made to connect to the vehicle in the course of as many seconds. An established connection means a successful initialization, and the system may begin to send and receive messages from the vehicle. The connection is continuously monitored with the function *isConnectedDetermined*.

### Task 2: Receiving and handling messages

All incoming messages are handled by the function *messageIn* which takes an IMC message as its only input. Every IMC message contains an id identifying the message type, and information about the source of the message, i.e., the name of the vehicle. Once the source is confirmed as a match to the vehicle object, the message is handled by a switch-case. Three message types are defined in the switch-case; estimated state, vehicle state, and plan control state, if the message is any other type, nothing is done. For an overview of the contents of the message types, see Section 3.4.

In the estimated state message the data concerning the vehicle's depth, heading and speed are stored in the Situation struct. The vehicle's position in latitude and longitude is computed based on the assumption that the earth is spherical using (4.1) given by *Calculate distance, bearing and more between Latitude/Longitude points* (n.d.). The *lat* and *lon* variables are constant through mission execution and represent the latitude and longitude where the vehicle was

initially booted.  $x$  and  $y$  are the displacement in meters from the initial booting point, the displacement is defined in the north-east-down frame.

$$\begin{aligned}\phi &= \Phi + x/R, \\ \lambda &= \Lambda + y/(R \cdot \cos \phi),\end{aligned}\tag{4.1}$$

where  $\phi$  = latitude,  $\lambda$  = longitude,  $\Phi$  = lat,  $\Lambda$  = lon, and  $R$  is the earth radius.

The only variable of interest in the vehicle state message is *operation mode*, which holds the mode of the vehicle. Possible values are; service, calibration, error, maneuvering, external control, and boot, and the mode is saved in the VehicleState struct. Data from the plan control state message is used to update the PlanControlState struct. The data in both structs is visualized in view 3.

### **Task 3: Sending mission plan and control commands**

The tasks of sending a mission plan, and starting and stopping mission execution are implemented in four functions; *startCustomMission*, *loadCustomMission*, *startMissionExecution*, and *stopMissionExecution*.

*startCustomMission* and *loadCustomMission* differ only by the operation request sent to the vehicle. The first function sends a *start* request in the plan control message, while the latter sends a *load* request. Both functions take four input variables; mission name, request ID (automatically generated by the MCS), list of waypoints, and the number of waypoints. Based on these inputs, plan control and plan specification messages are generated and sent to the vehicle. The waypoints are augmented with depth and speed data,  $z = 0$  and  $u = 1m/s$ , before *go to* messages are generated. *Plan transition* messages are generated based on the number of waypoints. The generated *go to* and *plan transition* messages are included in the plan specification message before it is sent to the vehicle.

*startMissionExecution* is used to start execution of a pre-loaded mission plan. The function takes two input variables; mission name and requestID (automatically generated by the MCS), and generates and sends a plan control message with a *start* request for the named mission.

*stopMissionExecution* is used when a user wishes to abort mission execution. It takes the same input as *startMissionExecution*, and sends a plan control message with a *stop* request to the vehicle.

### 4.3.2 View 1

View 1, shown in Figure 4.2, is the first an operator sees when the MCS starts. The view consists of four modules; *select vehicles*, *available vehicles*, *mission details*, and *formation control*, that allow the operator to review the available vehicles, set the vehicle team, formation, and constraints.

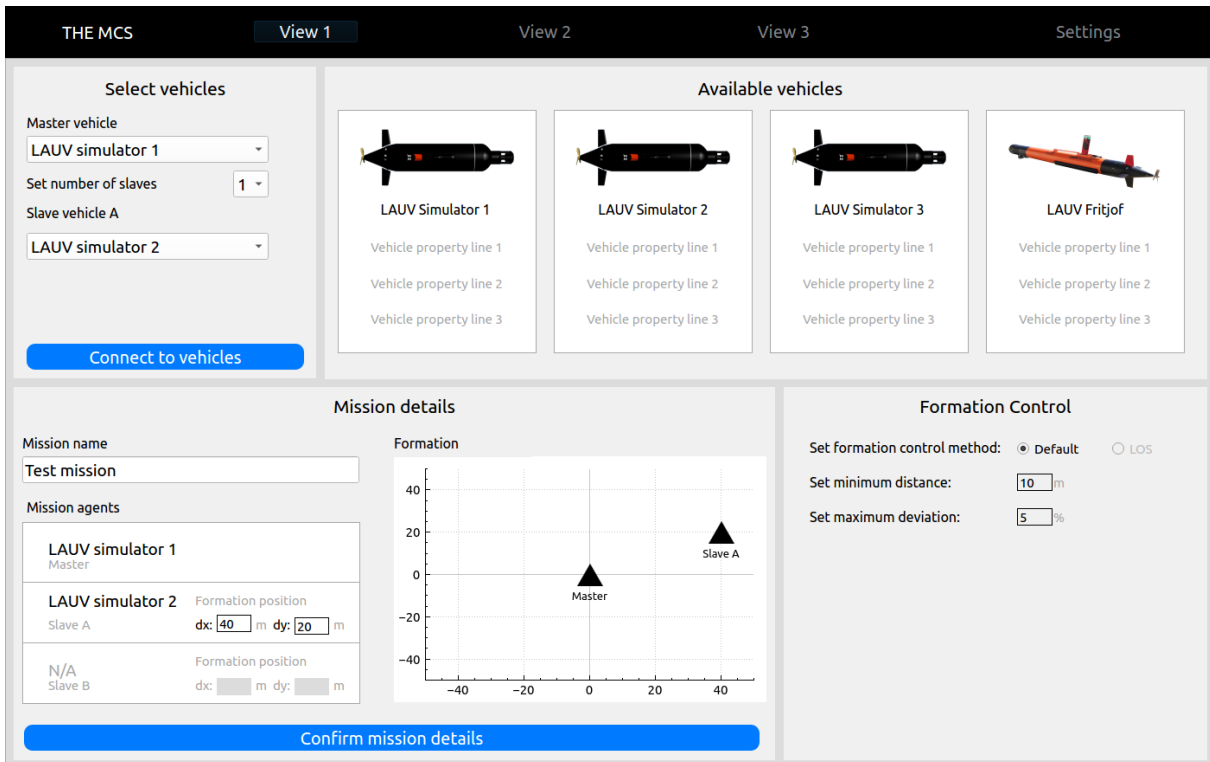
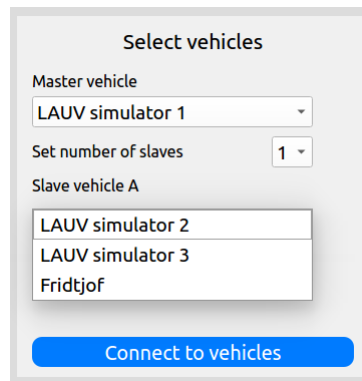


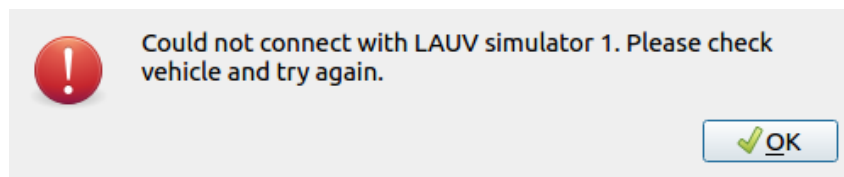
Figure 4.2: Screenshot of view 1.

#### Select vehicles

The select vehicles module is placed in the top left of the view and is a module for selecting a vehicle team of up to three vehicles, specifying them as either a master or slave and connecting to the vehicles. A dropdown menu appears for the user to assign a vehicle as master or slave, and once a vehicle is selected, it is removed from the other lists, as shown in Figure 4.3. Clicking the *connect to vehicles* button initializes an LAUV object for each team member, the server address and port for each vehicle is already stored in the system. The button text changes to *connected* when a connection is established. However, if the system is not able to establish a connection to a vehicle a warning (Figure 4.4) is prompted. In the case that the MCS only connects successfully to parts of the team, these vehicles and team assignments are locked, and the user is prohibited from changing these assignments. The only way to disconnect from a vehicle is to close the MCS.



**Figure 4.3:** Select vehicle module for a two-vehicle team showing dropdown menu for slave A.



**Figure 4.4:** Warning dialog prompted when the MCS is unable to connect to a vehicle.

### Available vehicles

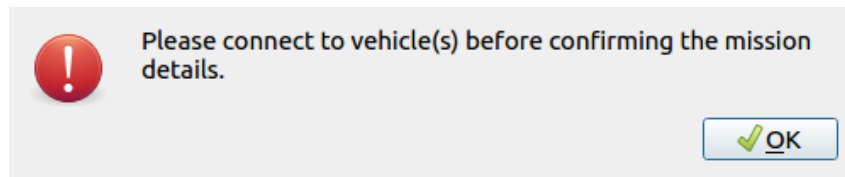
The available vehicles module is placed on the top right of the screen and has a frame for each vehicle stored in the MCS, containing the vehicle's name and picture. Space is reserved below for listing vehicle properties such as equipment and technical specifications.

### Mission details

The mission details module is placed in the bottom right of the view and is where the operator sets the vehicle formation and mission name. The operator inputs the displacement of each slave relative to the master and the formation is visualized in the plot generated with QCustomPlot. Clicking the *confirm mission details* button saves the mission name and formation. If the user wishes to change either the mission name or formation, the mission must be replanned. A button enabling replanning is placed in view 3. A prerequisite for confirming the mission details is that the vehicle team is confirmed, i.e., the system has successfully connected to the selected vehicles. If this is not the case, a warning (Figure 4.5) is displayed to the user.

### Formation control

The final module in this view is the formation control module. The formation control method for the mission can be selected using the radio buttons in the module. As only one formation control method has been implemented in this thesis, *default* is always selected. If other methods are added at a point in time, the user can select the preferred method by changing the radio button



**Figure 4.5:** Warning dialog prompted if the mission details are attempted confirmed before connections to the vehicles are established.

selection. Two constraint variables are inputted by the user in this view; *minimum distance*, and *maximum deviation*. Minimum distance,  $d_{min}$ , is entered in meters, and the value is used directly in the formation control loop. Maximum deviation,  $l$ , is inputted as a percentage and can be understood as a slave being allowed to deviate by up to  $l$  % of the given distance to the master. The value of  $\epsilon$  used in the control loop is given by (4.2).

$$\epsilon = \sqrt{dx^2 + dy^2} \cdot \frac{l}{100}. \quad (4.2)$$

### 4.3.3 View 2

View 2 is shown in Figure 4.6, and is the view where the operator generates a path for the mission and reviews the mission plan. The view has three modules; *map*, *path waypoints* and *mission details*.

#### Map

The map module takes up a large part of this view and is implemented with QCustomPlot. The map is an interactive plot with a background image which is placed so that the corner coordinates of the map coincide with the corners of the plot. The map depicts parts of the Trondheim Fjord where the field tests were conducted. Vehicles are represented by triangular markers showing the vehicles' positions and headings, and the markers are updated each time an estimated state message is received. The user adds waypoints to the mission plan by clicking the map, and the path is generated as straight lines between the waypoints. Only the master's path is visualized during this stage of the mission planning.



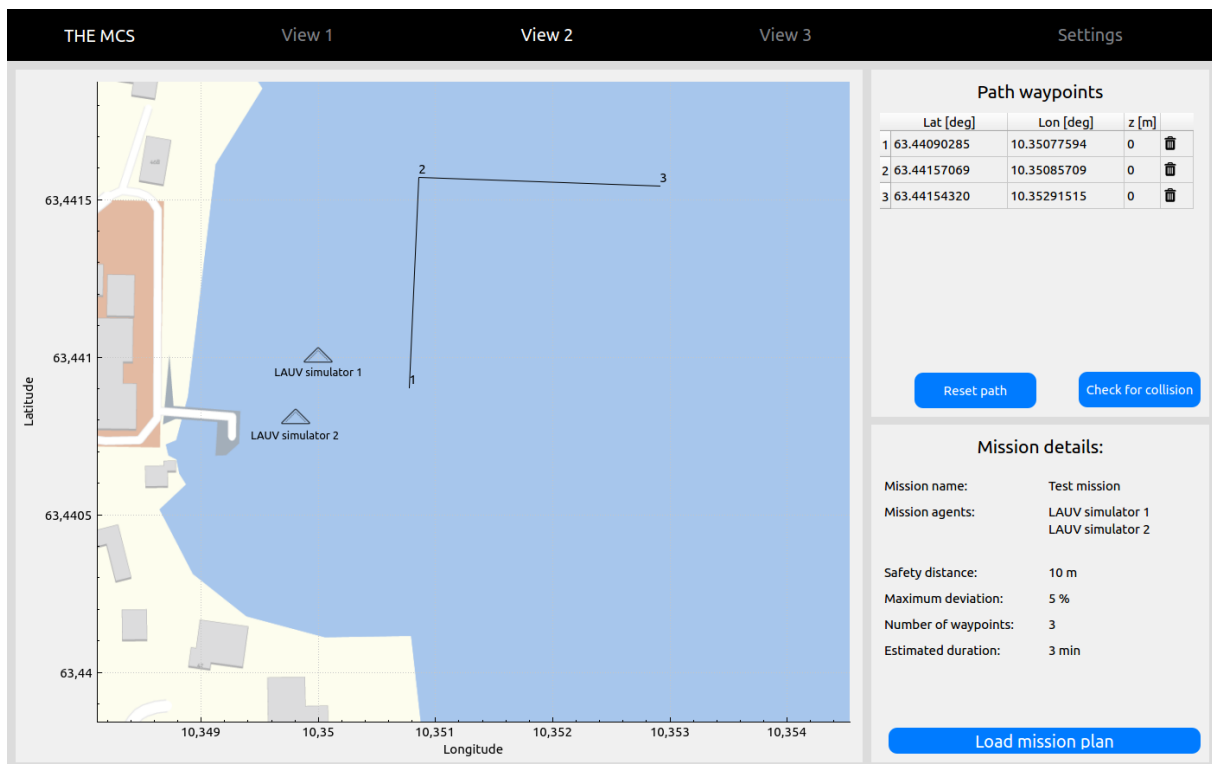


Figure 4.6: Screenshot of view 2.

### Path waypoints

A table listing the generated waypoints is available to the user in the path waypoints module placed in the top right of the view. The table holds id, latitude, longitude, and depth for each waypoint. The user may remove waypoints from the path by clicking the trash symbol in the leftmost column of the table. The remaining waypoints will then be connected to form a new path as seen in Figure 4.7.

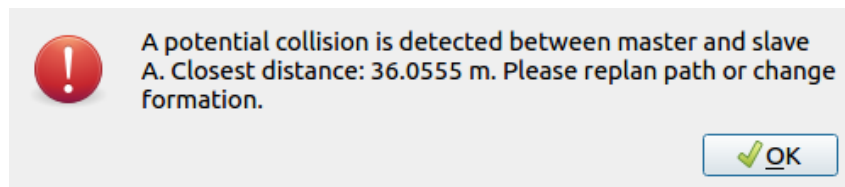
If the operator wishes to remove all the waypoints, the *reset path* button has this function. Clicking the *check for collision* button starts a function that checks the minimum distance between the vehicles' paths, and the result is either a warning dialog (Figure 4.8a) stating that the minimum distance is breached or an information dialog (Figure 4.8b) indicating that the path is safe.



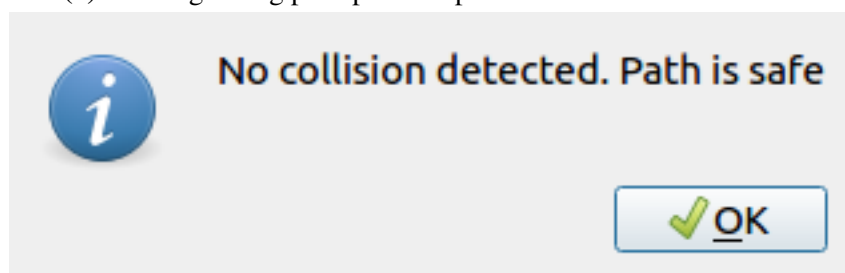
(a) Initial path with four waypoints.

(b) Modified path with three waypoints.

**Figure 4.7:** Path modification when a waypoint is deleted from the mission plan.



(a) Warning dialog prompted if a potential collision is detected.



(b) Information dialog prompted if no potential collisions are detected.

**Figure 4.8:** Notification dialogs prompted after a path collision check.

### **Mission details**

An overview of the mission is placed in the lower right part of the view and comprises the data in view 1 and the path waypoint list. In addition to data copied from other modules, the estimated duration of the mission is displayed. This number is computed by dividing the length of the path by the desired speed. When the user clicks the *load mission plan* button, a series of checks are run which may result in either one of the following warnings being prompted to the user.

- Please connect to vehicle(s) before loading mission plan.
- Please confirm mission details before loading a mission plan.
- Please enter a safety distance.
- Please enter a maximum deviation.
- A potential collision is detected (..) Please replan path or change formation.

If all the checks are passed, the `loadCustomMission` function is called for each vehicle with the required mission details as input. When this is done, the slaves' paths are drawn on the map, and view 3 is displayed.

#### **4.3.4 View 3**

View 3 (Figure 4.9) is where an operator monitors and controls mission execution. The previously discussed map module is also included in this view, the only change is that user interactions are disabled in this view. Two modules are placed to the right of the map; *vehicle overview* and *mission control*.

#### **Vehicle overview**

The operator may monitor each team member's position, speed, depth and operation mode in the vehicle overview module. The overview of a single vehicle is shown in Figure 4.10. This component is copied from the previous version of the MCS, with slight modification. The state field holds one of the following texts; ready, error, maneuvering, or busy. Busy represents the operation modes calibration, boot and external control. The remaining are copied directly from the vehicle state message. The green circle represents a connection to the vehicle, if the connection to the vehicle is lost the user prompted with a warning (Figure 4.11) and the light turns grey.

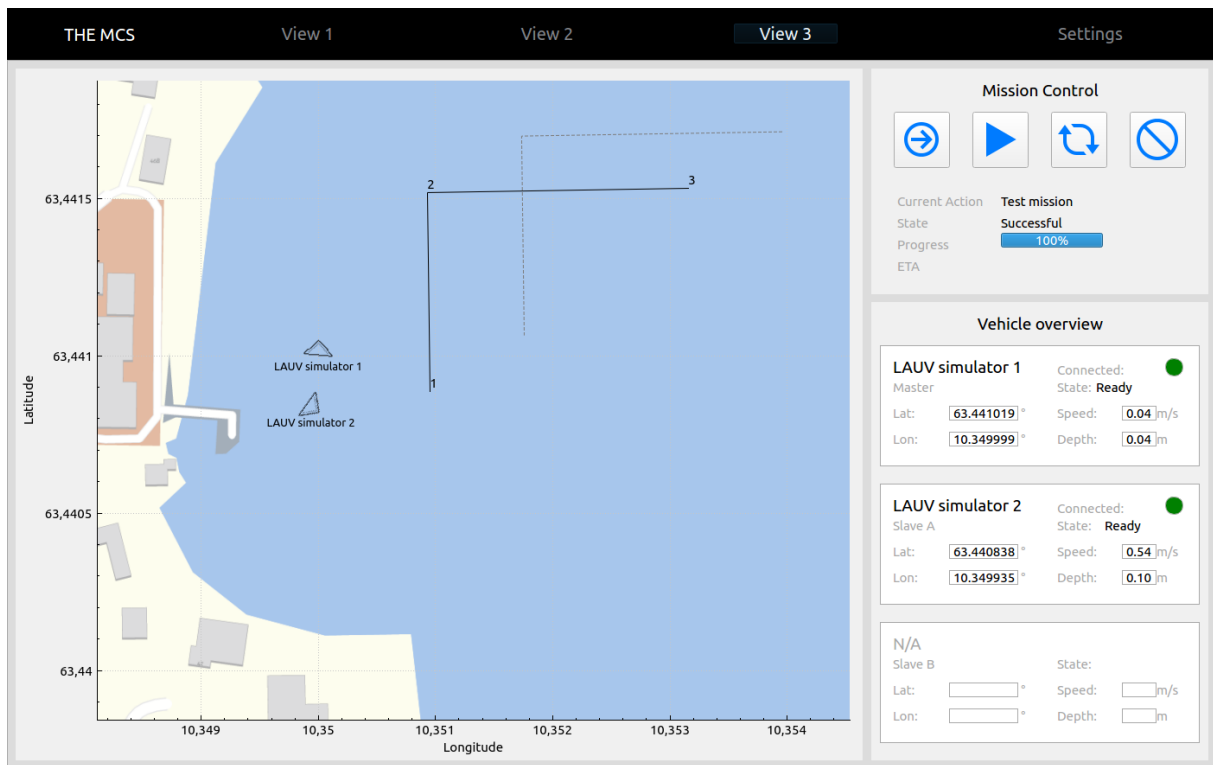


Figure 4.9: Screenshot of view 3.

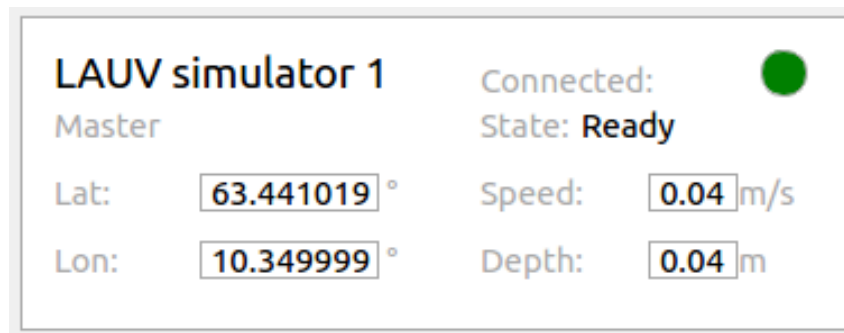


Figure 4.10: Vehicle overview for a single vehicle depicting a connected vehicle in a ready state.

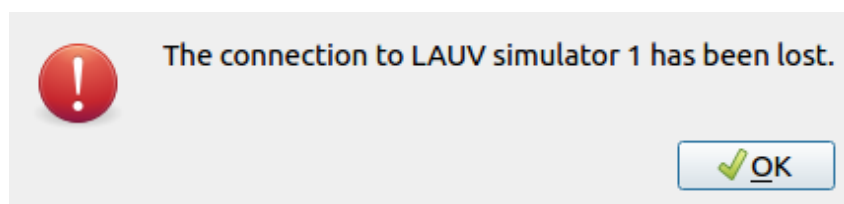


Figure 4.11: Warning dialog prompted when the connection to a vehicle is lost.

### **Mission control**

The mission control module informs the operator about the mission progress and has four buttons for mission control. The mission information displayed to the user includes; mission name, state, progress, and estimated time of arrival. The state of the mission can be one of two things; executing or successful. If a mission is aborted, this field is cleared. The mission progress and estimated time of arrival are set by comparing the data in the `planControlState` struct of each vehicle and selecting the lowest and highest values, respectively.

The buttons from left to right represent; *prepare mission execution*, *start mission execution*, *replan mission*, and *abort*. Clicking the prepare mission execution button generates a waypoint list consisting of two waypoints, and sends this as input to the `startCustomMission` function. How the two waypoints are generated is discussed in Section 3.5.2. Clicking the button ultimately results in each team member maneuvering to their initial point on the path, i.e., getting in formation. Clicking the start mission execution button sends a call to the `startMissionExecution` function. This button also starts the formation control loop enabling cooperative strategies to be invoked if a constraint is violated. Clicking the replan mission button sends a call to the `stopMissionExecution` function, stops the control loop, enables editing of the mission details set in view 1, and clears the list of waypoints in view 2. The abort button sends a call to the `stopMissionExecution` function and stops the control loop.

## **4.4 Verification**

Once implementation of the MCS concluded, a verification process took place to ensure that the application met the requirements and worked adequately. The process was split into two parts; evaluation of application requirements and comparison to Neptus.

### **4.4.1 Evaluation of Application Requirements**

Table 4.2 presents the evaluation of the application requirements. The *evaluation* column presents the author's justification of why a requirement is or is not attained, and the *fulfillment* column states whether or not the requirement was attained. The requirements are stated in Section 4.2.2.

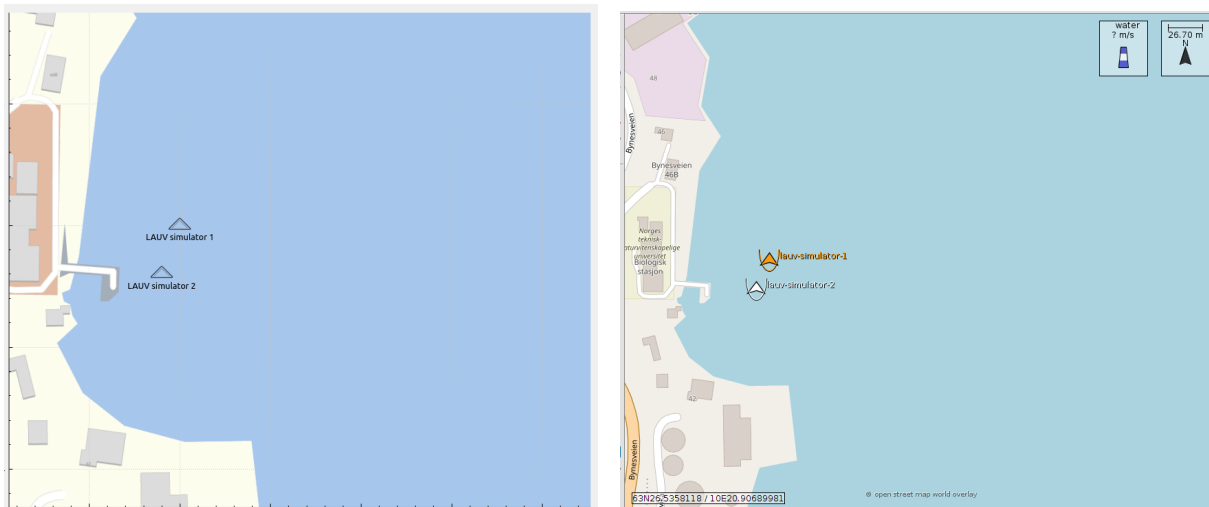
**Table 4.2:** Evaluation of application requirements.

Requirement ID	Evaluation	Fulfillment
R1	All communication between the MCS and the LAUVs is done via IMC messages.	Attained
R2	All system functionality works at it should when connected to a network that does not provide an internet connection, i.e., the network provided by the Manta.	Attained
R3	A warning is prompted to the user if the communication link to a vehicle is lost.	Attained
R4	The system automatically reconnects when a vehicle comes back online. The user is not explicitly notified, but the connection light in the vehicle overview module goes from grey to green.	Attained
R5	Disabling control and command functionality when the connection to a vehicle is lost has not been implemented. The clear warning a user receives when a connection is lost, caused this task to be less of a priority.	Not attained
R6	The MCS generates a path for each slave by displacing the master's waypoints. The slave paths are visualized when the mission plans are loaded onto the vehicles.	Attained
R7	Functionality for loading and sending a mission plan is split into two buttons in view 2 and 3, respectively.	Attained
R8	The mission plan is available for the operator in view 2 in the form of a table displaying the waypoints. The paths the vehicles are intended to follow are also visualized on the map.	Attained
R9	Mission execution is monitored in view 3.	Attained
R10	Clicking the abort button in view 3 aborts any ongoing mission for all the vehicles in the team.	Attained
R11	Clicking the replan mission button enables editing of the previously confirmed mission details, i.e., mission name, formation, constraints, and waypoints.	Attained
R12	The path is automatically generated when the user defines waypoints by clicking on the map.	Attained
R13	Handling the callbacks from the vehicle has not been implemented in the MCS. This functionality was not prioritized as the full mission plan is sent at once, eliminating the possibility of the vehicle only partly executing a mission due to communication failure.	Not attained

### 4.4.2 Comparison to Neptus

Neptus is a trusted command and control tool, and comparing the information given to an operator in the MCS to the information given by Neptus is a way to ensure that the implementation of the MCS is satisfactory. The verification was in two parts; verifying the map and placement of vehicle markers, and verifying the mission planning, monitoring, and control abilities through three use cases.

The maps and placement of the markers were verified first. Figure 4.12 shows the vehicle markers relative to the pier by the field test site in both Neptus and the MCS, and the placement of the markers appear to coincide in the two applications. Tests were also conducted in the field to confirm the observations made when comparing the two application. Before field tests, the LAUV was moved around on land to check that the position relative to buildings marked on the map was correctly visualized in the MSC. The marker was responsive, and the relative position looked correct. Based on these two tests the markers were considered to have satisfactory accuracy.

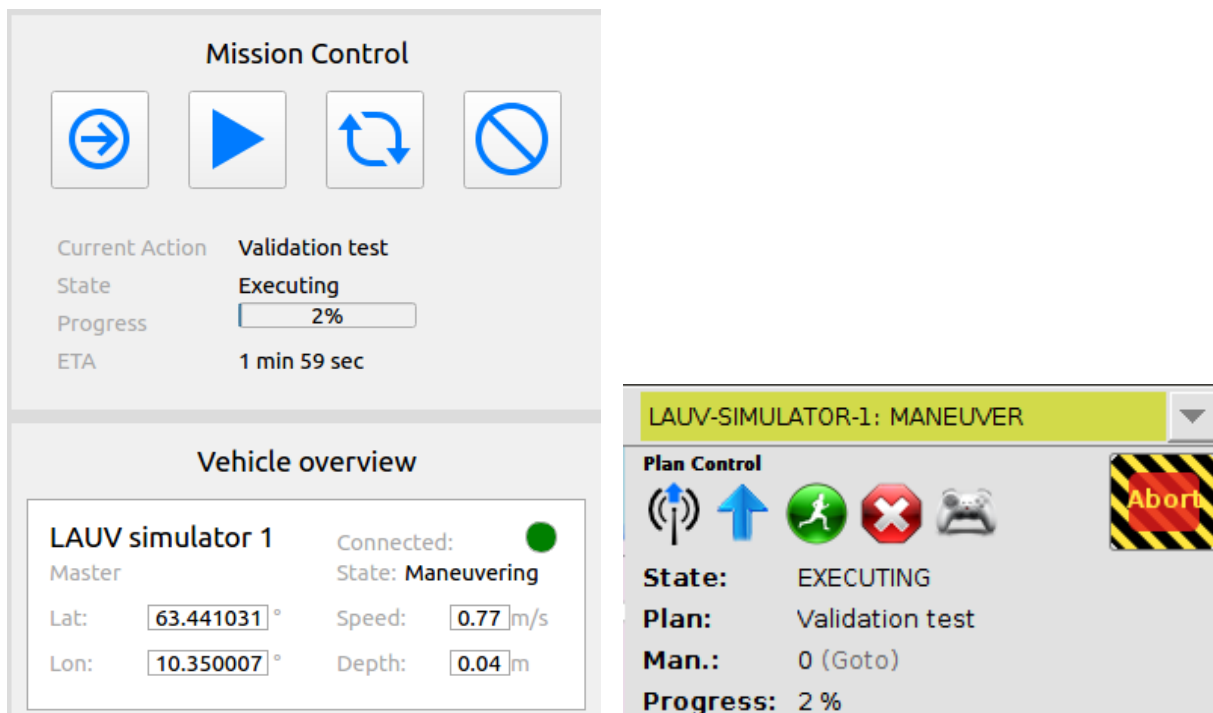


(a) Screenshot of the map in the MCS showing the vehicle marker relative to the pier by TBS.

(b) Screenshot of the map in Neptus showing the vehicle marker relative to the pier by TBS.

**Figure 4.12:** Verification of marker placement in the MCS by comparison to Neptus.

The first use case was defined as starting a mission in the MCS. Neptus was then used to verify that the mission plan was received by the vehicle and that mission execution began. A mission was defined in the MCS and named *Validation test*. Mission execution was started through the application and Figure 4.13 shows the mission details in Neptus and the MCS in the early stages of mission execution. The mission name, vehicle state and mission progress matched and the test was considered successful.



(a) Screenshot of the mission control and vehicle overview module in the MCS during mission execution.

(b) Screenshot of the mission control and vehicle overview module in Neptus during mission execution.

**Figure 4.13:** Verification of mission details for a mission defined in the MCS by comparison to Neptus.

The second use case was to start a mission from Neptus and check if this was registered by the MCS. The vehicle state changed to maneuvering and the vehicle marker followed the vehicle's position. The mission details, i.e., mission name, progress, and ETA were not possible to monitor in the MCS, which was as expected. The use case was considered successful. The final use case was to start a mission from either one of the control and command applications and abort it from the other. Aborting mission execution was successful in both cases, and the use case was considered a success. After completing the verification, the author is confident in saying that the MCS performs as expected, fulfills most of the requirements, and is satisfactory as a control and command software for its intended purpose.



### 4.4.3 Additional Requirements

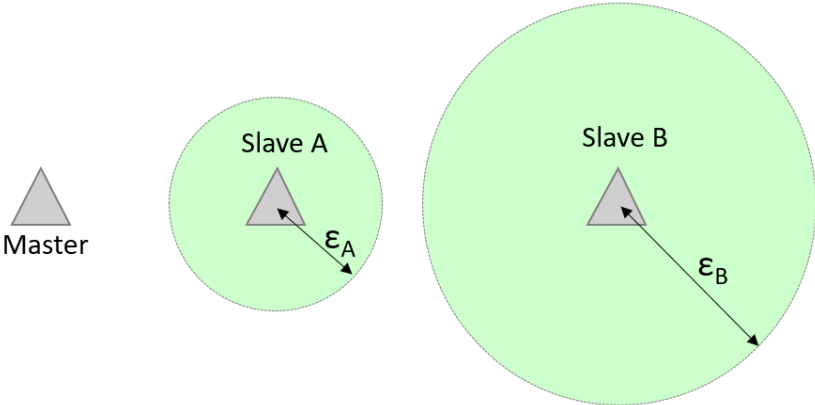
A list of additional requirements was comprised based on experiences after completing verification, field tests, and simulations. The additional requirements are stated in Table 4.3, and left as further work. AR1 and AR2 specify how the system should handle a detected error in one of the team members. Currently, an error in a vehicle would be depicted in the vehicle overview module, but the user is not explicitly warned, and the system does not take any action. AR3 is motivated by the fact that it is not possible to change the vehicle team once all the vehicles are connected. The only way to disconnect from a vehicle is to close the application, which over-complicates the case of modifying a vehicle team for a defined mission.

AR4 addresses the conversion between the maximum deviation inputted by the user and the deviation  $\epsilon$  used in the formation control. The current implementation results in different deviation boundaries  $\epsilon$  for two slaves if they are not placed symmetrically about the master, which may result in skewed vehicle formations. The new requirement states that the variable  $\epsilon$  be set directly by the user, yielding a more restricted formation than the current implementation. Both cases are illustrated in Figure 4.14. AR5 specifies that the system should have functionality for detecting if a mission becomes infeasible during mission execution. This functionality can be useful for an operator if the constraints are violated continuously due to external factors deeming the cooperative strategies insufficient for handling the constraint violations.

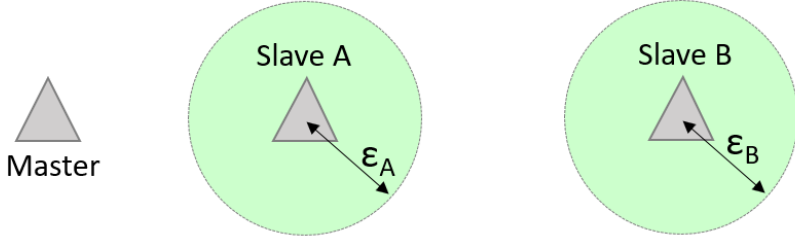
There is a requirement omitted from Table 4.3, as it requires modification in the vehicle software and not the MCS, however, it illustrates an important case. In the event of an unexpected failure in the MCS during mission execution, the vehicles continue maneuvering, but the formation control may be disabled. A system requirement addressing this case if formulated as follows; *The vehicles should abort mission execution and go into a safe mode if the MCS unexpectedly powers off.*

**Table 4.3:** Additional requirements for the MCS comprised after verification and testing.

Requirement ID	Requirement
AR1	If an error is detected in a master, the system should abort any ongoing mission, and set all vehicles in a safe mode.
AR2	If an error is detected in a slave, the system should put the vehicle out of service and in a safe mode without interrupting any ongoing mission.
AR3	It should be possible to disconnect from a vehicle while still in the application.
AR4	A user should be able to directly set the maximum allowed deviation, $\epsilon$ , in meters.
AR5	The system should have functionality for identifying if a mission becomes infeasible during mission execution.



(a) Current implementation of maximum deviation in a three-vehicle team.



(b) Suggested modified implementation of maximum deviation in a three-vehicle team.

**Figure 4.14:** Visualization of the implemented and suggested maximum deviation functionality.

## Simulations

Initial testing of the MCS and the formation control method, in particular, was performed through simulations with the LAUV simulators. The simulations were conducted first and foremost to validate the cooperative strategies used in the formation control method in a controlled environment before proceeding with field testing. In this section, results from four missions created to demonstrate the cooperative strategies designed in Section 3.5.3 are presented. In the case that the desired constraint violation did not occur naturally, vehicles were manually stopped and started, i.e., mission execution was aborted or resumed. All missions were conducted on the surface, without restrictions on the minimum distance between the vehicles. Each mission was named based on the desired test scenario, and the missions relate to the cooperative strategies in the following manner:

**Section 5.1:** *Slave B falls behind mission; CS1 and CS5.*

**Section 5.2:** *Master falls behind mission; CS2.*

**Section 5.3:** *Slave catches up to master mission; CS3.*

**Section 5.4:** *Master catches up to slave mission; CS3 and CS4.*

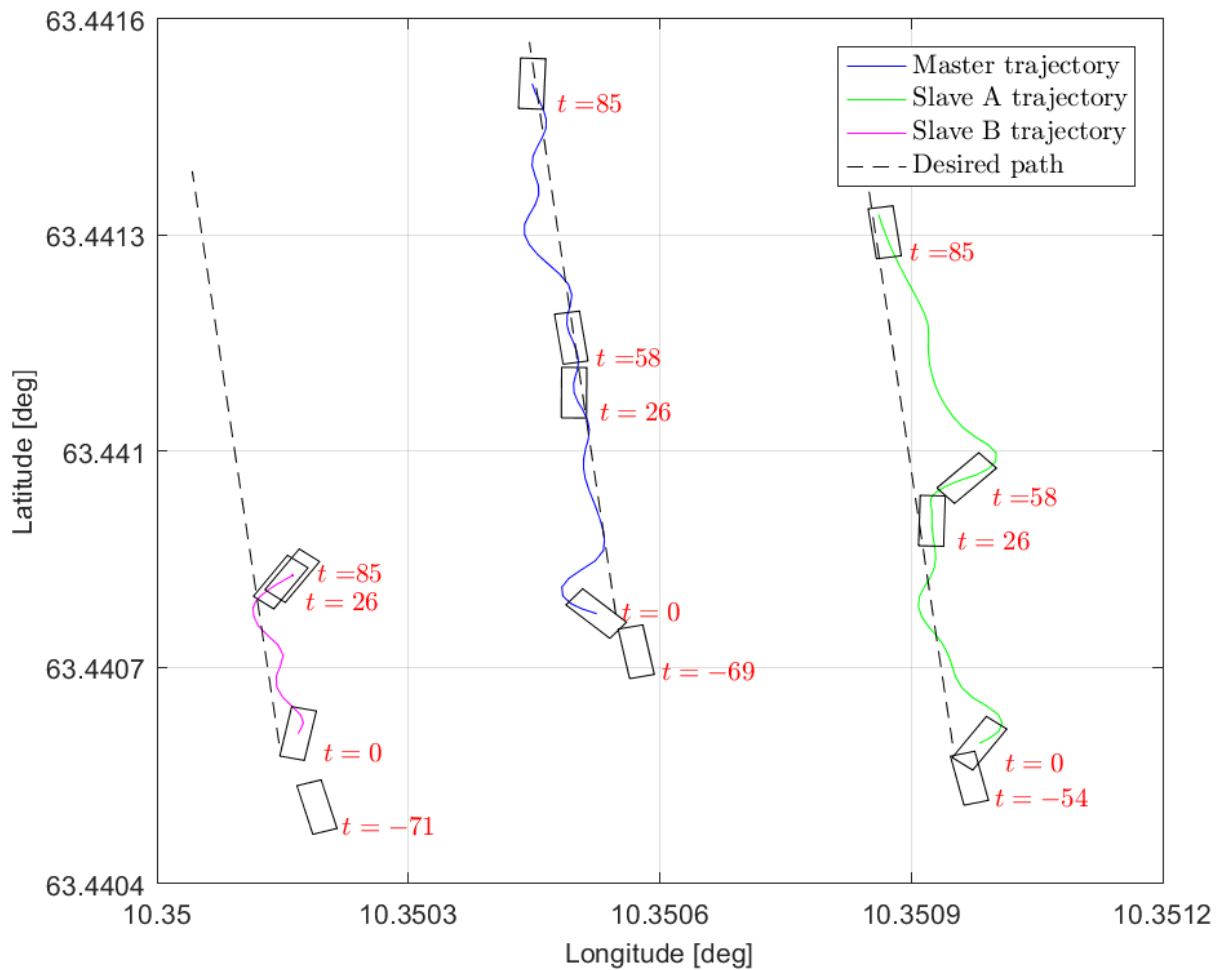
Additional simulations were performed where the number of slaves, formation, and constraints varied. The results presented in this chapter are representative of the observations made in the majority of these simulations, but there were exceptions where the system did not perform as expected. These cases are not considered representative of the system, but as isolated incidents. An outlier in the results is presented in Section 5.5. The chapter concludes with a discussion of the presented results.

## 5.1 Slave B Falls Behind Mission

Slave B falls behind mission was designed to demonstrate CS1 and CS5. The mission entailed a three-vehicle team maneuvering north in parallel vertical lines, and the team and formation are specified in Table 5.1. Maximum allowed deviation was set to 20%, which translates to  $\pm 5.7$  m for both slaves. The vehicles' paths and trajectories are depicted in Figure 5.1, and the vehicles' positions and headings at times of interest are represented by rectangular markers.  $t = 0$  marks the beginning of mission execution.

**Table 5.1:** Overview of vehicle team and formation in Slave B falls behind mission.

Assignment	Vehicle	$\Delta x$	$\Delta y$
Master	LAUV Simulator 1		
Slave A	LAUV Simulator 2	20 m	-20 m
Slave B	LAUV Simulator 3	-20 m	-20 m



**Figure 5.1:** Vehicle trajectories in Slave B falls behind mission.

Slave B reaches its initial position in the formation at  $t = -71$ , and by the time mission execution begins the slave has drifted from this position, and the heading has changed slightly. Drifting is also observed in the master and slave A, who reach their initial positions at  $t = -69$  and  $t = -54$ , respectively. Their change in heading is more significant compared to slave B. The vehicles are able to follow their paths and preserve the formation up until  $t = 26$ , and one can observe this in the trajectory plots.

Figure 5.2 shows the distance between the master and both slaves and the speed of all three vehicles during mission execution. One can observe in Figure 5.2a that the distance between the master and slave A lies within the constraints throughout mission execution, not activating any cooperative strategies. Slave B is manually stopped fourteen seconds into mission execution, and this is marked in all three plots. In Figure 5.2c it can be seen that there is a delay between when the vehicle is stopped and when the vehicle's speed begins to decrease. From this point on, the distance between the master and slave, displayed in Figure 5.2b, starts to increase.

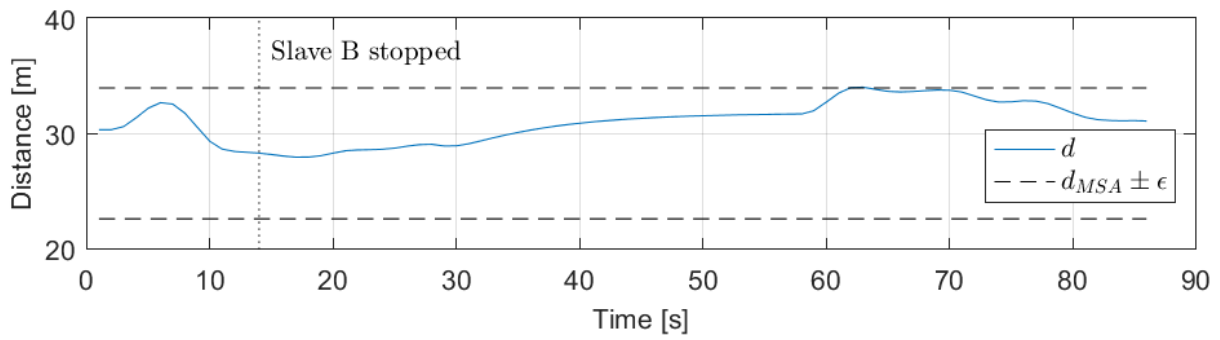
The constraint violation occurs in  $t = 26$ , twelve seconds after slave B is stopped. The vehicles' progress' at this point is given in Table 5.2, and it shows that slave B has lower mission progress than the master at this time. The two conditions for activating CS1 are satisfied, and at  $t = 28$ , two seconds after the constraint violation, the speed of the master and slave A start to decrease, indicating that they switched behaviors from path following to formation preservation. Slave A switches behaviors due to CS5 which is activated the moment the master needs to switch behavior. The speed of the two vehicles picks back up 30 seconds later, which is the maximum waiting period for a master.  $t = 58$  marks the resumption of mission execution, and as one can see in the trajectory plots, slave B is still at rest close to the position where it was initially stopped.

During the waiting period both the master and slave A drift. Slave A experiences a significant change of heading which results in a deviation from the path for the remainder of the mission. The vehicles' positions are marked at  $t = 85$ , moments before the mission is completed, and shows that the master and slave A complete the mission while holding the formation, and that slave B is left behind. The behavior of the master and slave A are in line with the actions defined in CS1 and CS5.

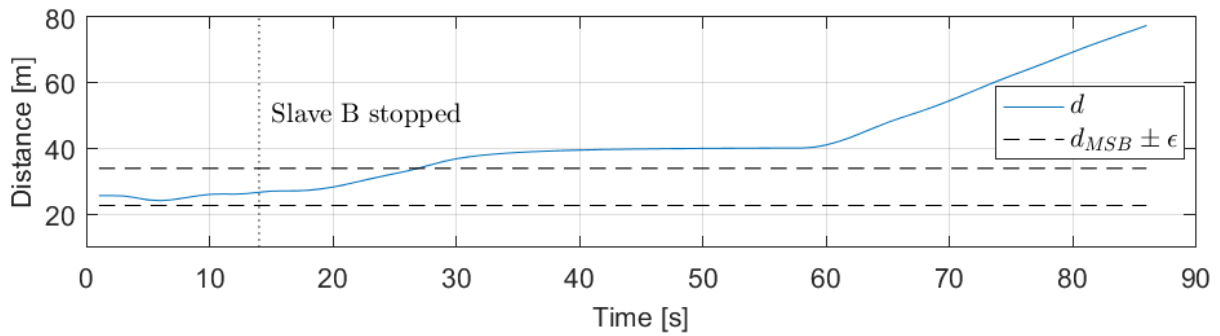
The final thing to note in these results is depicted in Figure 5.2c. All the team members exceed the desired speed of 1 m/s, but their speed profiles are quite similar.

**Table 5.2:** Mission progress during execution of Slave B falls behind mission.

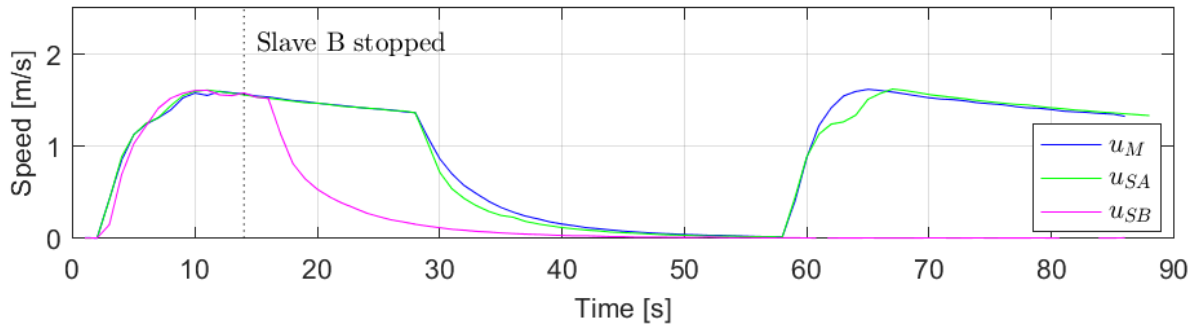
Time	Master progress	Slave A progress	Slave B progress
26 s	58.47 %	57.317 %	49.242 %



(a) Distance between master and slave A relative to constraints during execution of Slave B falls behind mission.



(b) Distance between master and slave B relative to constraints during execution of Slave B falls behind mission.



(c) Vehicle speed during execution of Slave B falls behind mission.

**Figure 5.2:** Speed and distance data from Slave B falls behind mission.

## 5.2 Master Falls Behind Mission

CS2 was demonstrated through Master falls behind mission, and the mission entailed a two-vehicle team maneuvering east on straight parallel lines. The vehicle team and formation are specified in Table 5.3, and the maximum allowed deviation was set to 15 %, which translates to  $\pm 7.5$  m.

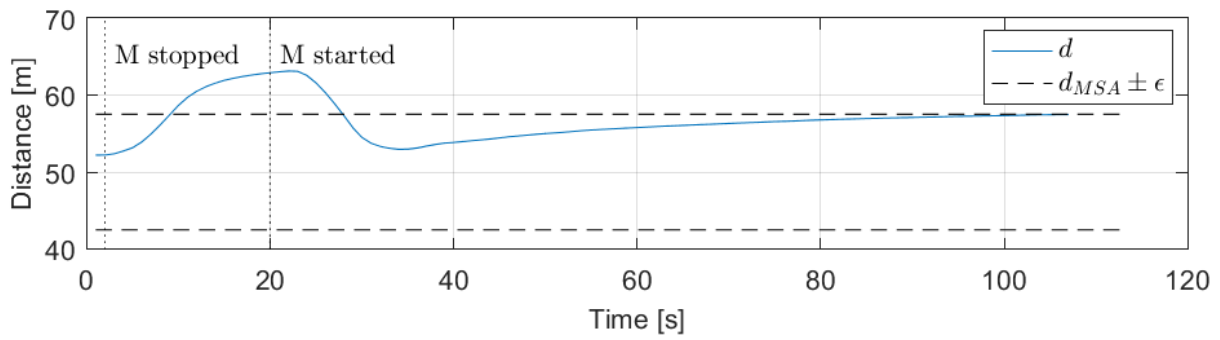
**Table 5.3:** Overview of vehicle team and formation in Master falls behind mission.

Assignment	Vehicle	$\Delta x$	$\Delta y$
Master	LAUV Simulator 1		
Slave A	LAUV Simulator 2	50 m	0 m

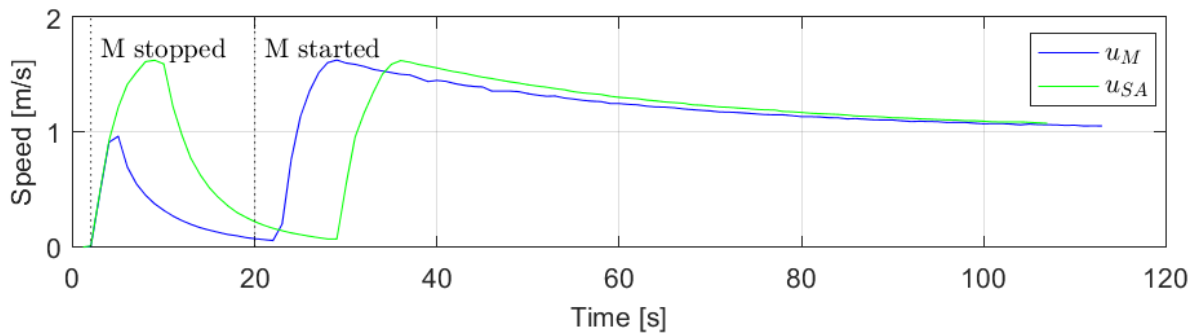
Figure 5.3 shows the logged distance, speed and progress data from the mission, and  $t = 0$  marks the beginning of mission execution. Before  $t = 0$ , the vehicles have maneuvered to their initial position in the formation. To induce CS2, the master is stopped at  $t = 2$ , and this is marked in all three plots. The slave keeps moving forward, and the distance between the two vehicles increases, as can be observed in Figure 5.3a. The upper constraint for the distance between the two vehicles is violated at  $t = 9$ , and the vehicles mission progress at that moment is seen in Figure 5.3c. The plot of the progress shows that the slave's progress is higher than that of the master.

The conditions for activating CS2 are satisfied, and a moment later at  $t = 10$ , one can observe in Figure 5.3b that the slave starts to reduce its speed. The reduction in speed indicates that the slave has switched to formation preservation behavior, allowing the master to catch back up and restore the formation. Although the master is still at rest, the distance between the vehicles continues to increase after this point as it takes time for the slave to reduce its speed to 0 m/s. The master is commanded to resume mission execution at  $t = 20$ , and the slave switched back to path following behavior at  $t = 29$ , after the distance between the vehicles is back within the boundaries.

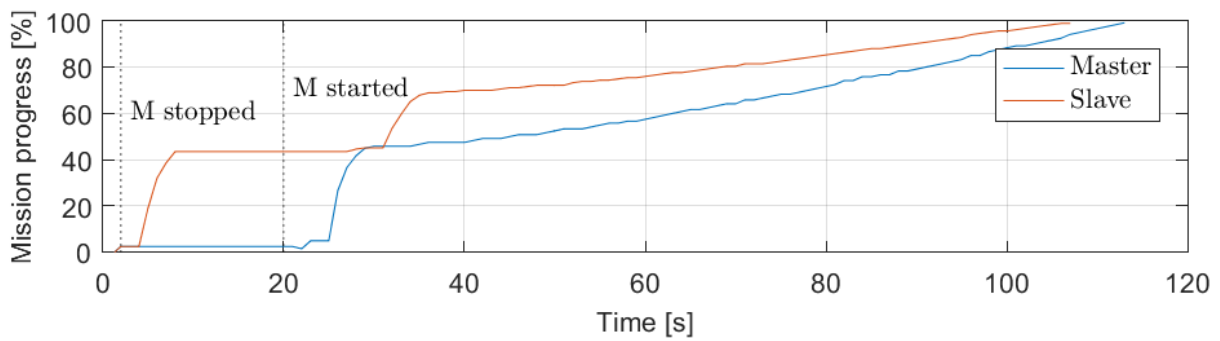
Note how the mission progress of both vehicles does not grow uniformly but has one or two growth spurts during mission execution. It can be observed that the vehicles are accelerating each time a growth spurt occurs. Also, as in Slave B falls behind mission, the vehicles maneuver at a higher velocity than specified, but the speeds seem to be converging towards 1 m/s, and the profiles are similar.



(a) Distance between master and slave A relative to constraints during execution of Master falls behind mission.



(b) Vehicle speed during execution of Master falls behind mission.



(c) Mission progress during execution of Master falls behind mission.

**Figure 5.3:** Speed, distance, and progress data from Master falls behind mission.



### 5.3 Slave Catches up to Master Mission

Slave catches up to master mission was defined as the vehicles maneuvering east in parallel horizontal lines and designed to test CS3. The mission was conducted by a two-vehicle team, and the team and formation are specified in Table 5.4. The maximum deviation was set to 15 %, which translates to  $\pm 4.2$  m.  $t = 0$  marks the point in time when mission execution began. The vehicles' paths and trajectories are depicted in Figure 5.4, and the vehicles' positions and headings at times of interest are represented by rectangular markers.

**Table 5.4:** Overview of vehicle team and formation in Slave catches up to master mission.

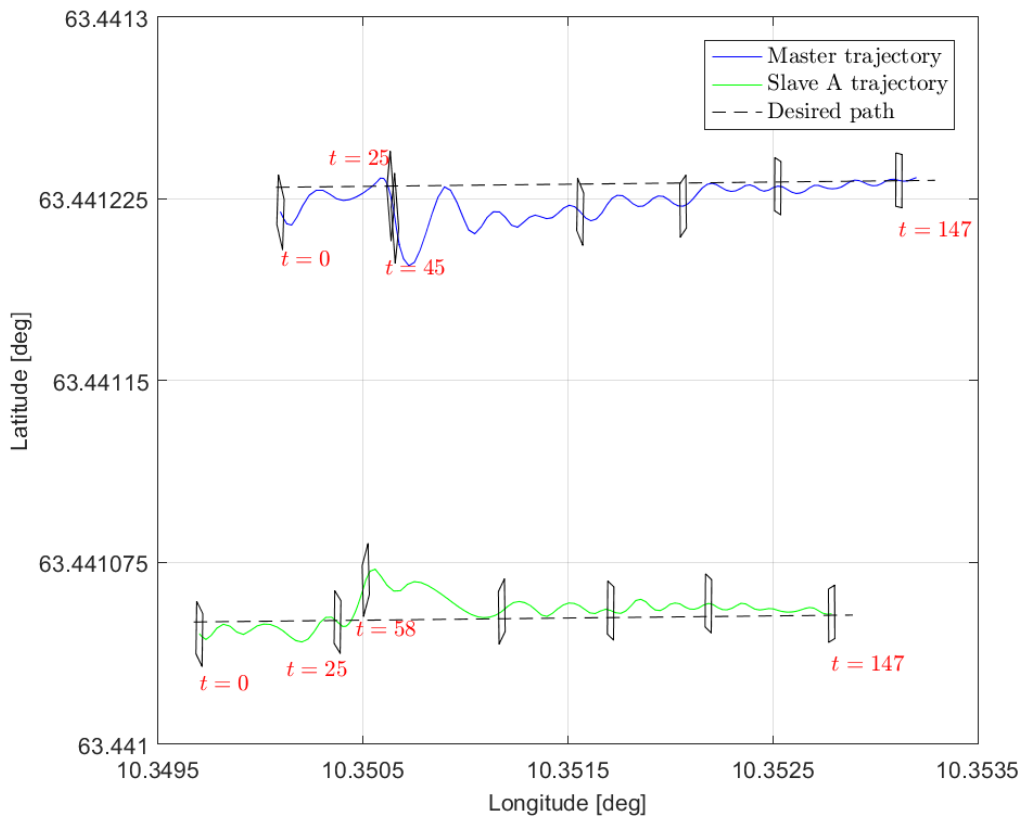
Assignment	Vehicle	$\Delta x$	$\Delta y$
Master	LAUV Simulator 1		
Slave	LAUV Simulator 2	-20 m	-20 m

To induce the conditions for activating CS3 the master is manually stopped at  $t = 17$ , while slave A continues mission execution, this results in the distance between the vehicles decreasing. Up until  $t = 25$ , the vehicles are able to stay within the formation, but as can be seen in Figure 5.5a this is when the constraint on the distance between the master and slave is violated. Table 5.5 shows the mission progress at the time of the constraint violation, which indicates that the slave has higher progress than the master.

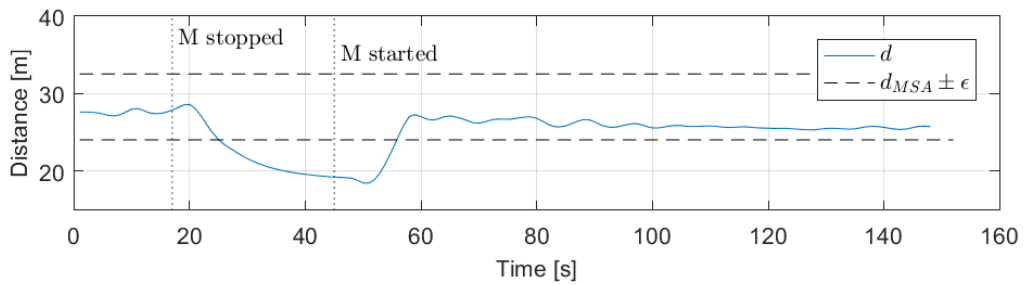
**Table 5.5:** Mission progress during execution of Slave catches up to master mission.

Time	Master progress	Slave progress
25 s	42.59 %	43.852 %

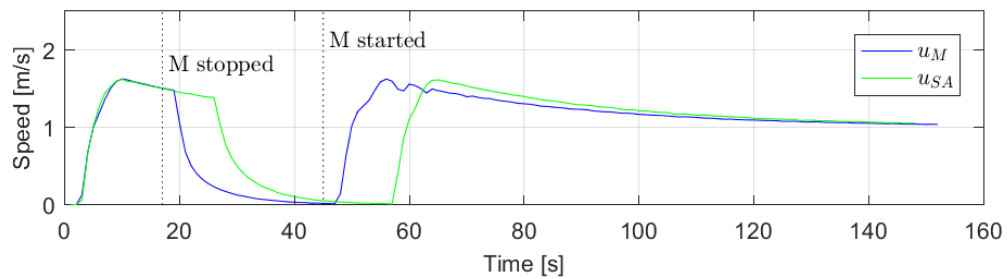
The conditions for activating CS3 are satisfied, and slave A has a visible response the next second when the vehicle's speed decreases, this is observed in Figure 5.5b. The speed reduction down to 0 m/s indicates that slave A switched behaviors and is now in formation preservation mode. Except for some drifting, both vehicles are at rest until the master is restarted at  $t = 45$ . The constraint on the distance is again satisfied at  $t = 56$ , and slave A resumes mission execution shortly after at  $t = 58$ , an action which is in accordance with the expected behavior of slave A. Notice that both vehicles deviate even further from their path when mission execution is resumed after a stop. However, both converge towards the path after a short while, and the vehicles are able to keep the formation for the remainder of the mission.



**Figure 5.4:** Vehicle trajectories in Slave catches up to master mission.



**(a)** Distance between master and slave relative to constraints during execution of Slave catches up to master mission.



**(b)** Vehicle speed during execution of Slave catches up to master mission.

**Figure 5.5:** Speed and distance data from Slave catches up to master mission.

## 5.4 Master Catches up to Slave Mission

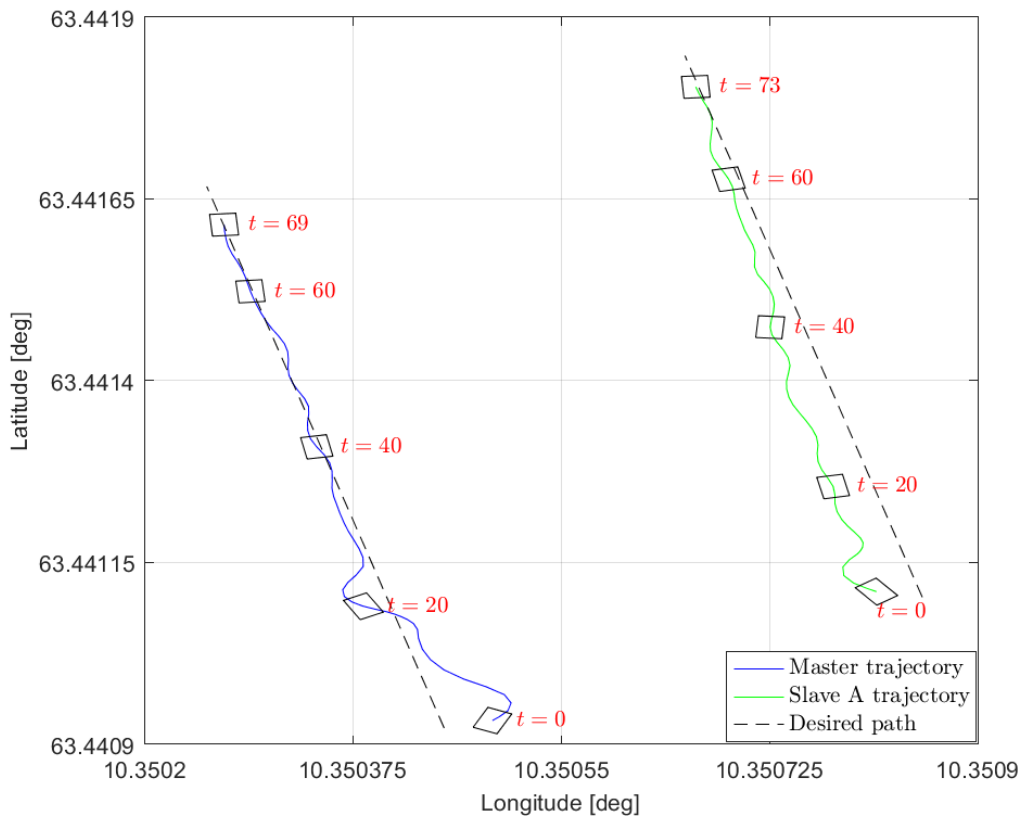
Master catches up to slave mission was designed to demonstrate CS4, however, during mission execution, CS3 was also activated. The mission objective was for two vehicles to maneuver north in parallel vertical lines, and the mission team and formation is given in Table 5.6. Maximum allowed deviation was set to 15%, which translates to  $\pm 4.2$  m. None of the constraint violations that occurred during mission execution were induced.

**Table 5.6:** Overview of vehicle team and formation in Master catches up to slave mission.

Assignment	Vehicle	$\Delta x$	$\Delta y$
Master	LAUV Simulator 1		
Slave	LAUV Simulator 2	20 m	20 m

The generated paths for the vehicles and their trajectories are shown in Figure 5.6, and the vehicles' positions and headings at times of interest are represented by rectangular markers. The slave deviates from the path nearly throughout mission execution, whereas the master is able to converge after approximately forty seconds. The vehicles' trajectories show that the deviation from the path is more significant in the early stages of mission execution, and this is also when the constraints are violated.

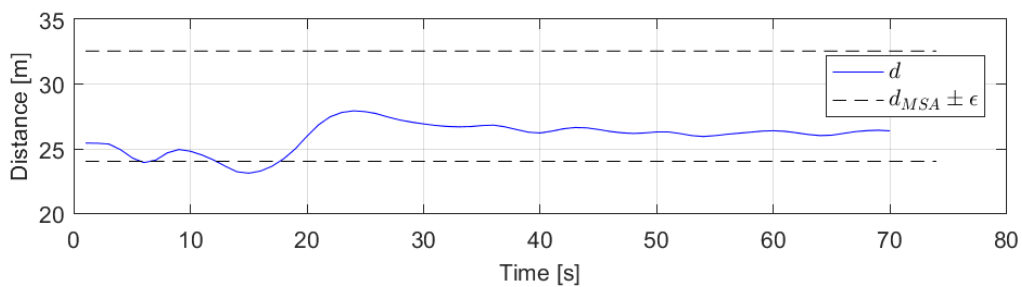
The two constraint violations occur at  $t = 6$  and  $t = 12$  and are due to the distance between the vehicles being closer than allowed, and Figure 5.7a shows the distance between the master and slave relative to the constraints. Table 5.7 shows the progress of the vehicles at the times of constraint violation. At  $t = 6$ , the slave's progress is higher than that of the master, which means the conditions for activating CS3 are satisfied. The team behavior that follows is in accordance with CS3. Figure 5.7b shows the vehicles' speeds, and a decreasing speed indicated a switch in behavior from path following to formation preservation. The slave stops and waits while the master continues mission execution, which results in the formation being restored and both vehicles taking on the path following behavior. The progress of the vehicles changes significantly before the second constraint violation at  $t = 12$ , and the master now has the higher progress. This time, the conditions for activating CS4 are satisfied, and the master switches behavior as expected, allowing the slave to restore the formation. The master resumes mission execution once the constraint is satisfied again at  $t = 18$ . The formation is upheld for the remainder of the mission, and the vehicles complete mission execution four seconds apart.



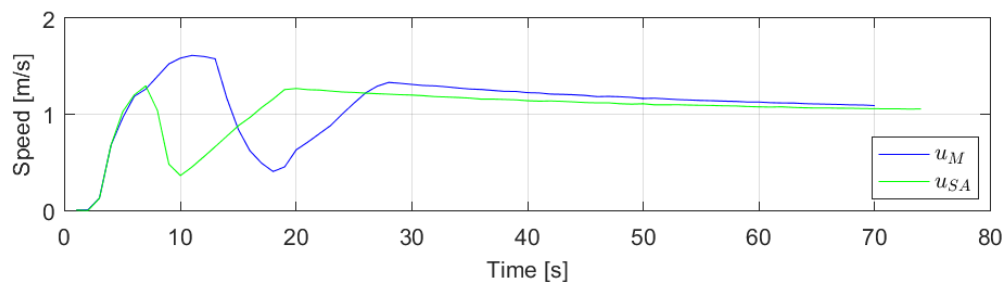
**Figure 5.6:** Vehicle trajectories in Master catches up to slave mission.

**Table 5.7:** Mission progress during execution of Master catches up to slave mission.

Time	Master progress	Slave progress
6 s	17.906 %	19.131 %
12 s	48.538 %	27.337 %



**(a)** Distance between master and slave relative to constraints during execution of Master catches up to slave mission.



**(b)** Vehicle speed during execution of Master catches up to slave mission.

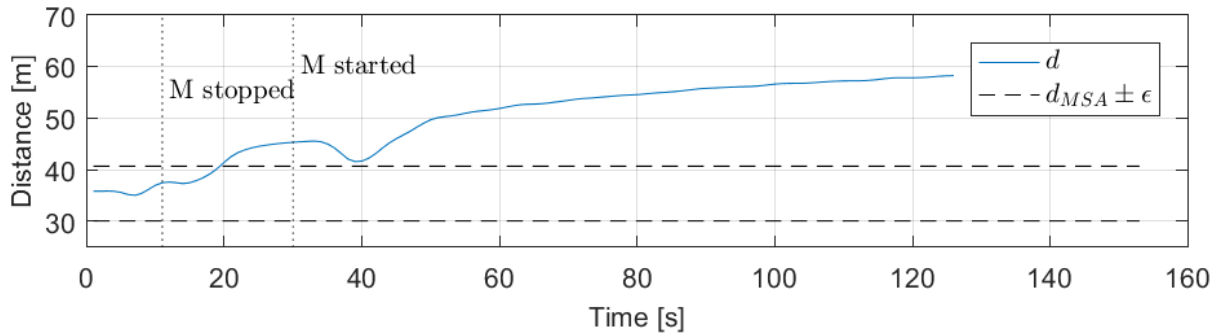
**Figure 5.7:** Speed and distance data from Master catches up to slave mission.

## 5.5 Unsuccessful Mission

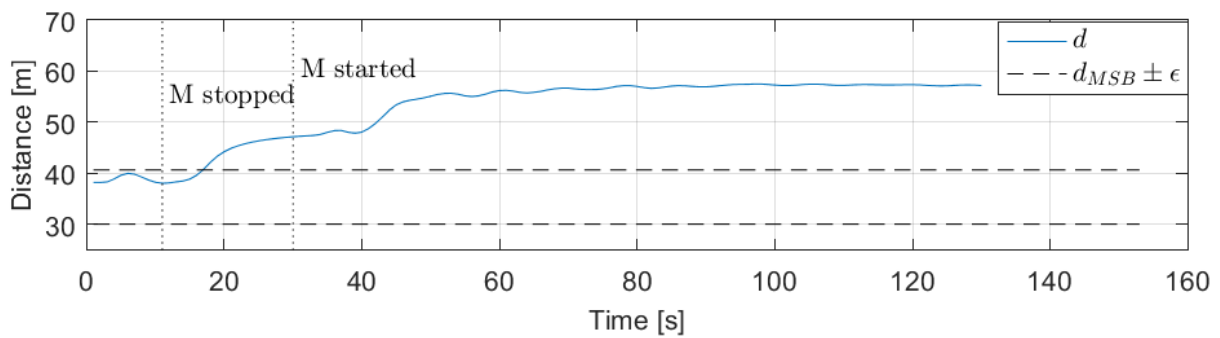
During the simulations, there was a mission conducted by a three-vehicle team that did not go as expected. Details such as formation and mission name are excluded, as relevant results are the behaviors of the vehicle team illustrated through the data in Figure 5.8.

The master was stopped during mission execution to induce a constraint violation at  $t = 11$  and started again at  $t = 30$ . The times of the start and stop are marked in all plots. The forced stop induces a constraint violation for both slaves by  $t = 20$ , as can be seen in Figures 5.8a and 5.8b. The progress of each team member is shown in Figure 5.8d, and at the times of the constraint violation the progress of all three vehicles are nearly the same. However, the master's progress is the lowest, and the conditions for activating CS2 are satisfied. Both slaves are observed to reduce their velocities in Figure 5.8c, indicating that they switched behavior from path following to formation preservation.

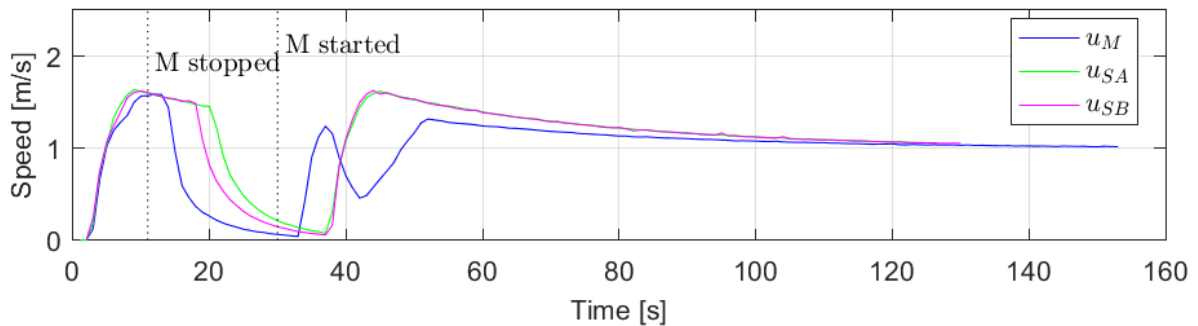
Shortly after the master is started again, its mission progress suddenly increases and surpasses the mission progress of the slaves, causing the conditions for activating CS1 to be satisfied. The master switches behaviors from path following to formation preservation by reducing its speed, and both the slaves switch behaviors from formation preservation to path following. These actions cause the deviation from the desired distance to decrease, and up until now, the team behavior has been by the book. Here on out, that changes. At  $t = 40$  the master no longer has the higher mission progress, causing the conditions for CS2 to be activated once again. The master switches back to path following behavior, and the expected behavior of the slaves would be to switch to formation preservation behavior. However, this is not the case, and both slaves maintain the path following behavior and the deviation from the desired position increases. The conditions for activating CS2 are satisfied for the remainder of the mission, but all team members maintain their past behavior and maneuver the remainder of the path.



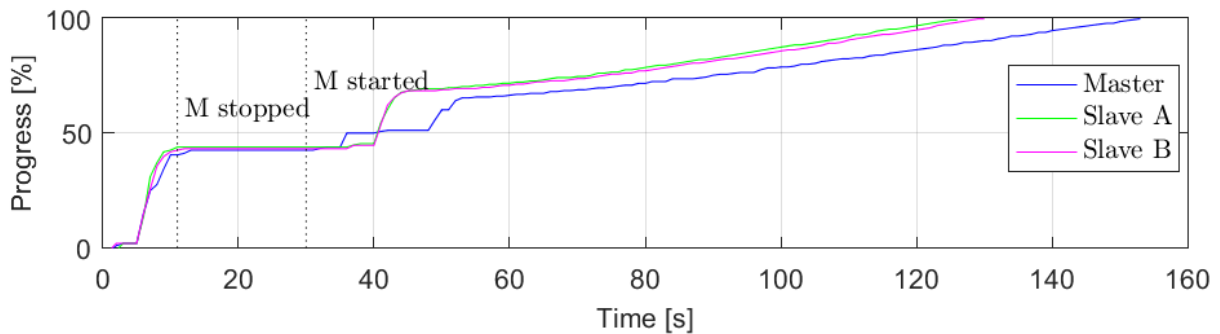
(a) Distance between master and slave A relative to constraints during execution of an unsuccessful mission.



(b) Distance between master and slave B relative to constraints during execution of an unsuccessful mission.



(c) Vehicle speed during execution of an unsuccessful mission.



(d) Mission progress during execution of an unsuccessful mission.

**Figure 5.8:** Speed, distance, and progress data from an unsuccessful mission.

## 5.6 Discussion

The initial discussion of the simulation results disregards the results from the unsuccessful mission presented in Section 5.5, as it is not considered to be representative of the system. A separate discussion of these results is given toward the end of this section.

The simulation results are very promising, and the first part of the discussion will be based on the initial conclusions drawn in Section 3.5.4 during the first verification of the formation control method. The first point to be made was that if the vehicles began mission execution at the same time, and were subject to the same external forces, their mission progress would be identical and the formation would be preserved. A mission without constraint violations has not been presented, but studying the formations before and after constraint violations points to this being true. Master catches up to slave mission, presented in Section 5.4 exemplifies this. The vehicles' trajectories, shown in Figure 5.6, appears to have similar progression along the path from  $t = 40$ , although some minor deviations are present, causing the vehicles to complete mission execution four seconds apart. Depending on the mission objective and mission duration, a four-second delay may very well be negligible.

The second case discussed in the initial verification of the formation control method was the case where a vehicle team was exposed to non-uniform external forces causing a violation of TC1. The four missions presented all demonstrate a violation of TC1, and the results point to the system being able to pick up on constraint violations, correctly identifying the required cooperative strategy, and the vehicles being able to switch behaviors and restore the formation before resuming mission execution.

A weakness in the implementation of the formation control was revealed through the simulations. Figures 5.3a, 5.5a, and 5.7a show the distance plots in the three missions where a constraint is violated and later satisfied, notice the distance between two vehicles relative to the constraints after a violation. As a general rule, once the constraint has been violated and the system is able to restore the formation, the distance between the two vehicles lies close to the previously violated constraint for the remainder of mission execution. The reason for this is that the waiting vehicle resumes mission execution immediately when the constraint is satisfied, meaning that little can be done to decrease or increase the distance as both vehicles are maneuvering from this point on. By changing the implementation of the formation control method, and only allowing a waiting vehicle to resume mission execution once the distance between the two vehicles satisfies a stricter constraint, the vehicle team would have better conditions for completing the mission without further constraint violations. This property was not noticed until after both simulations and field tests were conducted, the code is therefore left as is, and the modification is proposed as further work.

In the introduction to this chapter, it was specified that the presented results are representative of the majority of the simulation results. However, unexpected behavior did occur in a limited number of the simulations, and for the most part, it could be explained by analyzing the mission after it was completed. Note that the case presented in Section 5.5, is not included in this category. It is the author's understanding that it is a combination of factors that contribute to most of the unexpected behavior, all of them present in the presented missions, but at a lower extent and or as isolated incidents, resulting in the system being able to perform as expected.

The first factor that may contribute to an unsuccessful mission or abnormal behavior is that the vehicles drift a lot. Drifting occurs in two scenarios; when the vehicle has reached its initial position and remains idle before mission execution, and when a vehicle switches to the formation preservation behavior, effectively stopping mission execution and remaining idle. The behavior is natural, as the speed of the vehicle does not reach 0 m/s immediately when the propeller stops, but slowly decreases due to friction. Currents in the water will also cause drifting when the vehicle is idle, but this is a note for the field test as the simulations have been run without a current present. A consequence of the vehicles drifting is that the heading obtained before mission execution often changes. If the change is sufficiently large, the process of restoring the correct heading causes the vehicle to deviate from the path, often resulting in a constraint violation. A potential solution to the drifting problem can be to apply a loiter maneuver each time the vehicle is idle, ensuring that it stays close to its position. The loiter maneuver makes a vehicle circle a waypoint with a given radius. It is important to note that the heading is not preserved in this case. Depending on the sea state and spatial distribution of the mission the right way to approach this challenge may vary. The best solution would be to have several approaches and being able to choose before mission execution. One suggestion for an approach has been made, but more work should be put into suggesting additional approaches for limiting or handling drifting.

The second factor contributing to unexpected behavior is the calculation of mission progress. Figure 5.3c shows how the mission progress increases by more than 40 % within a few seconds during mission execution. The large growth of progress over a limited period may cause the wrong cooperative strategy to be activated, as the mission progress does not necessarily give an accurate description of a vehicle's mission progress relative to the other team members. An explanation for the uneven development of the mission progress may be that it is designed for missions of longer duration and that the scale of the simulated missions is too small to get an accurate description of the progress. Longer duration missions have not been conducted to confirm the author's assumption, and if the assumption is wrong, an alternative way of defining the vehicles' progress should be designed to ensure consistency in the system's performance. A possibility could be to calculate the mission progress by taking the length of the path covered



relative to the full path length. In the case of a vehicle deviating from the path, one would use the closest point on the path and consider this as the vehicle's position when calculating mission progress.

During all four missions, the vehicles maneuvered at a higher speed than 1 m/s, and this is brought forward as the third and final factor that may contribute to unexpected system behavior. A prerequisite for the formation control method to work is that the vehicles maneuver at the same speed, and specifying the speed in the mission plan was a way to ensure this. However, as can be seen in Figures 5.2c, 5.3b, 5.5b, and 5.7b the speeds of the vehicles exceed 1 m/s at some point during the mission, if not for the full duration. Despite this, all presented missions have been successful, and the explanation for this is that simulators have the same speed controller producing similar speed profiles for each vehicle, which again causes the vehicles to maneuver the path at approximately the same speed. This might not always be the case when utilizing physical vehicles, and the speed controllers should be checked before mission execution.

The team behaviors observed in the unsuccessful mission presented in Section 5.5 cannot be explained by any of the three factors mentioned above. Although maneuvering at speeds higher than 1 m/s and sudden growth in the mission progress are present, it does not explain why the slaves do not switch behavior as expected. Given that the formation control has picked up on all constraint violations and proven to be successful in previous examples, not to mention that cooperative strategies were activated twice during the mission in question, it is expected that the observed behavior in this mission was due to a bug in the system. Another reason for this assumption is that messages were successfully sent and received during the mission and there is no indication of the system having sent messages that the slaves ignored or did not receive, which means that stop requests were probably never generated. The case was attempted replicated, without success, and the bug has therefore not been identified. It is the author's firm belief that this case does not affect the validity of the formation control or the MCS as the functionality of both systems have been demonstrated in the previously presented missions.

Seeing the results in light of the various challenges related to mission planning, cooperative systems, and formation control discussed in Chapter 2, many of the known challenges were avoided due to the missions being simulated and the simulation environment being free of currents and obstacles. Also, the capabilities of the CCU in the centralized architecture did not limit the systems ability to perform as desired. The PC used as the CCU was able to handle all communications and computation required to plan and execute the missions, while also running the MCS and simulators. By only allowing one of the vehicle's possible behaviors to be active at a time, the possibility of unpredictable things happening due to unusual combinations of behaviors was avoided. The only experienced challenge that can be related to the previously

discussed challenges is the drifting. Drifting was observed in all the missions, and it points to the vehicle's dynamics not being taken sufficiently into account when designing the formation control method. The complications caused by drifting might become even more severe when the system is exposed to a real environment.

To summarize the results confirm that the formation control method is able to pick up on constraint violations, and identify the correct cooperative strategy for restoring the formation. Factors causing unexpected system behavior have presented themselves, but the occurrences have been few and far apart, and suggestions have been made on how to cope with some of the factors. There is believed to be a bug in the system, but the bug has not been identified as the situation revealing the bug has not been possible to replicate. In large, many of the challenges linked to mission planning and cooperative AUV systems were avoided as the missions were simulated and conducted on the surface. Finally, the system has proven to be robust during simulations, making field testing a natural next step in the process of testing the full system, and the MCS and formation control in particular.

## Field tests

Field tests were carried out in the Trondheim Fjord close to the Applied Underwater Robotics laboratory (AUR-lab) on 18th and 20th of April 2018 by the author and researcher Tore Mo-Bjørkelund. The AUR-lab is part of the Institute of Marine Technology at NTNU and is the laboratory that keeps and maintains LAUV Fridtjof. The system was set up according to the description given in Chapter 3 and LAUV Fridtjof operated with default vehicle configurations. The configuration is specified in Appendix D.2.

### 6.1 Organization

#### 6.1.1 Objectives

The objectives of the field tests are formulated below.

- Demonstrate the formation control in a field environment
- Confirm or discredit the findings from the simulation tests.
- Investigate the differences between LAUV Fridtjof and the simulators during mission execution.
- Demonstrate the MCS' ability to function in a field environment.
- Check the system integration between LSTS software and hardware, LAUV Fridtjof and the MCS.
- Detect problems or possibilities regarding further work.

## 6.1.2 Method of Approach

Test missions were conducted with two or three vehicles in the team; LAUV Fridtjof in combination with one or two simulated vehicles. Two-vehicle team missions were conducted on the first day of testing, and three-vehicle team missions on the second day. The sea state on both days can be categorized as relatively calm. There were few waves, and no strong currents were visible at the surface.

The simulators and the MCS ran on one PC, and Neptus on another. Neptus was used to command LAUV Fridtjof into station keeping between missions, and the remaining command and control was executed by the MCS.

The Manta was placed at the edge of the pier outside the AUR-lab, and the acoustic modem was lowered into the water. LAUV Fridtjof was launched from the beach 50 – 60 m from the pier. For safety measures, a rowboat was prepared by the beach in case the vehicle needed to be retrieved.

All missions were conducted on the surface without restrictions on the minimum distance between the vehicles, and all constraint violations occurred naturally without interference from the author. The results from three missions are presented in the following sections. The missions names are based on the shape of the path or the intended test case, and relate to the cooperative strategies in the following manner:

**Section 6.2:** *L mission*; No cooperative strategy activated.

**Section 6.3:** *Stop master Fridtjof mission*; CS3.

**Section 6.4:** *U mission*; CS1, CS4, and CS5.

The chapter concludes with a discussion of the presented results.

## 6.2 L Mission

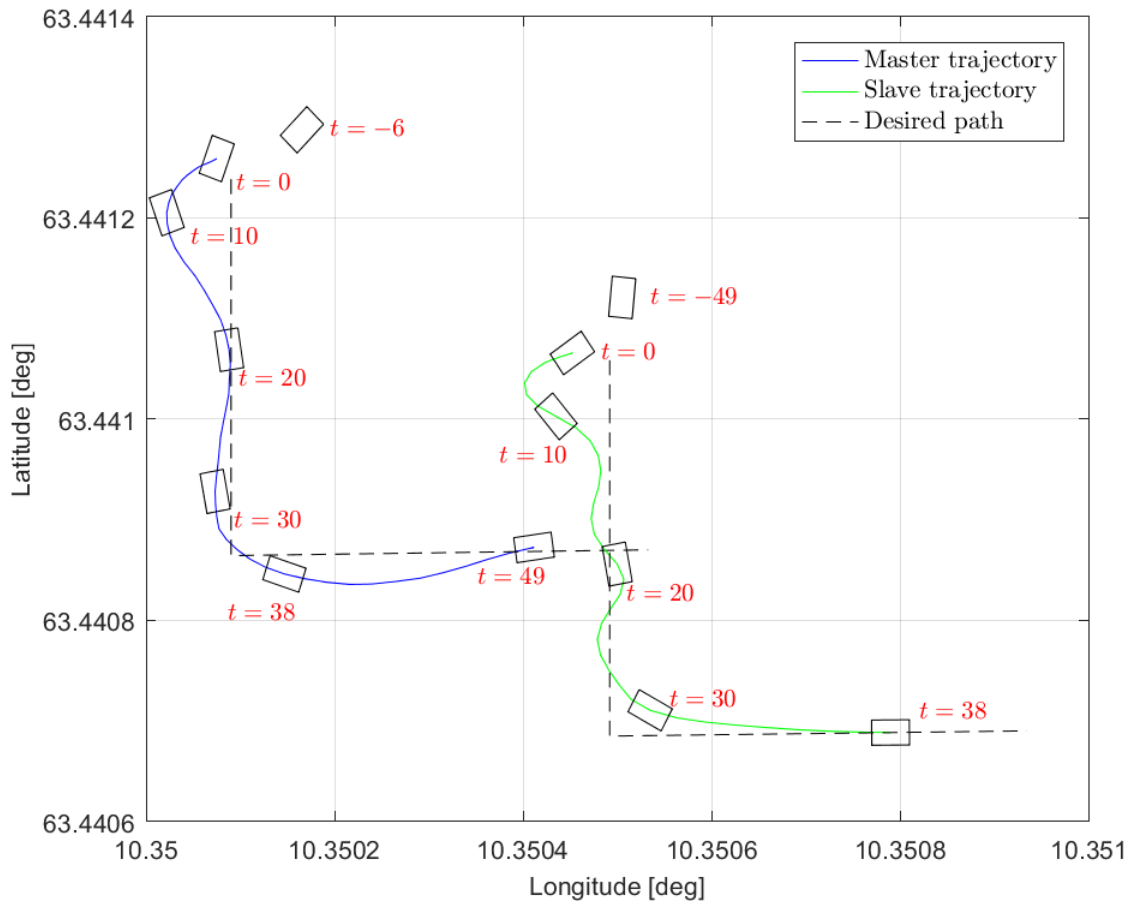
L mission was conducted by a two-vehicle team, and the objective was to maneuver a path in the shape of an L. The mission demonstrates the formation control method without the occurrence of constraint violations. Table 6.1 gives an overview of the vehicle team and formation, note that LAUV Fridtjof is assigned as master. The maximum deviation was set to 30 %, which translates to  $\pm 8.5$  m.

**Table 6.1:** Overview of vehicle team and formation in L mission.

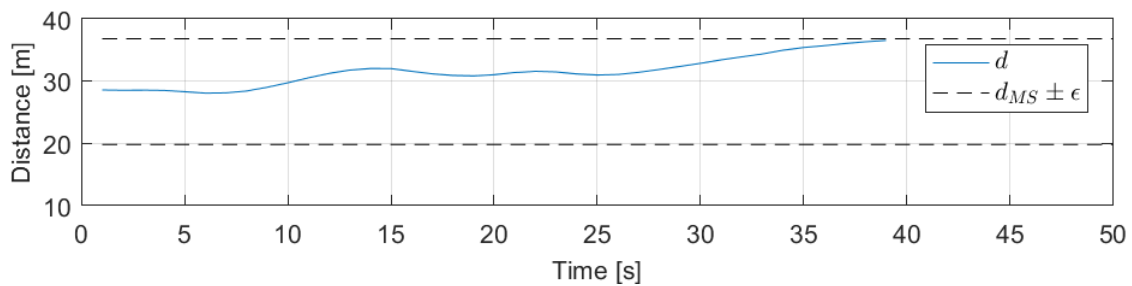
Assignment	Vehicle	$\Delta x$	$\Delta y$
Master	LAUV Fridtjof		
Slave	LAUV Simulator 1	20 m	-20 m

The generated paths and trajectories for both vehicles are shown in Figure 6.1. The slave is the first to reach its initial position at  $t = -49$  and drifts prior to mission execution. The master appears to drift an equal distance from the time when it reaches the initial position at  $t = -6$  until mission execution begins at  $t = 0$ . Both vehicles deviate from the path during mission execution but converge onto the path again, and the formation is kept throughout mission execution. Figure 6.2a shows how the distance between the two vehicles evolve during mission execution. Although the upper constraint is never violated, the distance grows throughout mission execution and is close to violating the constraint when the mission is completed.

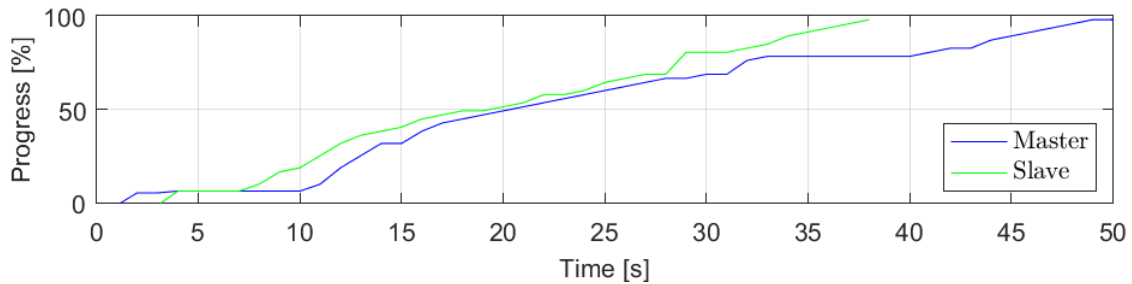
The progress of both vehicles during mission execution is shown in Figure 6.2b. The progress does not grow linearly throughout the mission but has some plateaus where the mission progress is constant over a longer period. For the master these occur at  $t \in [2, 10]$  and  $t \in [33, 39]$ . In the same figure, one can also observe that the slave's progress, in general, is higher than the master's and that the slave completes the mission eleven seconds before the master. The vehicles complete mission execution at  $t = 38$  and  $t = 49$ .



**Figure 6.1:** Vehicle trajectories in L Mission.



**(a)** Distance between master and slave relative to constraints during execution of L Mission.



**(b)** Mission progress for master and slave during execution of L Mission.

**Figure 6.2:** Distance and progress data from L mission.

### 6.3 Stop Master Fridtjof Mission

Stop master Fridtjof mission was conducted with the same vehicle team, formation, and constraints as L Mission and the goal was to maneuver north in vertical parallel lines. The paths and vehicle trajectories are shown in Figure 6.3. At  $t = 0$ , both vehicles have completed the preparation mission where they maneuver to their initial formation, and mission execution begins. Comparing the two vehicles at  $t = 0$  it can be observed that the master deviates further from the initial point than the slave, also the master's heading is off by more than  $90^\circ$  whereas the slave has less of a heading deviation.

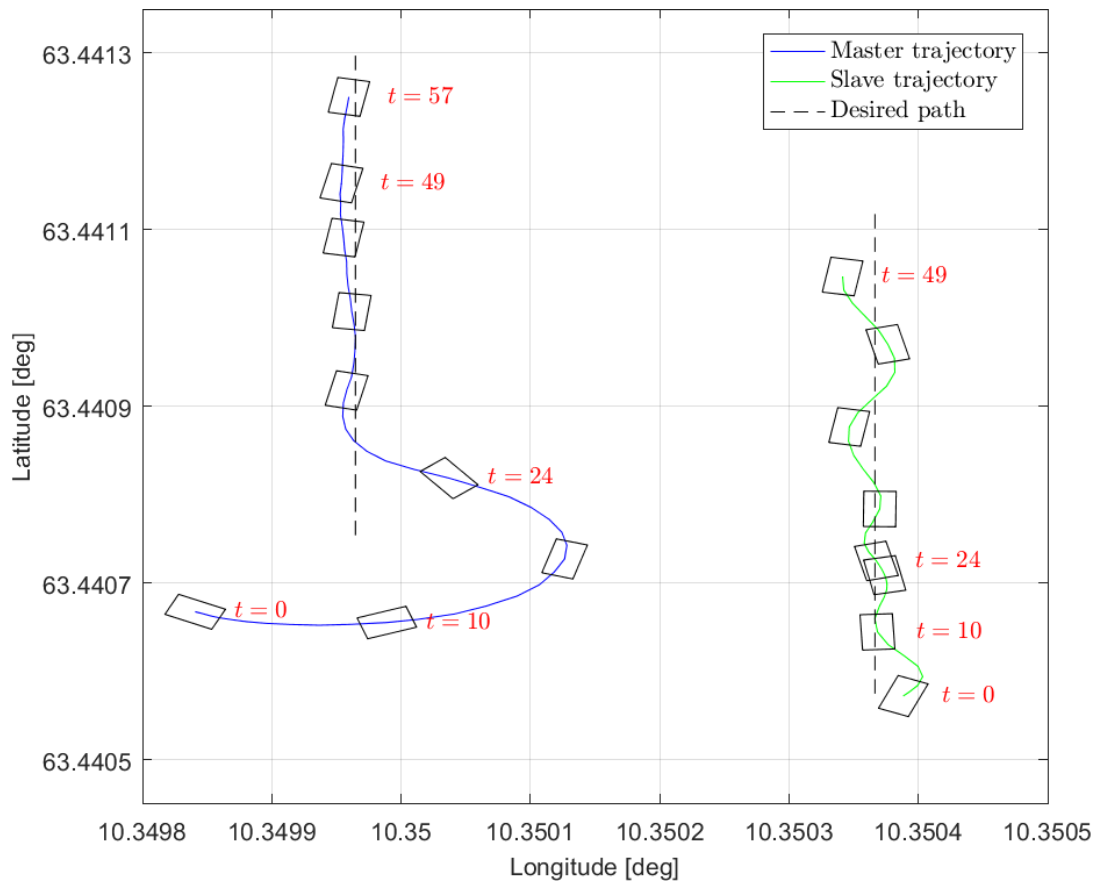
The distance between the vehicles is shown in, Figure 6.4a, and it can be seen that the lower constraint is violated at  $t = 10$ , meaning that the vehicles are closer than allowed. Studying the vehicles' positions at this point, we see that the master appears to be making a turn to maneuver onto the path, while the slave is on the path. Table 6.2 shows the progress of the vehicles at the same time, and the slave appears to have higher mission progress than the master. The conditions for activating CS3 are satisfied, and the team switches behaviors as expected. The master continues with path following, while the slave switches to formation preservation, allowing the master to get back into the formation. The constraint is again satisfied at  $t = 25$ , and the slave resumes mission execution.

No further constraint violations occur during the mission, and both vehicles converge to the path and complete the mission while keeping the formation. Note that although the formation is kept, the vehicles complete mission execution with eight seconds between them, and as can be seen in Table 6.2 there is a significant difference in mission progress at  $t = 49$  moments before the slave completes the mission.

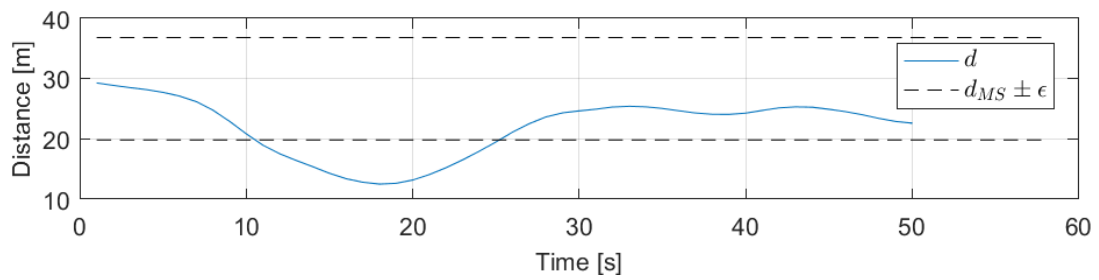
Figure 6.4b shows that both vehicles generally maneuver at a higher speed than 1 m/s which is specified by the MCS throughout mission execution. Also, note that the master has a speed of approximately 0.5 m/s at the start of the mission execution and that speed profiles of the vehicles bear little resemblance.

**Table 6.2:** Mission progress during execution of Stop master Fridtjof mission.

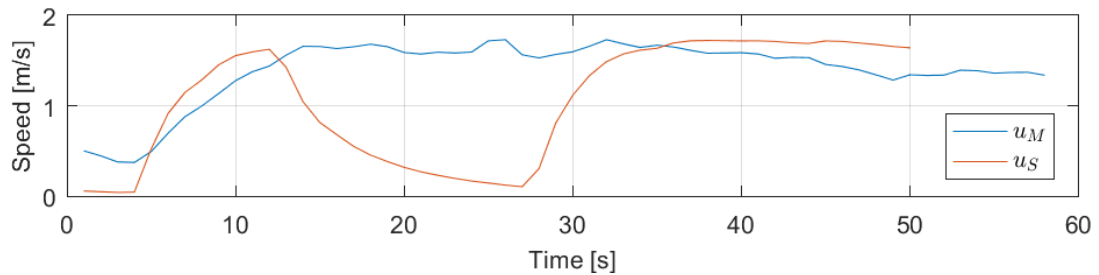
Time	Master progress	Slave progress
10 s	8.894 %	38.848 %
49 s	87.000 %	98.247 %



**Figure 6.3:** Vehicle trajectories in Stop master Fridtjof mission.



**(a)** Distance between master and slave relative to constraints during execution of Stop master Fridtjof mission.



**(b)** Vehicle speed during execution of Stop master Fridtjof mission.

**Figure 6.4:** Speed and distance data from Stop master Fridtjof mission.



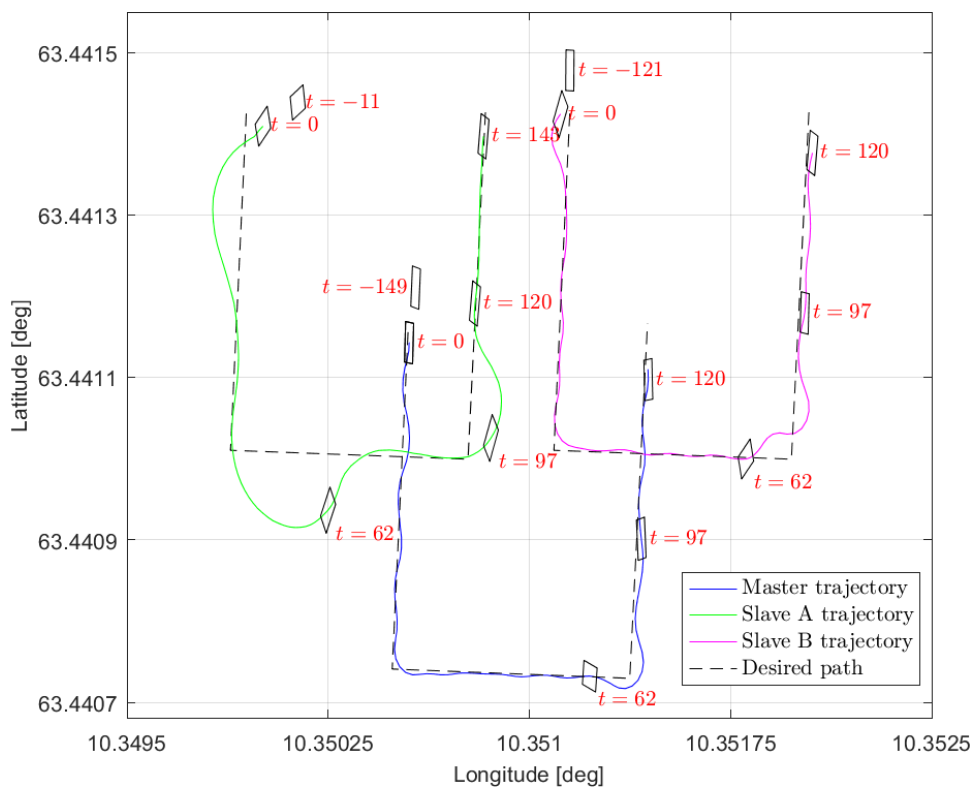
## 6.4 U Mission

U mission was conducted by a three-vehicle team, and the objective was to maneuver a path in the shape of a U. Table 6.3 gives an overview of the vehicle team and formation, note that LAUV Fridtjof is assigned as slave A. The constraint for maximum deviation was set to 30%, which translates to  $\pm 12.7$  m for both slaves.

**Table 6.3:** Overview of vehicle team and formation in U mission.

Assignment	Vehicle	$\Delta x$	$\Delta y$
Master	LAUV Simulator 1		
Slave A	LAUV Fridtjof	-30 m	30 m
Slave B	LAUV Simulator 2	30 m	30 m

The vehicles' paths and trajectories are shown in Figure 6.5, and the vehicles' positions and headings at various times during the mission execution are marked.  $t = 0$  marks the start of the mission execution. The master and slave B are the first to reach their initial positions, at  $t = -149$  and  $t = -121$ , respectively. Slave A reaches its position at  $t = -11$ , and it can be observed that slave, despite a shorter waiting period, drifts further than the remaining team members.



**Figure 6.5:** Vehicle trajectories in U mission.

Figure 6.6 shows the distance between the master and two slaves relative to the constraints, and the vehicles speeds. The distance between the master and slave B (Figure 6.6b) is relatively stable and satisfies the constraints throughout mission execution. The distance between the master and slave A (Figure 6.6a) varies more and violates the constraint twice during the mission. Table 6.4 shows the mission progress of each vehicle at various times during mission execution, including the times when the constraints are violated.

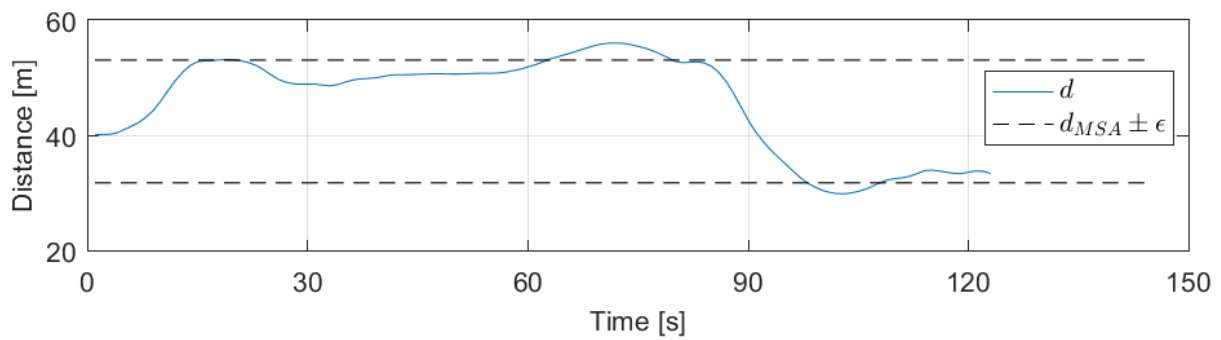
**Table 6.4:** Mission progress during execution of U mission.

Time	Master progress	Slave A progress	Slave B progress
62 s	61.669 %	47.536 %	60.883 %
97 s	83.627 %	70.95 %	85.375 %
120 s	99.335 %	81.942 %	99.835 %

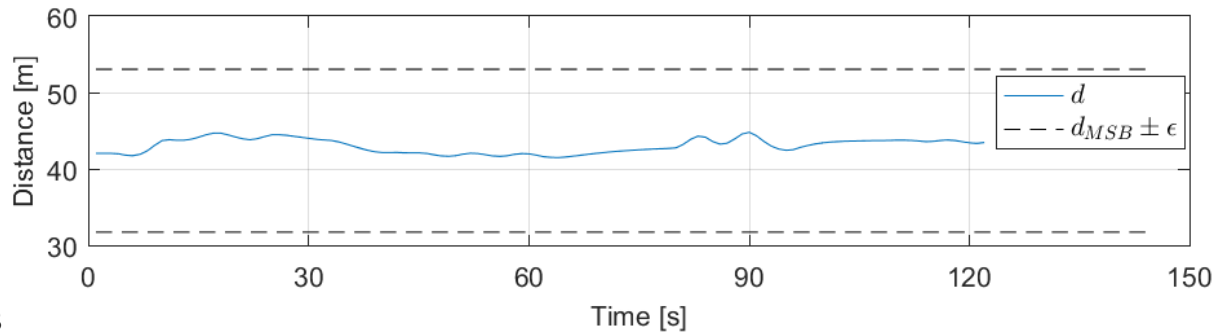
The first constraint violation occurs at  $t = 62$  after slave A maneuvers past the first  $90^\circ$  angle in the path. One can observe in Figure 6.5 that the master and slave B maneuver past the corner without significant deviation, however, slave A deviates significantly from the path and the distance between the master and slave A grows greater than the constraints allow. At this point, slave A's mission progress is lower than the master's, and the conditions required for activating CS1 are satisfied. The master and slave B switch behaviors at  $t = 63$ , stopping mission execution and waiting. Slave B stops due to CS5, which is activated when the master needs to stop. Slave A get back into the formation, and when the constraints are again satisfied at  $t = 79$ , the master and slave B switch back to path following behavior.

At  $t = 97$  the second constraint violation occurs, once again it is when slave A is maneuvering past a corner. This time the master and slave A are too close to each other, and the master's progress is still higher, satisfying the conditions for activating CS3. It can be observed from the speed plot (Figure 6.6c) that the master and slave B, once again, switch behavior from path following to formation preservation, and wait for approximately ten seconds until the constraint is no longer violated. They then resume mission execution.

The distance between the master and slave A continues to lie close to the lower constraint, but the remainder of the mission is completed without constraint violations. At  $t = 120$  the master and slave B are moments from completing the mission while slave A hangs behind. The progress at this time is given in Table 6.4, and shows that slave A's progress is significantly lower than the remaining team members' progress'. Slave A completes the mission at  $t = 143$ , more than twenty seconds behind the rest of the team.

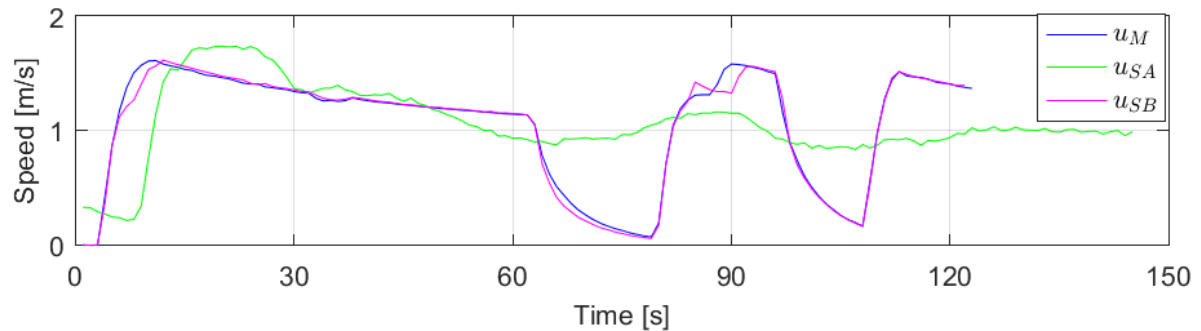


(a) Distance between master and slave A relative to constraints during execution of U mission.



B

(b) Distance between master and slave B relative to constraints during execution of U mission.



(c) Vehicle speed during execution of U mission.

**Figure 6.6:** Speed and distance data from U mission.

## 6.5 Discussion

The results from the field tests are promising and support the findings in the simulations and initial conclusions drawn in Section 3.5.4 about the formation control method. L mission demonstrated the case of a mission without constraint violations. The vehicles started mission execution at the same time and managed to keep the formation and complete the mission. In the two remaining missions, all violations of TC1 were detected and handled by a cooperative strategy, which resulted in the formation being restored. CS2 was not tested, but based on the available results from the simulations and field tests, one can assume that the case would not be any different for this cooperative strategy in particular. Hence, the formation control method has proven able to control the formation of a vehicle team in both a simulated and real environment.

To ensure that the verification procedure used to test the formation control is valid the characteristics and performance of the simulated vehicles and LAUV Fridtjof have been compared. Too great of a difference between the entities might have affected the results in a way that makes any conclusions drawn about the formation control method's validity and capabilities only true for these cases. The first thing to become apparent during field tests was the swing radius of LAUV Fridtjof. Whereas the simulators made smooth turns and were able to maneuver relatively sharp corners, LAUV Fridtjof was not as elegant. It is exemplified in all three missions, but Figure 6.5 depicting the vehicles' trajectories during U mission very clearly shows the difference. The difference in swing radius does not affect the test of the system. It merely introduced additional constraint violations which were handled.

Another thing that was observed to differ between the simulators and LAUV Fridtjof was the acceptance radius of a waypoint. Trajectory plots from all three missions show that LAUV Fridtjof is further away from the initial point when compared with the slaves after the preparation mission is completed. The difference is most evident in Stop master Fridtjof mission, Figure 6.3, where the positions of both vehicles deviate from the initial point at  $t = 0$ , but LAUV Fridtjof has the most substantial deviation. The difference in acceptance radius does not affect the performance of the formation control method, and the fact that all missions were successful attest to the robustness of the method. However, checking the guidance controllers prior to mission execution in the future and setting equal acceptance radius' could improve the performance even further.

Drifting was observed during simulation, and it is clear that the issue is also present during field tests. It is pointed out in the results from L mission that LAUV Fridtjof drifts more than the simulators. The drifting is most likely due to the currents and external forces that LAUV Fridtjof is exposed to. Despite the water being relatively calm the results show that some drifting is to be expected, and accentuates the importance of further studying the need to implement

functionality to counteract the drifting.

The communication between LAUV Fridtjof and the MCS was routed through the Manta, and the author expected that this might introduce some delays. However, the presented results neither confirm or deny this assumption. In Stop master Fridtjof mission, Figure 6.4b shows that both the simulator and LAUV Fridtjof start accelerating at the same time, indicating they received a start request at approximately the same time. The results from U mission, on the other hand, seen in Figure 6.6c, point to LAUV Fridtjof's acceleration beginning at a considerably later point in time, which might indicate that the vehicles did not receive the start request at the same time. The observed delay was not decisive for the mission execution, and the results seem unaffected by a potential delay. Concluding on whether or not there is a delay between when a simulated and real vehicle receives a message sent from the system is not possible based on these two cases, and further tests should be conducted. There is a possibility that communication delays with greater significance are introduced when the scale of the mission grows, i.e., the vehicle maneuvers further away from the Manta.

In the comparison of the simulators and LAUV Fridtjof, several properties seemed to be similar. The progress of the vehicles in L mission is shown in Figure 6.2b, and as can be observed they evolve in the same way with plateaus and growth spurts. It was previously pointed out that the way the progress evolves is not beneficial for the formation control method, and the author wishes to once again emphasize the importance of testing missions with greater spatial distribution and considering an alternative way of calculating mission progress.

Another property that the simulators and LAUV Fridtjof shared was the high speed. The formation control method is widely based on the vehicles dynamics being similar and the vehicles maneuvering the path at the same speed. Figures 6.4b and 6.6c show that LAUV Fridtjof also maneuvers at a higher speed than 1 m/s, but is able to converge when left to maneuver over more extended periods of time. An explanation for the difference in the speed profiles is, of course, the forces LAUV Fridtjof is exposed to during mission execution that are not present in the simulated environment. However, the difference does not seem to affect the performance of the system. What would be interesting to see is how the speed profiles of two or more real vehicles compare, as it would be deciding for the formation control method if the team members are not able to hold approximately the same speed.

Based on the observations made during the comparison of the simulators and LAUV Fridtjof, two modifications are suggested to ensure better system performance and less deviation between the two entities performance during mission execution. The implementation of the allowed deviation was discussed during the verification of the MCS, but L mission underlines once again the importance of the user being able to explicitly set the allowed deviation. Figure 6.1 shows

a formation that is very skewed towards the end of the mission although still satisfying the constraints. The second suggested modification is to the path generation. U mission demonstrates that a physical vehicle is not able to maneuver, the generated path without significant deviations. Given the current restrictions set on the cooperative strategies, a slave could be left behind because the defined path is infeasible. Path generation has not been a central part of this thesis, and therefore the behavior of LAUV Fridtjof was somewhat expected. By smoothing the path and making it feasible for a real vehicle to maneuver the number of constraint violations will reduce.

The field test results introduced a potential challenge of communication delays, that was not present during simulation, but except for this no new challenges were revealed. The reason for the low number of complications and challenges experienced is the fact that the missions are conducted on the surface and communication being enabled by Wi-Fi. Conducting tests on the surface was an active choice made to ensure that the work of this thesis could include field tests. If tests were to be conducted subsea, several challenges would have to be addressed and handled beforehand; the most important being to implement functions for sending and receiving acoustic messages, and to limit the information passed between the vehicles and the CCU. Currently, the IMC messages are not stripped for information that is not essential for mission execution. However, new IMC messages may be generated and passed through acoustics. Functionality for re-sending messages and handling lost messages must be implemented as well.

In addition to testing the formation control method, the field tests gave a chance to test the full system in a real environment with the software components running on separate hardware. All the missions were successfully conducted, which attests to all the communication links being consistent during the tests. The results point to the MCS being robust in a real environment, and no changes to the application's performance were detected when additional system components, i.e., Neptus and the Manta, and a real vehicle was introduced. Two days in the field without any unexpected behavior, loss of communication or system failure points to the system integration between all software and hardware components working as it should.

To summarize, the field tests paint a picture of an application and a formation control method that are robust in a real environment and able to plan, control and monitor missions and manage the formation of a cooperative AUV system that consists of both real and simulated vehicles. Most of the challenges associated with mission planning have been worked around by running tests on the surface and using Wi-Fi for communication. Some deviation between the properties of the simulated and real vehicles have been detected, and suggestions have been made on how to enhance the system's performance.

## Discussion

To recap, an application for operating a cooperative AUV system with mission planning and formation control capabilities has been developed. The formation control method was designed based on existing formation control approaches. Tests to verify the application and formation control were conducted through simulations and experiments in the field. The application proved robust in both environments and was able to operate both as a standalone application and as an integrated part of a greater ecosystem of software. The formation control also proved successful in both test environments. Results from simulations and field tests are discussed in Sections 5.6 and 6.5, respectively, and the latter also makes a comparison of the results.

In this chapter, the discussion addresses the uncertainties in the results and the experimental verification method. A general discussion on the formation control follows before the chapter concludes with a discussion of the mission planning demonstrated in this thesis and the MCS.

### 7.1 Uncertainties

Uncertainties influencing the formation control have been identified. Some of them are mentioned in Section 5.6 and a general overview of the uncertainties follows.

Despite the calm sea on the days of field testing, the results show that LAUV Fridtjof's performance was affected by external forces. The simulation environment is free of currents, and this caused a significant difference in the performance of the simulators and LAUV Fridtjof during mission execution, i.e., LAUV Fridtjof drifted a lot, and deviated greatly from the path when sharp corners were maneuvered. The formation control was able to compensate for Fridtjof's performance, but it is uncertain how efficient the formation control is at controlling a system when multiple vehicles are deployed in a sea state similar to the state during field tests, or a single vehicle is deployed in large currents.

Applications for vehicle teams that require formation control are usually conducted over a much larger area than the test cases were. The system's ability to detect constraint violations and the method's ability to restore the formation will most likely remain unchanged when the scale is increased, but there are uncertainties related to communication over a larger area, and the thirty-second maximum wait period for a master. Tests over a larger area should be conducted to check if the waiting period should be extended and if the formation control is robust against a potential delay in the communication.

The final uncertainty relates to measurement given by LAUV Fridtjof. Calibration of equipment was not conducted prior to the tests, making it possible that some of the measurements used in the results are inaccurate. However, the verification of the formation control does not rely on accurate measurements, but rather how the system responds based on measurements and processed data. This uncertainty is therefore deemed not to affect the validity of the results presented.

Overall, the detected uncertainties demonstrate the need for further and more varied testing of the formation control.

## 7.2 The Experimental Verification Procedure

The experimental verification procedure was designed to ensure that experiments could be conducted despite a single vehicle being at the author's disposal. The research on cooperative AUV systems and formation control conducted in this thesis did not yield any examples of other experiments being conducted in the same manner.

The reasons for this may be many, but from the author's point of view that this is a useful approach when testing new theories and functionality for cooperative systems. The approach may be regarded as a *fail-fast* approach where the goal is to test a theory or new idea at an early stage, regardless of the whole framework and all the details being in place, to confirm or disprove the viability of the theory or idea. For this thesis, that meant that experiments could be conducted to test the developed cooperative strategies before collision avoidance was implemented. By deploying a single vehicle in the fjord and running two simulators on the PC, the risk of collision was eliminated, while the team behavior could be studied and the implemented parts of the formation control verified. An additional advantage of the procedure is that fewer resources were required; only one vehicle needed to be booked, and the experiments were easily conducted by two people.

The MCS does not differ between a simulated and real vehicle, and introducing the LAUV into the system, therefore, became trivial. No modifications were required and once all system



components were set up and connected the components integrated as expected.

As previously pointed out in the discussion of the field test results, the field tests revealed things that were not apparent in the simulations. Although, safety measures would have been taken before field tests with multiple vehicles, properties like the unexpectedly large swing radius may be dealt with at an early stage by modifying the generated paths.

Compensating for shorter communication routes by adding delays on the messages sent to and from the simulators should be considered if a system using acoustic communication or with a large spatial distribution is to be tested.

## 7.3 Formation Control

The greatest challenge of designing the formation control method was the underactuation of the vehicles. Several of the reviewed approaches to formation control required explicit control of the vehicle's velocity vector, but for an AUV it is only possible to explicitly set the forward speed  $u$ . Therefore, combining different approaches to produce a feasible method became the solution. Another point to make in relation to the underactuation of the AUVs is that the designed method is based on the vehicles stopping and holding their position at certain times before and during mission execution. However, when controlling an AUV, stopping the vehicle does not mean the position is held, and the underactuation makes dynamic positioning, i.e., the vehicle holding its position despite external disturbances, difficult. The drifting experienced during testing is a reflection of this.

The results presented are based on missions where the control algorithm was loop five times per second, and the vehicles conveyed a large amount of information. It would be interesting run tests where the control algorithm is looped at a lower frequency to check system performance, as this might be the case for larger scaled missions, due to restrictions on the communication.

During the thesis, it has become evident that a significant reason for the lacking literature on experiments and practices with formation control of AUVs is the communication barrier. The developed method is applicable in three dimensions, but testing in three dimensions is not possible unless acoustic communication is implemented. As this has not been fully implemented for the LAUVs, and it was too large a task to include in the thesis, the tests were conducted on the surface.

To summarize the discussion on the formation control, many of the suggestions for improvement of the method are related to the implementation of the formation control method in this thesis. The method itself and the concepts presented in Section 3.5 have proven to be sufficient for

controlling the formation of a cooperative system in the surface. Nothing can be said for the validity of the method subsea before tests have been conducted.

## 7.4 Mission Planning and the MCS

Choosing the same language as the LAUV uses for on board control for mission planning was an advantage. Although the available documentation on IMC messages lacked some information, the fact that it was XML based made it possible to understand how to compose the messages the correct way by studying old mission logs. Using IMC messages also eliminated the need for a translator, which again limited the number of components where an error could occur.

Focus during development was on creating a well-functioning user interface, and a choice was therefore made to avoid complex mission planning components. This resulted in a very basic mission planner without decision making or complex computations. Observations made during the field tests indicate that the path generation should be made more advanced in order to generate feasible paths. It is important to note that infeasible paths would have occurred despite choosing another approach for generating the mission plan and that the mission plan may continue to be generated directly by a human operator when the path generation is made more advanced. Despite the generated paths not being optimal, the mission planning capabilities of the MCS have been proven to be satisfactory through the conducted tests.

Much work has been put into developing the user interface and the functionality of the MCS, and it has been developed in modules to allow for expansion of additional system functionality. This thesis prioritized implementing functionality that would work both standalone and together with Neptus, but if the MCS is considered solely as an addition to Neptus, the possibilities for controlling cooperative systems or AUVs in general grows. By taking advantage of the offerings of Neptus, and using the MCS to implement additional functionality, the time from an idea is developed until tests can be conducted is considerably limited. Suggestions to improve the existing functionality has been made throughout the thesis, but the most exciting developments for the MCS, and applications like it, in the future will probably be in relation to exploring how it can contribute with additional functionality to existing control and command software, like Neptus.

## Conclusions and recommendations for further work

In the final chapter, conclusions are drawn based on the all the work presented in the previous chapters, and recommendations for further work are given.

### 8.1 Conclusions

This masters thesis has presented the MCS, an application for controlling a cooperative system of LAUVs with mission planning and formation control capabilities. The application is a part of a system for operating cooperative AUV systems that consists of software and hardware from the LSTS. The MCS can function as the only command and control software in the system or in cooperation with Neptus, the command and control software offered in the LSTS-toolchain.

Verification and testing of the MCS, show that the mission planning capabilities of the application satisfy the application requirements and needs of a cooperative system. Although, the field tests revealed that the path generation should be made more complex to obtain paths that are better suited for physical vehicles.

The formation control capabilities in the MCS are enabled by a formation control method with properties from leader-follower systems, virtual structures, and behavior-based formation control. The combination of properties from different types of existing formation control proved successful for cooperative LAUVs, and results from simulations and field tests show that the vehicle team's formation was successfully maintained during mission execution. The experimental verification method allowed for testing before the for the formation method was completed, and collision avoidance should be designed and implemented before experiments with multiple physical vehicles are to be conducted.

The MCS has proved robust during both simulations and field tests, and the integration with the software and hardware in the LSTS-toolchain was effortless. By considering the additional requirements and suggestions for altered implementation given in this thesis, the application's performance may be further enhanced.

## 8.2 Recommendations for Further Work

Before any further work can take place, cooperative strategies for handling violations of TC2, i.e., two team member being closer than a set safety distance, must be designed.

The formation control method can be regarded as a separate entity, independent of the MCS. Once completed, the formation control method provides an approach for formation control that covers how to define a formation, generate paths for slaves and controlling team members behaviors during mission execution. Stricter constraints for a waiting vehicle to resume mission execution in order to better preserve the formation have been proposed. Beyond this point, further work could take two main directions. One is more research on the formation control method using cooperative AUV systems. Research could include implementing the method in another framework for other types of AUVs, field experiments with more than one LAUV using the MCS, or testing the method subsea. The other direction the work can be taken is into other multi-vehicle fields. The formation control can be tested on cooperative systems consisting of ground mobile robot or aerial vehicles.

The cause of the results in the unsuccessful simulated mission is not known as this thesis concludes. If the current implementation of the formation control in the MCS is to be used in the future, this should be further investigated. Regardless of which framework the formation control is implemented in, the suggested improvements discussed in Sections 5.6 and 6.5, should be taken into consideration. The suggestions are listed in the following:

- Evaluation of the need for implementing a maneuver to keep a waiting vehicle close to a waypoint through further simulation and field tests.
- Evaluation of the need for calculating mission progress in an alternative way, rather than using data from the plan control state message from the vehicle.

Further work on the MCS was suggested in the additional requirements derived in Section 4.4. Regarding the two requirements that were not attained in the initial requirements list, it is strongly suggested that R13 be implemented. The ultimate goal is to be able to use the MCS for cooperative systems subsea, and therefore the importance of a stable and reliable communication link remains. It was also suggested after the field tests that the path generation be made more advanced to ensure feasible paths. Although the MCS has proved to work as a

---

standalone application, the author recommends that the application be used as an addition to the LSTS toolchain. The reason for this being that the application can be used as a framework to implement and test other formation control methods, functions for path generation, or other features related cooperative systems or single AUVs running on DUNE. This way the time between designing a method and field testing may be reduced because the remaining system components can ensure a safe test environment and basic functionality that is time-consuming to implement.



# Bibliography

*About mapz* (n.d.). Available at:

<https://www.mapz.com/en/about> (Accessed: 14.05.2018).

AUR-lab (n.d.), *LAUV Fridtjof*. Available at:

<https://www.ntnu.edu/aur-lab/lauv-fridtjof> (Accessed: 07.05.2018).

Balch, T. and Arkin, R. C. (1998), Behavior-based formation control for multirobot teams, *IEEE Transactions on Robotics and Automation* 14(6). pp. 926–939. doi: 10.1109/70.736776.

Beard, R. W., Lawton, J. and Hadaegh, F. Y. (2001), A coordination architecture for spacecraft formation control, *IEEE Transactions on Control Systems Technology* 9(6). pp. 777–790. doi: 10.1109/87.960341.

Bibuli, M., Bruzzone, G., Caccia, M., Ranieri, A. and Zereik, E. (2015), Multi-vehicle cooperative path-following guidance system for diver operation support, *IFAC-PapersOnLine* 48(16). 10th IFAC Conference on Manoeuvring and Control of Marine Craft MCMC 2015, pp. 75 – 80. doi: 10.1016/j.ifacol.2015.10.261.

Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A. and ar, Y. S. (2012), Synthesis of Reactive(1) designs, *Journal of Computer and System Sciences* 78(3). In Commemoration of Amir Pnueli, pp. 911 – 938. doi: 10.1016/j.jcss.2011.08.007.

Breivik, M., Hovstein, V. E. and Fossen, T. I. (2008), Ship formation control: A Guided Leader-Follower Approach, *IFAC Proceedings Volumes* 41(2). 17th IFAC World Congress, pp. 16008 – 16014. doi: 10.3182/20080706-5-KR-1001.02706.

Buadu, S. (2017), *Mission Planning for AUV*, Project thesis, Norwegian University of Science and Technology. Unpublished.

---

*Calculate distance, bearing and more between Latitude/Longitude points* (n.d.). Available at:  
<http://www.movable-type.co.uk/scripts/latlong.html>  
(Accessed: 08.05.2018).

Community, Q. (n.d.), *About Qt*. Available at:  
[http://wiki.qt.io/About\\_Qt](http://wiki.qt.io/About_Qt) (Accessed: 10.05.2018).

Cosic, A., Susic, M., Graovac, S. and Katic, D. (2013), An Algorithm for Formation Control of Mobile Robots, *Serbian Journal of Electrical Engineering* 10(1). pp. 59–72. doi: 10.2298/S-JEE1301059C.

Cui, R., Ge, S. S., How, B. V. E. and Choo, Y. S. (2010), Leader–follower formation control of underactuated autonomous underwater vehicles, *Ocean Engineering* 37(17). pp. 1491 – 1502. doi: 10.1016/j.oceaneng.2010.07.006.

Eckstein, S., Glotzbach, T. and Ament, C. (2013), Towards innovative approaches of team-oriented mission planning and mission languages for multiple unmanned marine vehicles in event-driven mission, 2013 MTS/IEEE OCEANS - Bergen, pp. 1–8.

Eichhammer, E. (n.d.), *QCustomPlot*. Available at:  
<http://qcustomplot.com/index.php/introduction> (Accessed: 11.05.2018).

Essaouari, Y. and Turetta, A. (2016), Cooperative underwater mission: Offshore seismic data acquisition using multiple autonomous underwater vehicles, 2016 IEEE/OES Autonomous Underwater Vehicles (AUV), pp. 435–438.

FFI (2013), *HUGIN World Class Military AUV*. Available at:  
<https://www.ffi.no/no/Publikasjoner/Documents/FFI-fakta%202022-sidig%20eng%20HUGIN.pdf> (Accessed: 07.10.2017).

Fiorelli, E., Leonard, N. E., Bhatta, P., Paley, D. A., Bachmayer, R. and Fratantoni, D. M. (2006), Multi-AUV Control and Adaptive Sampling in Monterey Bay, *IEEE Journal of Oceanic Engineering* 31(4). pp. 935–948. doi: 10.1109/JOE.2006.880429.

Fletcher, B. (2000), UUV Master Plan: a Vision for Navy UUV Development, OCEANS 2000 MTS/IEEE Conference and Exhibition, Vol. 1, pp. 65–71.

Glotzbach, T., Eckstein, S. and Ament, C. (2015), An Approach for Planning a Safe Mission Begin and End for Teams of Marine Robots, *IFAC-PapersOnLine* 48(2). 4th IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles NGCUV 2015, pp. 100 – 106. doi: 10.1016/j.ifacol.2015.06.017.



---

González, J., Masmitjà, I., Gomáriz, S., Molino, E., del Río, J., Manuel, A., Busquets, J., Guerrero, A., López, F., Carreras, M., Ribas, D., Carrera, A., Candela, C., Ridao, P., Sousa, J., Calado, P., Pinto, J., Sousa, A., Martins, R., Borrajo, D., Olaya, A., Garau, B., González, I., Torres, S., Rajan, K., McCann, M. and Gilabert, J. (2012), AUV Based Multi-vehicle Collaboration: Salinity Studies in Mar Menor Coastal Lagoon, *IFAC Proceedings Volumes* 45(5). 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, pp. 287 – 292. doi: 10.3182/20120410-3-PT-4028.00048.

Henriksen, E. H. (2014), *ROV Control System for Positioning of Subsea Modules*, Master's thesis, Norwegian University of Science and Technology. Available at: <https://brage.bibsys.no/xmlui/handle/11250/239244> (Accessed: 10.12.2017).

Holsen, S. A. (2015), *DUNE: Unified Navigation Environment for the REMUS 100 AUV*, Master's thesis, Norwegian University of Science and Technology. Available at: <https://brage.bibsys.no/xmlui/handle/11250/2350792> (Accessed: 10.12.2017).

Inc., T. (n.d.), *C++ Programming Language*. Available at: <https://www.techopedia.com/definition/26184/c-programming-language> (Accessed: 10.05.2018).

Jacobi, M. (2015), Autonomous inspection of underwater structures, *Robotics and Autonomous Systems* 67(Supplement C). *Advances in Autonomous Underwater Robotics*, pp. 80 – 86. doi: 10.1016/j.robot.2014.10.006.

Jia, Q. and Li, G. (2007), Formation Control and Obstacle Avoidance Algorithm of Multiple Autonomous Underwater Vehicles(AUVs) Based on Potential Function and Behavior Rules, 2007 IEEE International Conference on Automation and Logistics, pp. 569–573.

Kalwa, J. (2010), Final results of the european project grex: Coordination and control of cooperating marine robots, *IFAC Proceedings Volumes* 43(16). 7th IFAC Symposium on Intelligent Autonomous Vehicles, pp. 181 – 186. doi: 10.3182/20100906-3-IT-2019.00033.

Kalwa, J., Pascoal, A., Ridao, P., Birk, A., Eichhorn, M., Brignone, L., Caccia, M., Alves, J. and Santos, R. (2012), The European R&D-Project MORPH: Marine robotic systems of self-organizing, logically linked physical nodes, *IFAC Proceedings Volumes* 45(27). 9th IFAC Conference on Manoeuvring and Control of Marine Craft, pp. 226 – 231. doi: 10.3182/20120919-3-IT-2046.00039.

- 
- Karimanzira, D., Jacobi, M., Pfuetzenreuter, T., Rauschenbach, T., Eichhorn, M., Taubert, R. and Ament, C. (2014), First testing of an AUV mission planning and guidance system for water quality monitoring and fish behavior observation in net cage fish farming, *Information Processing in Agriculture* 1(2). pp. 131 – 140. doi: 10.1016/j.inpa.2014.12.001.
- Kongsberg (2016), *Autonomous Underwater Vehicles AUV/Marine Robots*. Available at: <https://goo.gl/uiN4Gs> (Accessed: 04.11.2017).
- Kothari, M., Pinto, J., Prabhu, V. S., Ribeiro, P., de Sousa, J. B. and Sujit, P. (2012), Robust Mission Planning for Underwater Applications: Issues and Challenges, *IFAC Proceedings Volumes* 45(5). 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, pp. 223 – 229. doi: 10.3182/20120410-3-PT-4028.00037.
- Lapierre, L., Soetanto, D. and Pascoal, A. (2003), Coordinated motion control of marine robots\*, *IFAC Proceedings Volumes* 36(21). 6th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2003), Girona, Spain, 17-19 September, 1997, pp. 217 – 222. doi: 10.1016/S1474-6670(17)37810-2.
- Lewis, M. A. and Tan, K.-H. (1997), High Precision Formation Control of Mobile Robots Using Virtual Structures, *Autonomous Robots* 4(4). pp. 387–403. doi: 10.1023/A:1008814708459.
- Li, X., Zhu, D. and Qian, Y. (2014), A Survey on Formation Control Algorithms for Multi-AUV System, 2. pp. 351–359. doi: 10.1142/S2301385014400093.
- Lin, Y., Hsiung, J., Piersall, R., White, C., Lowe, C. G. and Clark, C. M. (2017), A Multi-Autonomous Underwater Vehicle System for Autonomous Tracking of Marine Life, *Journal of Field Robotics* 34(4). pp. 757–774. doi: 10.1002/rob.21668.
- LSTS (2013a). Available at: <https://github.com/LSTS/dune> (Accessed: 25.05.2018).
- LSTS (2013b), *Neptus source code repository*. Available at: <https://github.com/LSTS/neptus> (Accessed: 25.05.2018).
- LSTS (2017), *DUNE Unified Navigation Environment*. Available at: <http://lsts.fe.up.pt/toolchain/dune> (Accessed 01.06.2018).
- LSTS (n.da), *IMC Inter-Module Communication Protocol*. Available at: <https://lsts.fe.up.pt/toolchain/imc> (Accessed: 05.06.2018).
- LSTS (n.d.b), *Manta Communications Gateway*. Available at: [https://lsts.fe.up.pt/support\\_systems/manta](https://lsts.fe.up.pt/support_systems/manta) (Accessed 02.05.2018).
-

- 
- Madureira, L., Sousa, A., Braga, J., Calado, P., Dias, P., Martins, R., Pinto, J. and Sousa, J. (2013), The light autonomous underwater vehicle: Evolutions and networking, *OCEANS - Bergen, 2013 MTS/IEEE* . pp. 1 – 6. doi: 10.1109/OCEANS-Bergen.2013.6608189.
- MahmoudZadeh, S., Powers, D. M., Sammut, K. and Yazdani, A. (2016), Toward efficient task assignment and motion planning for large-scale underwater missions, *International Journal of Advanced Robotic Systems* 13(5). pp. 1–13. doi: 10.1177/1729881416657974.
- Martins, R., Dias, P. S., Marques, E. R. B., Pinto, J., Sousa, J. B. and Pereira, F. L. (2009), Imc: A communication protocol for networked vehicles and sensors, *OCEANS 2009-EUROPE*, pp. 1–6.
- McColgan, J. and McGookin, E. W. (2016), Coordination of Multiple Biomimetic Autonomous Underwater Vehicles Using Strategies Based on the Schooling Behaviour of Fish, *Robotics* 5. p. 2. doi: 10.3390/robotics5010002.
- McIntyre, D., Naeem, W., Ali, S. S. A. and Anwer, A. (2016), Underwater surveying and mapping using rotational potential fields for multiple autonomous vehicles, *2016 IEEE International Conference on Underwater System Technology: Theory and Applications (USYS)*, pp. 77–82.
- McMahon, J. and Plaku, E. (2016), Mission and motion planning for autonomous underwater vehicles operating in spatially and temporally complex environments, *IEEE Journal of Oceanic Engineering* 41(4). pp. 893–912. doi: 10.1109/JOE.2015.2503498.
- OceanScan-MST (2015), *LAUV System - Guidelines for Software Development*, OceanScan MST.
- OceanScan-MST (2016), *LAUV Operator Manual - Release 2.4-rc0*, OceanScan MST.
- Paliotta, C., Belleter, D. J. and Pettersen, K. Y. (2015), Adaptive Source Seeking with Leader-Follower Formation Control, *IFAC-PapersOnLine* 48(16). 10th IFAC Conference on Manoeuvring and Control of Marine Craft MCMC 2015, pp. 285 – 290. doi: 10.1016/j.ifacol.2015.10.294.
- Palomeras, N., El-Fakdi, A., Carreras, M. and Ridao, P. (2012), Cola2: A Control Architecture for AUVs, *IEEE Journal of Oceanic Engineering* 37(4). pp. 695–716. doi: 10.1109/JOE.2012.2205638.
- Palomeras, N., Ridao, P., Carreras, M. and Silvestre, C. (2008), Towards a Mission Control Language for AUVs, *IFAC Proceedings Volumes* 41(2). 17th IFAC World Congress, pp. 15028 – 15033. doi: 10.3182/20080706-5-KR-1001.02543.

- 
- Pan, W.-W., Jiang, D.-P., Pang, Y.-J., Li, Y.-M. and Zhang, Q. (2017), A multi-AUV formation algorithm combining artificial potential field and virtual structure, 38. pp. 326–334. doi: 10.3969/j.issn.1000-1093.2017.02.017.
- Pantelimon, G., Tepe, K., Carriveau, R. and Ahmed, S. (2018), Survey of Multi-agent Communication Strategies for Information Exchange and Mission Control of Drone Deployments, *Journal of Intelligent & Robotic Systems* . doi: 10.1007/s10846-018-0812-x.
- Pinto, J., Calado, P., Braga, J., Dias, P., Martins, R., Marques, E. and Sousa, J. (2012), Implementation of a Control Architecture for Networked Vehicle Systems, *IFAC Proceedings Volumes* 45(5). 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles, pp. 100 – 105. doi: 10.3182/20120410-3-PT-4028.00018.
- Py, F., Rajan, K. and McGann, C. (2010), A Systematic Agent Framework for Situated Autonomous Systems, Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 2 - Volume 2, AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 583–590. Available at: <http://dl.acm.org/citation.cfm?id=1838178.1838183> (Accessed 01.12.2017).
- Rist-Christensen, I. (2016), *Autonomous Robotic Intervention using ROV*, Master's thesis, Norwegian University of Science and Technology. Available at: <https://brage.bibsys.no/xmlui/handle/11250/2440564> (Accessed: 01.12.2017).
- Rout, R. and Subudhi, B. (2016), A backstepping approach for the formation control of multiple autonomous underwater vehicles using a leader–follower strategy, *Journal of Marine Engineering & Technology* 15(1). pp. 38–46. doi: 10.1080/20464177.2016.1173268.
- Ruud, F. J. (2016), *Autonomous Homing and Docking of AUV REMUS 100*, Master's thesis, Norwegian University of Science and Technology. Available at: <https://brage.bibsys.no/xmlui/handle/11250/2410753> (Accessed: 01.12.2017 ).
- Sariel, S. (2007), *An integrated planning, scheduling and execution framework for multi-robot cooperation and coordination*, PhD thesis, Istanbul technical university. Available at [http://web.itu.edu.tr/sariel/thesis/sariel\\_PhD\\_Thesis\\_2007.pdf](http://web.itu.edu.tr/sariel/thesis/sariel_PhD_Thesis_2007.pdf) (Accessed: 01.06.2018).

- 
- Soares, J. M., Aguiar, A. P., Pascoal, A. M. and Martinoli, A. (2013), Joint asv/auv range-based formation control: Theory and experimental results, 2013 IEEE International Conference on Robotics and Automation, pp. 5579–5585.
- Sorbi, L., De Capua, G., Fontaine, J.-G. and Toni, L. (2012), A Behavior-Based Mission Planner for Cooperative Autonomous Underwater Vehicles, 46. Available at: [https://infoscience.epfl.ch/record/182865/files/5.Manuscript\\_Sorbi\\_et\\_al.pdf](https://infoscience.epfl.ch/record/182865/files/5.Manuscript_Sorbi_et_al.pdf) (Accessed: 23.01.2018 ).
- Sujit, P. B. and Saripalli, S. (2013), An Empirical Evaluation of Co-ordination Strategies for an AUV and UAV, *Journal of Intelligent & Robotic Systems* 70(1). pp. 373–384. doi: 10.1007/s10846-012-9728-z.
- Willcox, S., Goldberg, D., Vaganay, J. and Curcio, J. A. (2006), MULTI-VEHICLE COOPERATIVE NAVIGATION AND AUTONOMY WITH THE BLUEFIN CADRE SYSTEM. Available at: [https://www.researchgate.net/publication/241654294\\_MULTI-VEHICLE\\_COOPERATIVE\\_NAVIGATION\\_AND\\_AUTONOMY\\_WITH\\_THE\\_BLUEFIN\\_CADRE\\_SYSTEM](https://www.researchgate.net/publication/241654294_MULTI-VEHICLE_COOPERATIVE_NAVIGATION_AND_AUTONOMY_WITH_THE_BLUEFIN_CADRE_SYSTEM) (Accessed: 02.04.18).
- Yan, Z., Liu, X., Jiang, A. and Wang, L. (2016), Formation control of multiple uuv's based on virtual leader, 2016 35th Chinese Control Conference (CCC), pp. 4621–4626.
- Yao, Y. (2013), Cooperative Navigation System for Multiple Unmanned Underwater Vehicles, *IFAC Proceedings Volumes* 46(20). 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013, pp. 719 – 723. doi: 10.3182/20130902-3-CN-3020.00127.
- Zhang, L. C., Wang, J., Tonghao, W., Liu, M. and Gao, J. (2016), Optimal formation of multiple AUVs cooperative localization based on virtual structure, OCEANS 2016 MTS/IEEE Monterey, pp. 1–6.



# Appendix **A**

## Attachments

Appendix A contains the attachments to this thesis, which are stored in a ZIP-file delivered electronically together with the thesis. The file includes the following folders:

### **A.1 LogFiles**

For each simulation and field test, the results are computed based on logged measurement data. This folder includes the complete log files, scripts for reading the files, process the data, and create figures.

### **A.2 Poster**

The A2 poster (2xA3) has been included in .pdf-format, and is found in this folder.

### **A.3 Source**

All code required to run both the simulators and the MCS are included in this folder. The contents of each subfolder are described briefly, and the most important files are named in the following subsections.

#### **A.3.1 dune**

The dune folder contains all the code files required to run the simulators. Files of interest are the initialization files; *backseat1.ini*, *backseat2.ini*, and *backseat3.ini*. These files link to all the remaining files, and how to run the simulator is discussed in Section 3.1.1.

---

### A.3.2 MCS

All developed code for the MCS, the qcustomplot library, and images used in the application are included in this folder. Files of interest are *gui.cpp* and *gui.h*. The user interface is defined in the auto-generated file *gui.ui* in XML-format. To study the project as a whole, Qt is recommended as the IDE.

### A.3.3 MCSlib

All developed code for MCSlib is included in this folder. Files of interest are *mcslib.cpp* and *mcslib.hpp*, the remaining files were auto generated by Qt when the library was compiled and built. The library also includes two header files, which are placed in a folder named *include* within this folder. *imcDefs.h* stores the id number of relevant IMC message types, and *TcpLink.hpp* is developed by OceanScan, and defines all the functions that handle the TCP connection.



Appendix **B**

Abstract Submitted to 2018 IEEE OES  
Autonomous Underwater Vehicle  
Symposium

# Mission planner for AUV swarms: A verification procedure combining simulation and experiments

Stephanie Buadu  
*Dept of Marine Technology*  
NTNU  
Trondheim, Norway  
[stephaniebuadu@gmail.com](mailto:stephaniebuadu@gmail.com)

Tore Mo-Bjørklund  
*Dept of Marine Technology*  
NTNU  
Trondheim, Norway  
[toremobjo@gmail.com](mailto:toremobjo@gmail.com)

Ingrid Schjølberg  
*Dept of Marine Technology*  
NTNU  
Trondheim, Norway  
[Ingrid.Schjolberg@ntnu.no](mailto:Ingrid.Schjolberg@ntnu.no)

Keywords: AUV, mission planning, formation control, cooperative system.

## Introduction

Among the current applications for AUVs, both civil and military many require time, space, and functional distribution which is impossible to obtain with a common single vehicle approach. Multi-vehicle application introduces new challenges that must be addressed, such as formation control; the task of controlling multiple vehicles to complete mission objectives while keeping a formation. Formation control becomes a central part of the high-level mission plan which is broken down and specified to compute lower-level mission plans for the individual team members.

In 2014 a survey conducted by (Li, Zhu, & Qian, 2014) stated that the literature on formation control of AUVs is much fewer than those on mobile robots and aircraft. Also, most of the available research was pure theoretical without sufficient experiments and practices. The current situation within the field has not changed significantly, although there are more examples of literature referring to experiments and practice in connection with formation control of AUVs the number fades in comparison to other fields such as ground robots and aerial vehicles. In other words, the applications are ready, and there is a need for well tested and functioning approaches to formation control for AUVs, but the current research does not cover the need.

In this paper, we propose a cooperative AUV system based on the LSTS toolchain (Pinto, et al., 2013) that includes a newly developed application for mission planning, execution, and monitoring of multi-vehicle missions, the MCS (Mission Control System). Within the application, a proposed algorithmic formation control method designed by adapting existing multi-vehicle mission theory from other fields is implemented. The system has been tested in simulation, and experiments with simulated and physical vehicles in cooperation were conducted. The motivation behind this paper is to contribute to the research in the field by documenting the experiences during implementation and testing and sharing the experimental results.

## System description and results

gives an overview of the proposed system including hardware and software components. Starting at the top left, the Light Autonomous Underwater Vehicle (LAUV) Fridtjof is the physical vehicle used during field tests. The Manta Communications Gateway enables communication between LAUV Fridtjof and the remaining software components by routing WiFi-signals. Neptus is the mission command and control software offered in the LSTS toolchain, and in this system, it is utilized first and foremost for commanding LAUV Fridtjof between missions and as a backup control application. There is no explicit functionality for cooperative systems in Neptus, hence the need for the additional functionality that the MCS introduces next to the toolchains offerings. The MCS may be run both standalone and in combination with Neptus. During simulations for instance, Neptus can be left out of the system architecture, as there is no risk of damaging the vehicle, and the extra security and single vehicle control functionality that Neptus offers is not required.

The LAUV simulators used for testing run on the same computer as the MCS, and all communication throughout the system was enabled by Inter-Module Communication (IMC) messages, which are also included in the LSTS toolchain.

Figure 1-4 show the MCS' user interface in which an operator may generate a high-level mission plan which is decomposed and sent to the individual vehicles. The MCS handles formation control and allows the operator to monitor and intervene during mission execution.

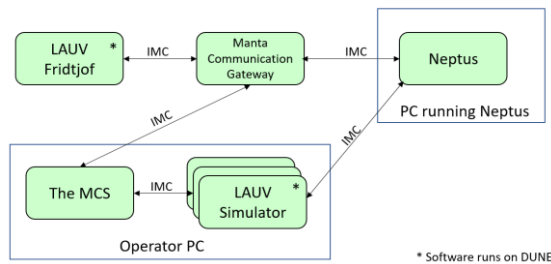


Figure 1: Illustration of system overview.



Figure 2: View 1 of the MCS. For defining the vehicle team, formation, and constraints.

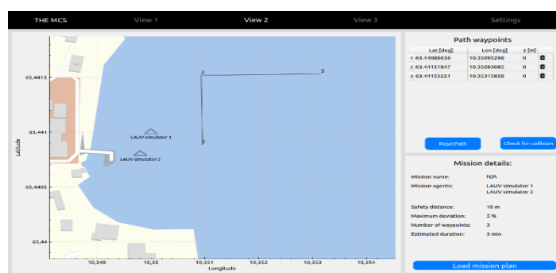


Figure 3: View 2 of the MCS. Enables defining and reviewing the path for the team and loading the mission plan onto the individual vehicles.



Figure 4: View 3 of the MCS. The view for starting mission execution and monitoring mission execution.

The designed formation control method includes elements from virtual structures, leader-follower, and behavioral methods. Each vehicle implements three behaviors; path following, obstacle avoidance, and formation preservation. Path following is the default behavior. Five cooperative strategies were implemented to handle constraint violations and set the right behavior onto each vehicle.

## Summary

The system was extensively tested through simulations before experiments were conducted in the Trondheim Fjord with a cooperative system consisting of two simulators and LAUV Fridtjof. The results were promising, the MCS performed in accordance with its requirements, and the formation control proved able to control the formation during mission execution.

## References

- Li, X., Zhu, D., & Qian, Y. (2014, 10). A Survey on Formation Control Algorithms for Multi-AUV System. *Unmanned Systems*.
- Pinto, J., Dias, P. S., Martins, R., Fortuna, J., Marques, E., & Sousa, J. (2013). The LSTS toolchain for networked vehicle systems. *2013 MTS/IEEE OCEANS - Bergen*, (pp. 1-9). Bergen.



## IMC Message Specifications

The presented messages are messages extracted from vehicle logs after field tests and simulations and visualized in the Neptus mission review and analysis module. Each section below presents a message type, but the headers are not included. The messages, their relevance to the system, and fields of interest are discussed in Section 3.4.

### C.1 Vehicle State

<b>VehicleState</b>	<b>10 fields</b>
op_mode	MANEUVER
error_count	0
error_ents	
maneuver_type	450
maneuver_stime	1527428942.98295550
maneuver_eta	59 s
control_loops	PATH DEPTH YAW SPEED TORQUE
flags	
last_error	
last_error_time	-1.00000000

**Figure C.1:** The structure of a vehicle state message.

---

## C.2 Estimated State

EstimatedState	20 fields
lat	1.10725086
lon	0.18063809
height	-0.16823564 m
x	31.351217 m
y	101.24058 m
z	-0.24171564 m
phi	0.010207577642345218 deg
theta	-0.5754454119251312 deg
psi	66.28906910267061 deg
u	0.070388295 m/s
v	-0.0034249376 m/s
w	0.021250697 m/s
vx	0.021863757 m/s
vy	0.0575358 m/s
vz	0.021869723 m/s
p	0.01902129810353864 deg/s
q	0.16279745214554814 deg/s
r	0.004801148992067896 deg/s
depth	0.042892102 m
alt	119.943184 m

**Figure C.2:** The structure of an estimated state message.

### C.3 Plan Control State

PlanControlState	8 fields
state	EXECUTING
plan_id	SMB-4 from WP 2
plan_eta	55 s
plan_progress	46.87795 %
man_id	0
man_type	450
man_eta	55 s
last_outcome	FAILURE

Figure C.3: The structure of a plan control state message.

### C.4 Plan Control

PlanControl	7 fields	
type	SUCCESS	
op	START	
request_id	10002	
plan_id	Example mission Formation	
flags		
arg	<b>PlanStatistics</b>	<b>7 fields</b>
	plan_id	Example mission Formation
	type	PREPLAN
	properties	
	durations	Total=37.4,Execution=37.4,Calibration=0.0,Maneuver 0=22.4,Maneuver 1=15.1 tuplelist
	distances	tuplelist
	actions	tuplelist
fuel	Total=0.07,Hotel=0.00,Payload=0.00,Motion=0.07,IMU=0.00 tuplelist	
info	0: executing maneuver	

Figure C.4: The structure of a plan control message.

## C.5 Plan Specification

<b>PlanSpecification</b>	<b>9 fields</b>		
plan_id	Example mission Formation		
description			
vnamespace			
variables			
start_man_id	0		
maneuvers	<b>PlanManeuver</b>	<b>4 fields</b>	
	maneuver_id	0	
	data	<b>Goto</b>	<b>11 fields</b>
		timeout	1000 s
		lat	1.10725825
		lon	0.18063743
		z	0.0 m
		z_units	DEPTH
		speed	1.0
		speed_units	METERS_PS
		roll	0.00000000
		pitch	0.00000000
		yaw	-1.00000000
	custom	tuplelist	
start_actions			
end_actions			
<b>PlanManeuver</b>	<b>4 fields</b>		
maneuver_id	1		
data	<b>Goto</b>	<b>11 fields</b>	
	timeout	1000 s	
	lat	1.10725839	
	lon	0.18064443	
	z	0.0 m	
	z_units	DEPTH	
	speed	1.0	
	speed_units	METERS_PS	
	roll	0.00000000	
	pitch	0.00000000	
	yaw	-1.00000000	
custom	tuplelist		
start_actions			
end_actions			
transitions	<b>PlanTransition</b>	<b>4 fields</b>	
	source_man	0	
	dest_man	1	
	conditions	ManeuverIsDone	
actions			
start_actions			
end_actions			

**Figure C.5:** The structure of a plan specification message containing two maneuvers and a transition.



# Additional information about the vehicle and simulators

## D.1 Simulator Modifications

Table D.1 gives an overview of the files duplicated and modified to enable simulation of multiple vehicles simultaneously.

**Table D.1:** Overview over required duplication and modification to enable several LAUV simulator on the same PC. All files are named relative to the 'dune/etc' folder.

File	Duplicated	Modified variables
backseat.ini	Yes	Port, required files
lauv-simulator.ini	Yes	Vehicle, required files
auv/basic.ini	Yes	Required files
common/imc-adresses.ini	No	IMC adress for new vehicles
common/transport.ini	Yes	HTTP port
transport.ini	Yes	UDP local port, TCP Server local port
simulator.ini	Yes	Initial position

---

## D.2 Vehicle Specification

The configuration and equipment on LAUV Fridtjof during testing is stated in Table D.2.

**Table D.2:** Configuration of LAUV Fridtjof during field tests. Copied from AUR-lab (n.d.)

---

<b>Technical specification</b>	
Vehicle length	180 cm
Weight	25.8 kg
Max operational depth	100 m
Battery	Li-ion
Speed	[0.5, 2] m/s
Endurance	up to 8 h
Data storage capacity	16 Gb and 64 Gb
<b>Communication</b>	
WLAN	up to 1 km
GSM	within 3g coverage
Underwater acoustic	up to 1 km
Iridium SBD module	
Emergency acoustic pinger	up to 2 km
Directional hydrophone	up to 2 km
Attitude and heading reference system (AHRS)	aided by DVL
<b>Equipment</b>	
Downward looking camera	Lumenera Le165
Lighting	LED lighting
Doppler velocity logger	Nortek DVL 1MHz
USBL acoustic modem	Evologics S2CR 18/34
Forward looking sonar	Imagenex 852
User computer	Nvidia Jetson TX1

---

---

### D.3 Connecting to LAUV Fridtjof and Simulators

Table D.3 gives an overview of the server addresses and ports utilized for LAUV Fridtjof and the simulators. Note that the simulators are run on the PC running the MCS and that the server addresses refer to the local PC. To understand how the connection is established, review the start function in `mcslib.cpp`. If the simulators should run on a separate PC, simply modifying the server address in the `gui.cpp` file and copying the required simulator files in the DUNE folder would be sufficient. For the simulators, all their connections can be monitored through the terminal they run in, and Figure D.1 shows how a successful connection between the MCS and a simulator is represented in the terminal.

**Table D.3:** Overview of server addresses and ports for LAUV Fridtjof and simulators.

Vehicle	Server address	Server port
LAUV Fridtjof	10.0.10.70	32603
LAUV Simulator 1	127.0.0.1	32603
LAUV Simulator 2	127.0.0.1	32604
LAUV Simulator 3	127.0.0.1	32605

---

```
umer for 'NavigationUncertainty'
[2018/06/04 11:20:10] - DBG [Transports.TCP.Server/Backseat] >> registering cons
umer for 'DesiredSpeed'
[2018/06/04 11:20:10] - DBG [Transports.TCP.Server/Backseat] >> registering cons
umer for 'DesiredHeading'
[2018/06/04 11:20:10] - DBG [Transports.TCP.Server/Backseat] >> registering cons
umer for 'DesiredHeadingRate'
[2018/06/04 11:20:10] - MSG [Transports.Logging] >> log started '20180604/112010
'
[2018/06/04 11:20:10] - MSG [Monitors.Entities] >> Path Control : Boot -> Normal
| idle
[2018/06/04 11:20:11] - MSG [Monitors.Entities] >> GPS : Boot -> Normal | active
[2018/06/04 11:20:11] - MSG [Monitors.Entities] >> Attitude : Boot -> Normal | i
dle
[2018/06/04 11:20:11] - MSG [Monitors.Entities] >> Operational Limits : Boot ->
Normal | active
[2018/06/04 11:20:15] - MSG [Monitors.Entities] >> Navigation : Boot -> Normal |
active
[2018/06/04 11:20:15] - MSG [Supervisors.Vehicle] >> entity errors cleared
[2018/06/04 11:20:15] - WRN [Supervisors.Vehicle] >> now in 'SERVICE' mode
[2018/06/04 11:20:15] - MSG [Plan.Engine] >> vehicle ready
[2018/06/04 11:20:29] - DBG [Transports.TCP.Server/Backseat] >> accepted connect
ion from 127.0.0.1:54160, client count is 1
```

**Figure D.1:** A successful communication link between a simulator and the MCS visualized in the terminal running the simulator.

# Appendix **E**

## Video from the First Day of Field Testing

A short video (45 sec) showing how the field tests were conducted with LAUV Fridtjof and how it could be monitored in the MCS is available at <https://goo.gl/8wSXSZ>. The shots of LAUV Fridtjof in the water were taken by a drone.