COOPERATIVE ALGORITHMS FOR A TEAM OF AUTONOMOUS UNDERWATER VEHICLES

TAN YEW TECK

(B.Sc.(Hons), M. Eng.)

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Signed: TAN YEW TECK

Date: 25 - 11 - 2014

“Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning.”

Albert Einstein

Acknowledgements

This thesis could not have been completed without the help and support of many friends and colleagues for the last four year in NUS and MIT.

First, I would like to thank my thesis advisor, Dr. Mandar Chitre for allowing me to carry out the PhD studies under his supervision. His technical guidance, support, encouragement and expertise has proved invaluable. Furthermore, I very much appreciate him for sacrificing so much of his personal time in helping me either in software design or hardware development. One of the most exciting periods during my PhD studies was when he allowed me to lead a team to design and build an unmanned surface vehicle for water quality monitoring project. I picked up many skills and experiences that otherwise would not have been obtained from my research topic. I would also like to thank Professor Nicholas Patrikalakis for helping me to secure the SMART PhD fellowship. This work would not be possible without the support of the funding.

Not forgetting the members of Acoustic Research Laboratory (ARL), especially the STARFISH project team: Koay Teong Beng, Eng You Hong, Gao Rui, Chew Jee Loong, Bharath Kaylan, Shilabh Suman and Varadarajan Ganesan for their guidance and support while working with the STARFISH AUV. Their great company during numerous field trials made the experience more enjoyable. Many thanks to Dr. Venugopalan Pallayil and Mr. Mohan Panayamadam for making sure I got hold of all the tools that I needed, both software and hardware, for my research.

The six months research residency in MIT last year was truly a great exposure and the most memorable experience throughout my PhD studies. Great thanks to Professor

Franz Hover for agreeing to have me as a visiting scholar in his lab and allowing me to make use of the kayaks for experiments in the Charles River. Many thanks to members of the HoverGroup too: Mei Yi Cheung, Eric Gilbertson, Brooks Reed, Pedro Vaz Teixeira and Joshua Leighton for their help during the experiments in Boston. It has been a great pleasure knowing and working with them for the period in MIT.

I would like to thank Dr. Kanna Rajan for the opportunity as visiting research scholar in Monterey Bay Aquarium Research Institute (MBARI). Although brief, the time spent in MBARI allowed me to get to know the T-REX reactive mission planner for AUVs. Thanks also to Dr. Frederic Py and Dr. Rishi Graham for their help and support during the stay in MBARI.

Finally, I would also like to thank my foster father, Brian Kelly and my family for their love and continued support throughout this process. Without them, none of this work would have been possible.

This work was funded by Singapore-MIT Alliance for Research and Technology (SMART) PhD fellowship.

Contents

Summary

Multi-vehicle missions offer several advantages over single-vehicle missions in terms of mission complexity and tolerance to single-vehicle failure. However, missions involving multiple underwater vehicles pose two main challenges – the absence of a reliable positioning reference (GPS) and the extremely limited communication bandwidth among the vehicles – both of which limit the application of multi-vehicle cooperation techniques that are commonly used by their land and aerial counterparts.

This thesis develops two cooperative algorithms for a team of Autonomous Underwater Vehicles (AUVs) that address the challenges. First, we design a cooperative navigation strategy for a beacon vehicle to serve as navigation beacon for a team of AUVs. The exchange of navigation information between the beacon and other vehicles improves their individual position estimates. We propose dynamic positioning algorithms for the beacon vehicle and analyse their performances in minimizing the position errors of other vehicles in the team. Second, given the bathymetric terrain maps, we develop cooperative localization using a team of sensor-limited AUVs. The localization of each vehicle is performed via decentralized particle filtering on its bathymetric measurements, assisted by acoustic range and information obtained from peer vehicles through acoustic communication. We extend the filter of an individual vehicle to incorporate information received from another vehicle to better estimate its position, and investigate the impact of communication interval, sensor noise and biases on the localization performance.

Designing a Command and Control (C2) system for a single AUV that is robust and easily extensible to accommodate the requirements of multi-vehicle cooperative missions is another focus of the thesis. In particular, we develop a hierarchical agent-based C2 system for a low-cost modular AUV - the STARFISH AUV - that allocates mission, navigation and vehicle tasks to individual self-contained agents. The collective interactions among the pool of agents enables the AUV to achieve its mission objectives autonomously. The C2 system has been developed and successfully deployed for various single-vehicle, adaptive missions as well as multi-vehicle cooperative missions.

Using both simulations and field testings, we demonstrate the feasibility and capability of the developed algorithms in minimizing the position errors accumulated by the AUVs during mission execution.

List of Tables

List of Figures

List of Abbreviations

aRMS	average Root Mean Square
AUV	Autonomous Underwater Vehicle
AS	Agent Service
ASP	Associated Stochastic Problem
BD	Backseat Driver
C2	Command and Control
CE	Cross Entropy
COP	Combinatorial Optimization Problem
DVL	Doppler Velocity Log
DP	Dynamic Programming
DPS	Direct Policy Search
DR	Dead-Reckoning
EEC	Estimated Error Covariance
EKF	Extended Kalman Filter
GA	Genetic Algorithm
GSM	Global System for Mobile Communications
GPS	Global Positioning System
GUI	Graphical User Interface
INS	Inertial Navigation Systems
IMU	Inertial Measurement Unit
KF	Kalman Filter
LBL	Long Baseline

MDP	Markov Decision Processes
OCXO	Oven-Controlled Crystal Oscillator
OWTT	One Way Travel Time
PF	Particle Filter
PV	Peer Vehicle
RL	Reinforcement Learning
RV	Receiving Vehicle
RMS	Root Mean Square
ROV	Remote Operated Vehicle
SIR	Sampling Importance Resampling
SLAM	Simultaneous Localization and Mapping
TAN	Terrain Aided Navigation
TOA	Time of Arrival
TOT	Time of Transmission
TWTT	Two Way Travel Time
USBL	Ultra-Short Baseline
VLGA	Variable-length Genetic Algorithm

List of Symbols

τ	beacon transmission period
Γ	total mission time
$\mathbf{x}_t^j$	position of the survey AUV j at time t
$\hat{R}_t^j$	range between beacon vehicle and survey AUV j at time t
σ_R^2	variance of range error
ϕ_t^j	heading of the survey AUV at time t
θ_t^j	direction of minimum error for survey AUV j at time t
ε_t^j	error along the direction θ_t^j at time t
$\bar{\varepsilon}_t^j$	error tangential to the direction θ_t^j at time t
σ_B^2	variance of the beacon vehicle's error
$\mathbf{x}_t^B$	position of the beacon vehicle at time t
ϕ_t^B	heading of the beacon vehicle at time t
s^B	navigational speed of the beacon vehicle
α	constant of proportionality, position error growth factor
δ_t^B	turning angle made by the beacon vehicle at time t
$\dot{\phi}_{\max}^B$	maximum turning rate of the beacon vehicle
$D_{\min,\max}$	minimum and maximum distance between the vehicles
η	optimization criteria for the MDP formulation
ψ	stopping criteria for the MDP formulation
ρ	quantile of beacon paths to be evaluated in the MDP formulation
μ	smoothing parameter for the MDP formulation
N_a	action space in the policy table
N_z	state space in the policy table

P_s	elitism selection rate for VLGA
P_c	crossover rate for VLGA
P_m	mutation rate for VLGA
c_x	tidal current, easting direction
c_y	tidal current, northing direction
π_t^i	ith particle at time t
$\mathbf{u}_t$	control input vector
$\mathbf{y}_t$	vehicle's measurement at time t
$\mathbf{F}$	state transition coupling matrix
$\mathbf{G}_{u,t}$	input coupling matrix
ζ_t	process noise
ξ_t	measurement noise
$\mathbf{h}(\pi_t)$	measurement function relates bathymetry information to π_t
σ_ζ^2	variance of process noise
σ_ξ^2	variance of measurement noise
w_t^i	relative weight associated with ith particle at time t
ℓ	length of trajectory history
$\hat{\pi}_t^{\mathrm{PV}}$	peer vehicle's position estimate at time t
$\hat{\pi}_t^{\mathrm{RV}}$	receiving vehicle's position estimate at time t
$\mathbf{P}_t^{\mathrm{PV}}$	peer vehicle's estimated error covariance matrix at time t
$\mathbf{y}_t^{\mathrm{PV}}$	peer vehicle's water depth measurement at time t
ϖ_i^{PV}	peer vehicle's particle approximated by receiving vehicle
ϖ_i^{RV}	auxiliary particle of receiving vehicle

Chapter 1

Introduction

Over the past decade, autonomous robotic systems have been deployed for various exploration missions. These robotic systems typically act as platforms to carry sensors that collect data in an environment that is risky or inaccessible by humans. Perhaps the best known examples are the robotic rovers that were sent to planet Mars in year 2003, where different sensors and apparatuses were instrumented onto the rovers to gather scientific data from the remote planet. The rovers have successfully carried out various missions autonomously and are still operational after more than 10 years on the planet.

Besides space, another environment in which autonomous robotic systems have been deployed, but received less attention, is in the ocean. The ocean is the lifeblood of the Earth; it plays an important role in supporting all living organisms, driving weather and regulating temperature. However, the extent of its influence is still not well understood till this day, due to the lack of available data. According to NOAA [1], More than 70 % of the Earth's surface is covered by the ocean, yet only about 5 % has been explored by humans. Classical ocean exploration relies on static buoys, manned surface and underwater vehicles. The high cost and substantial deployment and retrieval efforts have limited their effectiveness in exploring and gathering scientific data from the ocean.

[1]National Oceanic and Atmospheric Administration – Ocean. http://www.noaa.gov/ocean.html

In recent years, the advancement in the Autonomous Underwater Vehicles (AUVs) technology provides an attractive alternative. They require less efforts to operate, and the cost of maintenance is marginal compared to those of manned vessels. Furthermore, the levels of autonomy that can be implemented in an AUV, or a team of AUVs, has enabled the operators to instruct the AUVs to carry out complex mission tasks which otherwise would not have been possible using the conventional approaches.

1.1 Autonomous Underwater Vehicles

AUVs are fundamentally computer-controlled robotic systems that operate underwater. In contrast with the manned or tethered underwater vehicles, they are self-guided, self-powered vehicles, and have no physical connection to their operator. In general, there are two different classes of AUVs: propeller-driven and buoyancy-driven. A propeller-driven AUV uses propulsion systems like thruster or water-jet to propel itself forward, while the buoyancy-driven AUV utilizes small changes in its buoyancy in conjunction with wings to convert vertical motion to horizontal. Although biomimetic propulsion has emerged as a new class of propulsion, it is still in the research stage and not commonly used in a commercial AUV.

The class of AUVs used are typically dictated by the mission objectives. Propeller-driven AUVs are capable of fast and precise maneuverings, and are suitable for short-range, time-limited missions. Among the vehicles in this class (Fig. 1.1) are the Bluefin 9, 12 and 21 series [1], REMUS [2] and STARFISH [3] AUVs. These vehicles have a cruising speed range from between $1 \sim 3$ m/s and endurance of a few to tens of hours depending on the power source carried onboard.

Conversely, buoyancy-driven AUVs have long endurance but much slower cruising speed. They are suitable for missions that require long-range and *yo-yo* shaped transects, yet do not require precise maneuvering control. An AUV performing a *yo-yo* shaped transect typically descends and ascends between two specific depths while navigating towards a pre-planned location. This maneuvering pattern allows the AUV to

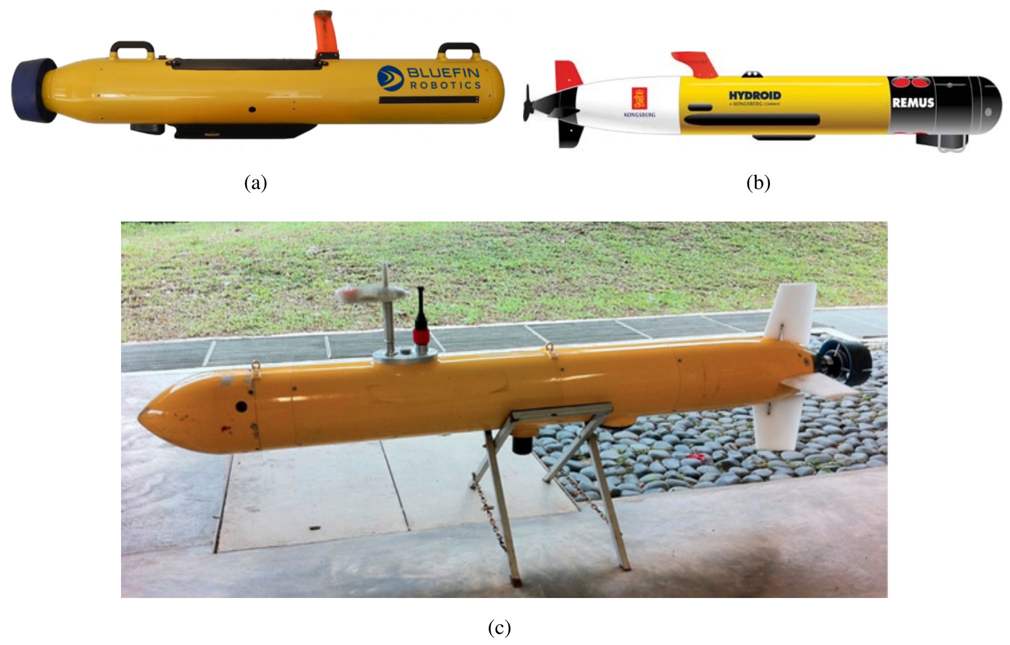


FIGURE 1.1: Propeller-drive AUVs. (a) Bluefin 9 series [1]. (b) REMUS 100 [2]. (c) STARFISH [3].

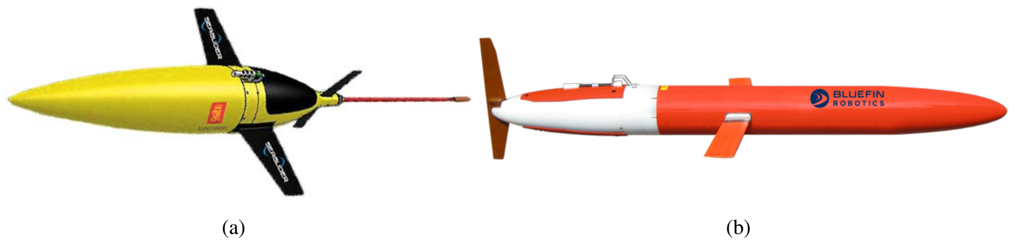


FIGURE 1.2: Buoyancy-driven AUVs. (a) Seaglider [4]. (b) Spray glider [5].

sense and profile the water column between the start and the end point of a mission. Examples of buoyancy-driven AUVs (Fig. 1.2) are the Seaglider [4] and Spray glider [5]. This class of AUVs is capable of cruising around 0.2-0.5 m/s, and covering a range of 6000 km [6].

Apart from ocean exploration, AUVs have been used for a wide range of applications. AUVs equipped with sonar systems are deployed for sea floor [7] and underside of sea ice [8] mapping. More recently, cameras have also been attached to AUVs for mapping coral reefs around shallow waters [9]. Due to strong attenuation of light underwater, the camera can only capture a small area at a time. A complete picture can obtained by mosaicking a series of pictures taken around the coral reefs. Elsewhere, in order to understand the evolution of ocean features like harmful algal blooms or frontal upwelling fronts, scientists have equipped AUVs with chemical sensors and implemented sophisticated motion-planning algorithms on the AUVs to track the features [10]. The examples listed only represent a small subset of many possible applications.

The development of acoustic modems has enabled AUVs to perform acoustic communication. Data can be shared wirelessly with other AUVs or operator working on a mothership, within their communication range. The availability of inter-vehicle communication has opened up possibilities for multi-vehicle operations and cooperation during an underwater mission.

1.2 Motivation

Multi-vehicle missions offer several advantages over single-vehicle missions in terms of mission complexity and tolerance to single-vehicle failure. Multiple vehicles are capable of simultaneously surveying different points of a mission area, thus providing spatio-temporal sampling that a single vehicle simply cannot. This is particularly important in the environmental sensing and monitoring missions where the dynamic of the features of interest evolves at multiple spatial and temporal scales. However, missions involving multiple underwater vehicles pose two main challenges – the absence of a reliable positioning reference (GPS) and the extremely limited communication bandwidth among the vehicles – both of which limit the application of multi-vehicle cooperation techniques that are commonly used by their land and aerial counterparts.

Instead of developing complex, expensive monolithic AUVs for underwater missions, researchers nowadays are moving their attention towards building simpler, low-cost modular AUVs [2, 3, 11]. Depending on the mission requirement, new payload modules can be built and tested independently, before being integrated into the AUV. This approach promotes modularity, thus reduces overall system complexity while increasing the system maintainability. Due to cost restrictions, these AUVs are generally equipped only with low-grade proprioceptive sensors for underwater navigation, resulting in the accumulation of large position errors over the course of a mission. The accuracy of the vehicles' position estimates plays a crucial role during an autonomous underwater mission. First, the quality of the data collected by the vehicles is directly related to the accuracy of their position estimates. Second, missions that call for adaptive behaviors among the vehicles may require their trajectories to be re-planned based on the current position estimates. Having a large position error may have catastrophic consequences, as the vehicles' new trajectories may deviate far from their estimates in reality, causing them to move into uncharted areas, where total loss of vehicle could occur.

Surfacing periodically to get a GPS fix to correct the position error may be an option for some missions, but surfacing can jeopardize the vehicles' safety when operating near busy shipping channels, or in rough seas. Surfacing from significant depth also consumes time and energy. For example, an AUV that is capable of descending at a rate of 0.5 m/s, would spend approximately 30 minutes round-trip to and from the surface, if the depth of the water column is 500 m. On the other hand, if remain underwater and navigate at 1.5 m/s horizontally, the AUV could cover a distance of 2.7 km using the same amount of time and energy. Alternatively, navigation methods that involve deploying acoustic beacons are sometimes used. Among these are Long Baseline (LBL) [12], Ultra-Short Baseline (USBL) [13] and GPS Intelligent Buoy (GIB) [14] arrangements, which provide a geo-reference to correct an AUV's position estimate. These methods not only require considerable operational effort, but they also are limited in the operating range, and are costly.

To overcome the issues, alternate means of underwater navigation must be employed. The research presented here focuses on non-conventional, cooperative navigation methods for multi-vehicle missions, using a team of low-cost *sensor-limited* AUVs. In this thesis, *sensor-limited* refers to vehicles equipped only with minimum, low-accuracy sensor-suite such as altimeter, depth sensor and compass. The vehicles are also equipped with underwater modems, allowing them to communicate acoustically with other vehicles in the team. Even though the AUVs are capable of measuring terrain information as well as estimating inter-vehicle ranges with these sensors, these measurements are not commonly used, especially for underwater localization.

Developing and deploying Command and Control (C2) systems for AUVs is a difficult task. As the demand for AUV autonomy and capability increases, a C2 system not only has to cope with increasing mission complexities, but also has to handle new mission requirements introduced by new sensor payloads. A part of this thesis is devoted to the development of a C2 system that is easily extendable to cope with new mission requirements and allow Software-In-The-Loop simulation [2]. Such a system expedites the development and testing processes, thus shortens the mission turn-around time.

1.3 Objectives

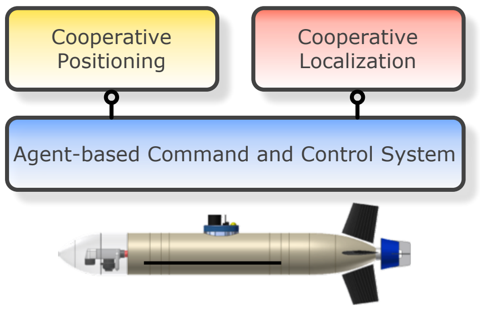


FIGURE 1.3: The objectives of the thesis are to develop cooperative algorithms as well as command and control system for a team of low-cost *sensor-limited* AUVs.

The main goals of this thesis is to design, develop and test cooperative algorithms for the purpose of underwater positioning and localization using a team of AUVs (Fig. 1.3). To meet these goals, this thesis focuses on the following objectives:

1. To develop a cooperative positioning algorithm for a moving beacon so that its position broadcasts can be used to minimize the uncertainties in the position estimates of a team of low-cost, sensor-limited AUVs.

2. To develop a cooperative localization algorithm using terrain information and acoustic communications among a team of low-cost, sensor-limited AUVs.

[2]Software-In-The-Loop simulation allows an actual system software to be tested in a simulation environment, before migration to a physical system.

3. To design and develop a fully autonomous C2 system that allows the proposed algorithms to be easily incorporated and tested in an AUV. The C2 system decouples the low level vehicle control from the high level mission planning and execution, thus enables the developers to focus on developing high level inter-vehicle cooperative behaviors. Furthermore, the C2 system's capabilities must be easily extendable to cope with new mission behaviors of a low-cost modular AUV.

1.4 Thesis Contributions

This thesis contributes to the design and development of cooperative missions using a team of AUVs. The key contributions are listed below.

1. The formulation of AUVs' underwater positioning that relies only on a moving beacon or the bathymetry information of a mission area, assisted by inter-vehicle acoustic communications. These approaches avoid the need to deploy an underwater positioning system such as USBL or LBL, and allow the AUVs to remain submerged underwater for a longer period of time without incurring large position error.

2. A cooperative path planning algorithm for a moving beacon to support other AUVs in team operation. The algorithm is formulated within a Markov Decision Process (MDP) framework, which takes into account and minimizes the positioning errors being accumulated by the AUVs.

3. Two different approaches of learning the cooperative path planning policy for a moving beacon(a) using the cross-entropy (CE) method and (b) using the variable-length genetic algorithm (VLGA). Both alleviate the "curse of dimensionality" problem usually associated with MDP formulation when the state space is large.

4. A new approach for cooperative localization based on decentralized particle filtering, using a team of sensor-limited AUVs. Each vehicles runs a particle filter to estimate their respective positions using its own bathymetry measurements,

and broadcast the filter's information via acoustic communication. Once received by other vehicles in the team, the information is used to influence their filters' particle distribution and assist the position estimation.

5. Empirical studies of the impact of various parameters on the performance of the cooperative localization filter.

6. A hierarchical agent-based C2 system for a single AUV that is robust and easily extensible to accommodate the requirements of multi-vehicle cooperative missions. The C2 system that clearly allocates mission, navigation and vehicle tasks into individual self-contained agents, each with their own responsibilities and behaviors. The C2 system has been successfully deployed on the STARFISH [3] AUVs for numerous field experiments around the Singapore coastal waters.

7. Adoption of Backseat-driver paradigm at the Supervisory level of the C2 system where mission decisions are made based on the inputs provided by a pool of Backseat-driver (BD) agents, each implements different algorithms to achieve specific mission objectives. The C2 system's mission capabilities can be easily extended via the introduction of new BD agents that exhibit desired mission behaviors. Besides, the approach also allows online mission adaptation since the BD agents are able to interrupt the mission execution and propose alternate mission objectives when necessary. The extensibility and adaptability of the C2 system framework have enabled various single and multi-vehicle missions with very different mission requirements to be conducted successfully, both in the lake and sea environments.

8. Field experimental results using the C2 system on different robotic platforms have demonstrated its practicality in coping with different mission scenarios and verified the performance of the proposed methods.

1.5 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 presents an overview of the state of the art in the domains which are the focus of this thesis: underwater communication, cooperative positioning using acoustic beacon, bathymetry-based localization and command and control system for AUVs.

Chapter 3 introduces the cooperative positioning problem using a single moving beacon, and presents the formulation of the beacon's path planning policy within a MDP framework. Two approaches are adopted to automatically learn the resulting policy: the cross-entropy method and the variable-length genetic algorithm. Simulation and field trial results are also presented.

Chapter 4 presents cooperative localization of a team of AUVs using terrain information from a given bathymetry map, and acoustic communications among the vehicles in the team. Field data collected from trials in two locations with different terrain variabilities are used for performing offline localization. Studies are carried out to investigate the impact on performance of sensor noise, communication intervals and losses, and the existence of an ocean current.

Chapter 5 presents the design and development of the hierarchical agent-based C2 system. The concept of back-seat driver paradigm at the mission level of the control system is introduced. The capabilities of the resulting C2 system are illustrated through simulations and field deployments on the STARFISH AUVs. Finally, Chapter 6 summarizes the contribution of this thesis and highlights the future research directions.

Chapter 2

Background

In order to carry out a cooperative mission underwater, a team of AUVs must be able to carry out inter-vehicle communication, and estimate their individual's position reliably, repeatedly. This chapter reviews previous research related to cooperative positioning using a moving beacon, as well as bathymetry-based localization for AUVs. It also reviews some popular command and control systems that are currently being deployed in autonomous robotic systems. Apart from providing a brief background on the existing body of work in the domain of this thesis, it also aims to highlight the gaps that help to identify the problems and issues being addressed by this thesis.

2.1 Cooperative Positioning

Recent advancements in the development of AUVs and underwater communications have made inter-vehicle acoustic ranging a viable option for underwater cooperative navigation and localization. The idea of cooperative positioning is to have a vehicle with good quality position information (beacon vehicle) to transmit its position and time-of-transmission (TOT) acoustically to supported AUVs (survey AUVs) within its communication range during navigation. The time-of-arrival (TOA) is recorded when the data is received at the receiver's transducer. The difference between the TOA and the encoded TOT (known as time-of-flight) are then combined with received position information of the beacon vehicle to estimate range. This approach requires timing synchronization between the beacon vehicle and the survey AUVs. The time-of-flight is known as one-way-travel-time (OWTT) [15]. However, in the absence of timing synchronization, the vehicle must interrogate other vehicles in the acoustic network and measure the time-of-flights between it and all replying vehicles. The inter-vehicle range is then estimated using the two-way-travel-time (TWTT) of the acoustic signal.

The range information between the vehicles can then be fused with the data obtained from proprioceptive sensors in the survey AUVs to reduce the positioning error during underwater navigation. Generally, the beacon vehicle is equipped with high accuracy sensors that are able to estimate its position with minimum errors. In some cases, the beacon vehicle may operate at the surface and have access to GPS for position estimation. Between acoustic communication, the individual vehicle's position is estimated solely by dead-reckoning. Dead-reckoning is the process of computing one's current position using a previously known position, advanced by a known or estimated speed over elapsed time and path.

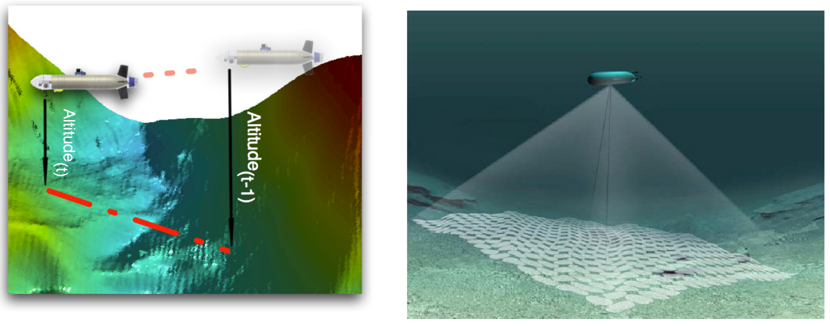
Depending on the accuracy of the beacon vehicle's position information, cooperative positioning is able to provide bounded-error position estimates. In addition, when compared to the statically-deployed underwater positioning systems, which offer only a few kilometers operating range, this approach has an advantage in that the navigation can be conducted on an unbounded area as long as the beacon vehicle navigates within the communication range of the survey AUVs.

The idea of cooperative positioning with a few vehicles that know their positions well and other AUVs with poor navigational sensors is not new. The vehicles with accurate position estimates are referred to by some authors as master vehicles [16], and by others as communication and navigation aids (CNA) [17, 18]. Although multiple beacon vehicles can provide higher accuracy navigation, our research focuses on single beacon cooperative navigation due to its operational advantages and lower inter-vehicle communication requirements. The earliest related research known to the authors is reported in [19], where a least-square approach is adopted to estimate AUV's position from a series of range data transmitted from a LBL-beacon system. A LBL system uses a network of sea-floor mounted baseline transponders as reference points for navigation. The network of transponders measures the distance from a vehicle acoustically and use the measurements to triangulate the position of the vehicle. Although only simulation results were presented, this research has motivated different methods in cooperative positioning. However, the reliance on the sea-bottom fixed beacons for underwater positioning limits its operational flexibilities as they have limited operating range, as well as being time consuming for deployment and retrieval.

In the absence of underwater positioning systems like the LBL system, a mobile CNA is used as the navigational aid. A number of related research are reported in [15, 16, 18, 20], where the acoustic signal transmitted by the CNA is used by the receiving vehicles for cooperative localization. In [16] the authors made use of range information and an Extended Kalman Filtering (EKF) transmitted by the master vehicle to estimate other AUV's position. The authors in [18] adopted a similar approach and compared its performance with two other estimators: Particle Filtering and Nonlinear Least Square (NLS) optimization. Field experiments using a surface craft as the CNA shows that NLS provided the best performance. In [15] the authors extended a centralized EKF approach to a Decentralized Extended Information Filter (DEIF) for cooperative localization and showed comparable filter performance with its centralized counterpart in localizing a single underwater vehicle, but with a lower communication requirement.

Although most of these authors acknowledge that the relative motion of the vehicles is key to having single beacon range-only positioning perform well, the problem of determining the optimal path of the beacon vehicle given the desired path of the survey AUVs has received little attention. For example, the research in [16] assumes a circular path for the beacon vehicle, while [18] uses zig-zag path during experiments. In order to maximize the mission period of a survey AUV for cable or pipeline surveyings, the author [20] suggested that the leading beacon vehicle would likely have to maneuver off course from its pre-planned path to achieve sufficient relative change of motion to fix the survey AUV's position. More recently, the authors in [15] also adopted a similar approach and maneuvered the beacon vehicle above the survey site in a diamond shape while keeping station at each apex to increase observability. These approaches of maneuvering the beacon vehicle in minimizing the survey AUVs' position error are opportunistic and sub-optimal as best. Ranging information is broadcast by the beacon vehicle at some pre-determined periods and paths, without taking into account the position error accumulated by the survey AUVs. The only research known to the author and specifically designed to address this problem is reported in [21, 22]. In [21], the CNA determines its optimal position for acoustic communication based on the prediction of the AUVs' future trajectories. The optimal position is defined as the location reachable by the beacon vehicle at the next immediate time step, such that the ranging information could best minimize the position error of the receiving vehicle. The prediction is performed by using navigational information received from the periodic broadcasts of the AUVs. However, the approach is optimal in a local sense (based on what is optimal at the time the decision is made). As its authors noted, the approach can lead to a sub-optimal long-term solution as the distance between the vehicles constantly grows until the distance is too long for acoustic transmissions. The requirement to broadcast the pose estimates, covariance matrix, course and speed could lead to substantial amounts of data to transfer in a very limited acoustic communication channel.

In [22], the author applied the Dynamic Programming (DP) approach in computing the optimal position for the beacon vehicle to broadcast the ranging information. Given the current location of the beacon vehicle, the DP approach computes an optimal path recursively until the end of the mission, and assigns the first point in the path as the next position for the beacon vehicle. However, this approach suffers from the drawback of high computational load and is not practical for real-time implementation. Part of the work presented in this thesis is concerned with designing a cooperative positioning algorithm for the beacon vehicle, in which the authors extended [22] and formulate the problem within a MDP framework as described in Chapter 3, and utilize machine learning techniques to automatically learn their planning policy. Simulations and field experiments are conducted to demonstrate the capability of the algorithms in minimizing the survey vehicles' position errors.



(a) (b)

FIGURE 2.1: Different approaches for measurement model's update stage. (a) Sequential approach. (b) Batch approach from [26].

2.2 Bathymetry-based Localization

Bathymetry-based localization and navigation, also known as Terrain Relative Navigation (TRN) [23], Terrain-aided Navigation (TAN) [24], and Bathymetric-aided Navigation (BAN) [25] has been used for decades in aircraft and cruise missiles. Given a bathymetric map, the idea of bathymetry-based localization is essentially to match water depth measurements with the map, in order to estimate the vehicle's position. The performance of this localization technique obviously depends heavily on the variability of bathymetry in the area of operation.

Bathymetry-based localization generally employs sequential Bayesian filtering to estimate the probability of a vehicle being at a particular location in the map, using process and measurement models [23–25]. The measurement model can be updated using two different approaches: batch or recursive. The batch approach is based on matching all the terrain profile measurements periodically with a prior bathymetry map, while in recursive approach, the profile measurements are processed sequentially as they arrive, to estimate the vehicle's position. Typically, the type of sensor used for measuring the terrain profile determines the approach employed: single-beam echo-sounder or altimeter calls for sequential approach, while multi-beam sonar or the Doppler Velocity Log (DVL) which consists of 4 acoustic beams to measure velocity as well as altitude of the device, can be used in batch approach. Fig. 2.1 illustrates both the approaches.

Since there is no closed-form solution for the *posterior* probability density, due to the highly non-linear bathymetric measurement model, sequential Monte Carlo filtering methods are used as an approximation of the density [27, 28]. In [27] the authors applied both the Point Mass Filter (PMF) and the Particle Filter (PF) for underwater navigation using multi-beam echo-sounder. Offline filtering with field data showed that the PMF slightly outperformed the PF, though it is more computationally expensive. While in [28], the authors adopted the PF for underwater navigation and compared the estimation results to that of those computed by the Cramer Rao Lower Bound (CRLB) along the experimental trajectories to illustrate the efficiency of the filter. Although the CRLB provides a good indicator of the performance of the localization filter, it is not the focus of this thesis.

Often a particle filter is designed to estimate and track a large number of system variables which requires a large number of particles for the filter to converge. This poses a challenge for the AUVs' limited computational power onboard. In order to alleviate this, a number of researchers have adopted an approach called the Marginalized Particle Filter (MPF), also referred to as Rao-Blackwellization [29–34]. The idea behind the MPF is to marginalize the system states that exhibit linear dynamics, and to estimate the marginalized states using a Kalman Filter. The remaining part of states with reduced dimension can then be estimated by the PF, thus lowering the number of particles required to produce comparable results. The MPF has been employed in [29], in an integrated navigation system of an aircraft with a state vector of more than 15 dimensions, and simulation results showed good performance with a much lower computational load. In the domain of underwater navigation, the authors in [31] have shown the feasibility of applying the MPF for an AUV with a particle set as low as 500 and was able to achieve good localization. The results have encouraged the application of MPF-based localization techniques in low-cost, limited computational-power AUVs. The work presented in this thesis adopts the MPF localization technique due to its advantages.

In most marine applications, the data for the vehicle's measurement model are provided by on-board multi-beam echo sounders [23, 26, 35]. This enables multiple simultaneous altimeter measurements at every time step and improves the filter's performance. Furthermore, if the vehicle is fitted with a DVL, like the research reported in [36], velocity information is available for more accurate propagation of the process model. In fact, the combination of these high data-rate and high accuracy navigational sensors also make underwater bathymetry Simultaneous Localization and Mapping (SLAM) possible. For example, the research reported in [33, 34, 37] made use of multi-beam sonar, DVL, INS and/or IMU to localize the vehicle's position while building 3 - D maps along the vehicle's trajectories. However, these techniques are not suitable for a low-cost AUV, which is capable of carrying only low accuracy sensors and possibly dead-reckon upon its own thruster model to estimate its position. An example is shown in [26] where the localization filter may diverge easily due to multiple occurrences of similar terrain information within the bathymetry map, if the vehicle is assumed to have only a single-beam measurement.

In recent years, researchers also complement bathymetry-based localization with information obtained from other sources of sensor measurements, to better estimate the position of the vehicles. This approach also has the potential to overcome the problem that arises with bathymetry-based localization when the vehicle is over a terrain that contains insufficient information for the filter to converge. The authors in [38] fused both acoustic ranging (obtained from a surface beacon) and position information of underwater targets (obtained by side-scan sonar) to better estimate a vehicle's position and demonstrated the filter's performance via offline filtering with data collected from the field. Another related research is reported in [39], where the DVL measurements are fused with TAN for position estimates. Again, the reliance on these high data-rate and high accuracy sensors makes these techniques not suitable for localization of low-cost AUVs.

The research presented in this thesis is closely related with [40] where range measurements are fused within the bathymetry-based localization filter to estimate a vehicle's position. In contrast with [40], this research does not consider a fixed beacon on the sea floor where an absolute positioning reference can be obtained. Instead, the author employed a team of low-cost AUVs where the localization of an individual vehicle is based on the collective filters' information, fused with the range measurements derived from the communicating vehicles. Even though the cooperative localization approach does not depend on a beacon, it requires the individual filter's information to be broadcast via acoustic communication.

Despite advances in underwater communications, conventional methods of sharing a subset of particles [41] in the implementation of a distributed particle filter simply cannot be applied in the underwater domain due to extremely limited bandwidth and reliability. Various particle distribution aggregations have been developed as alternatives for alleviating communication limits [42, 43], but none of them have been applied in the underwater domain. The approach proposed in this thesis is the first attempt in applying the aggregation technique within the underwater domain.

2.3 Command and Control Systems

Developing the C2 system or mission controller for autonomous robotic systems is a challenging task for researchers. In an autonomous mission, the underlying C2 system's responsibilities include the high-level mission planning and supervisory, as well as the low-level vehicle and navigational control. Furthermore, to carry out the mission successfully, the C2 system has to be robust and flexible in handling uncertainties and animosities that might arise during the robot's operation in a highly hazardous and unknown environment.

The C2 systems generally fall into two different architectures: reactive and deliberative [44]. Deliberative architecture is both hierarchical and top-down in its control structure [45]. Planing and decision making are done at the upper level and passed down to the lower level for execution. Deliberative architecture relies heavily on the information of the world model. During a mission, raw data from the sensors are processed and used to update the model. This dynamically acquired and updated model is then used for new plans or actions when necessary. While handling problems in dynamic and partially unknown environments with the latest acquired information is desired for AUV navigation, this approach suffers from computational latency during the sense-model-plan-act process.

On the other hand, Reactive architecture is also known as bottom-up or behavioral architecture [46]. It consists of a set of elemental behaviors that define the AUV's capabilities. Global behavior emerges from the combination of several elemental behaviors activated in parallel when interacting with the world. Behavioral architectures react to the environment directly without involving any high level reasoning or re-planning process. Data are taken directly from the sensors to evaluate the current world model and appropriate behaviors are chosen to adapt to the model. This sense-react principle is suitable for operations in a highly dynamic world. However, this architecture may lead the AUV into dead-ends while navigating because only the immediate sensing is utilized to react with the environment.

Due to the requirement of self-supervisory, goal-oriented and complex nature of an autonomous mission, most of the mission controllers adopt a hybrid approach, which integrates different architectures to utilize the advantages of some architecture while minimizing the limitations of others [46]. In [44] the authors adopted a hybrid approach that utilizes reactive, deliberative, distributed and centralized control within the control architecture of an intelligent autonomous mobile robots. The author applied fuzzy logic for centralized command arbitration by integrating activated behaviors from distributed decision making processes running asynchronously across the robotic system. The modular design of the control architecture allowed subsystems to be designed, developed, tested and modified separately as necessary. Although the mission-based control tasks of the modules were not clearly defined, its architectural design has inspired the work presented in the thesis.

For AUV mission controllers, [47] reports the implementation of a hierarchical mission controller which combined deliberative and reactive control architecture in their semi-AUV, the SAUVIM, to allow both predictability and reactivity. Elsewhere, the authors in [48] developed a reconfigurable mission controller called ARICS that combines the characteristic of both reasoning-based and reactive-reflexive behaviors to provide goal-directed planning and good responsiveness. While the architectures clearly allocated the mission and vehicle tasks in different subsystem modules at different levels of control hierarchies, their capabilities in coping with new mission requirements and scenarios remain unclear.

In terms of software for robotic systems, the challenge lies in building the software stack starting from low level driver and vehicle control, and continuing up through high level perception, supervisory and beyond. Due to the complexity, robotic software frameworks typically consist of integrated modules, each responsible for different functions of the robotic system. Functional modularization helps control dependencies, distribute implementation and increase system flexibility and robustness. Among the frameworks that are available include Orca [49] and ROS [50]. Orca is an open-source Component-Based software engineering framework designed for mobile robotics. It comes with an online repository that provides free, reusable software components for building mobile robots. To promote software reusability, the framework defined a set of commonly-used communication interfaces so that any component implementing the same interfaces could be deployed in the same framework. ROS, on the other hand, is a peer-to-peer software framework for robotic system that was developed with focus in supporting multiple programming languages, tools-based development and runtime environment. It is a general purpose middleware that facilitates inter-module communication and requires the developers to define the control structure as desired.

In AUV research, developers have started to adopt modular based software developments for the control system. A popular example is MOOS [51]. Similar to ROS, MOOS is an open-source middleware that allows a suite of distributed processes to be built and deployed. However, in contrast to the peer-to-peer communication mechanism adopted by ROS, the processes running on top of MOOS communicate with each other via a centralized database process. More recently, the MOOS-IvP [52], an extension of the MOOS middleware that incorporates Interval Programming (IvP) technique for decision making, was developed for unmanned marine vehicles. The focus of the work is on the high-level autonomous decision making where mission decisions are provided by individual mission behaviors implemented in separate MOOS modules. The IvP technique is used for arbitrating among these modules whenever a conflict arises in this behavior-based architecture.

While these frameworks are typically designed with a specific purpose and aspect that are deemed important to the particular developers, they either do not explicitly define the control flow between components or do not allow the framework's mission capabilities to be extended easily. In this thesis, our focus is to develop a C2 framework for modular AUVs that clearly allocates navigational, mission and vehicle tasks into an individual self-contained software module, termed as agent, each with its own vehicle and mission responsibilities. Agent-based modeling provides several advantages in terms of separation of concerns at different levels of a control hierarchy, well-defined inter-agent communication interfaces and organization [53]. Furthermore, the author also emphasizes scalability and extendability of the C2 framework in coping with new mission requirements, both in single and multiple vehicles mission scenarios. Detailed framework design and development is described in Chapter 5.

2.4 Summary

In this chapter, we briefly reviewed cooperative positioning using a single moving beacon. Although subject to extremely limited bandwidth and lossy channel, acoustic communication is a viable option for cooperative navigation. Researchers have been using filtering techniques to estimate the position of an AUV underwater using the acoustic range measurements. However, little attention has been put on the relative motion and the relative geometry of the vehicles involved during navigation, which has a significant impact on the performance of the filters.

We also reviewed various approaches in bathymetry-based localization. Without GPS signal or beacon as a geo-reference, AUVs that are capable of measuring terrain profiles can compare the measurements against *a priori* bathymetry map to estimate their positions underwater. The PF is a common technique used by researchers for the position estimation. The MPF is adopted to alleviate the computational load of PF, yet provides comparable performance. In order to improve performance and address the short-comings of bathymetry-based localization, information obtained from other sensor measurements has been used to complement the filter. However, these techniques are not applicable for a low-cost *sensor-limited* AUV due to the absence of the high-accuracy and high data-rate sensors.

Last, we reviewed different control architectures for mobile robotic systems that are commonly adopted by robotic researchers, and have identified the challenges of developing a robotic software framework. The control architectures are typically designed with specific purpose and aspect that are deemed important to the particular developers. They either do not explicitly define the control flow between components, or do not allow the framework's mission capabilities to be extended easily.

Chapter 3

Cooperative Positioning with a Single Moving Beacon

Since GPS signal is not available underwater, AUVs rely on the on-board sensors such as compass, Doppler Velocity Log (DVL) and Inertial Navigation System (INS) for their position estimation. However, dead reckoning upon these sensors suffers from unbounded error growth due to the integration involved. Although this problem can be avoided by having the AUV surface and obtain a GPS fix, or deploying fixed beacons around the mission area, it may put the AUV and the beacons' safety in jeopardy especially around busy shipping channels. Besides that, in an AUV team operation which has the advantages of simultaneous monitoring and surveying, it is not cost effective to have every AUV carry expensive DVL or INS that can provide an accurate position estimate. With the development of underwater acoustic modems which are capable of measuring the time of travel of acoustic signals among the AUVs, having a single beacon AUV that is equipped with accurate position estimate to cooperatively support other AUVs within its acoustic range seems an attractive option.

Cooperative positioning missions typically consist of a beacon vehicle that acts as a navigational aid for the survey AUVs which are deployed for monitoring or surveying missions. By having a beacon vehicle supporting a team of survey AUVs, we can avoid having to equip every single AUV with expensive navigational sensors. This not only

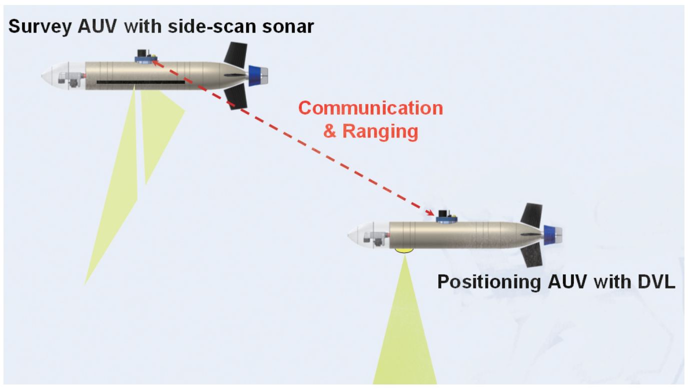


FIGURE 3.1: The two AUVs for cooperative positioning. The range measurement (Ranging) is derived from the travel time of acoustic communication between the AUVs, assuming a known sound-speed profile.

reduces the space required to house all the electronics in the vehicles, but also prolongs precious mission time due to lower power consumption.

Although previous works acknowledged that relative motion between the beacon and survey AUVs play an important role in determining the performance of the cooperative positioning algorithm, little attention has been put towards the motion planning of the beacon vehicle. In this chapter, we develop a path planning algorithm for the beacon vehicle that takes into account the inter-vehicle geometries, and estimated position errors accumulated by the survey AUVs. In order to estimate the position errors, the beacon vehicle keeps track locally an error model of the supported survey AUVs and updates the model whenever there is a ranging information exchange between the vehicles. The planning policy is then learned through simulations using machine learning techniques. Through computer simulations and field experiments in the costal waters around Singapore, we compare the performance of the algorithms with another method described in [22]. Please refer to [54–56] for related publications.

3.1 Cooperative Positioning using Acoustic Ranging

Cooperative AUVs need to communicate in order to cooperate. Hence they are usually fitted with underwater acoustic modems that may also be used to measure range between two vehicles using the travel time of the acoustic signals (Fig. 3.1). The measurements

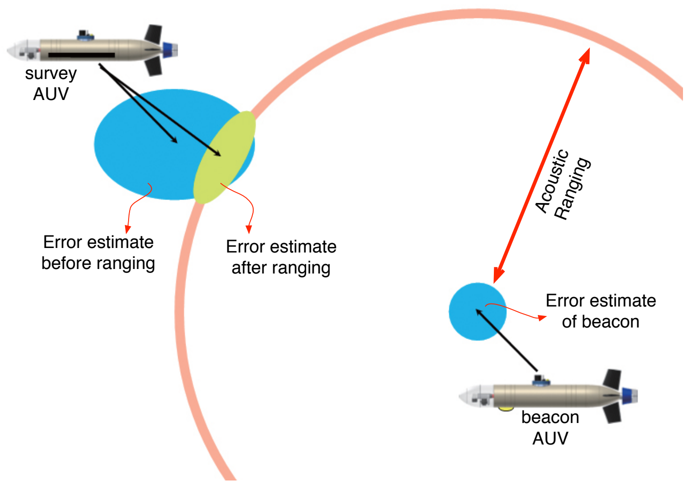


FIGURE 3.2: Illustration of error estimates by range measurements. The error ellipse of the survey AUV (larger blue ellipse next to survey AUV) was reduced (yellow ellipse) by acoustic ranging with beacon vehicle. The error estimate of the beacon vehicle is assumed constant (circle next to beacon vehicle).

are typically performed under the assumption of known sound-speed profile. Depending on the availability of timing synchronization across the vehicles, OWTT or TWTT can be used to estimate the inter-vehicle range. In either case, the position of the beacon vehicle and the estimated range between the vehicles is communicated to the survey AUVs periodically. Upon receiving it, the survey AUVs can fuse the information into its local position estimation filter to better estimate its position.

Although our focus in this work is on the problem of a single beacon vehicle supporting a single survey AUV, we provide a general mathematical formulation where the beacon vehicle may support multiple survey AUVs. The approximate paths to be followed by the survey AUVs are pre-planned. The beacon vehicle's path is planned in real time through a series of sequential decisions made by the onboard command and control system, using information about the survey AUVs' desired path and reported positions during mission execution. The decisions are made with an optimization criteria that minimizes the estimated position error of the survey AUVs, avoids collision between the vehicles and attempts to keep the vehicles within communication range.

Fig. 3.2 shows that the position error estimate (larger blue ellipse) of the survey AUV is reduced in the radial direction (yellow ellipse) of the ranging circle centered at the beacon vehicle each time a range estimate becomes available. The error in the tangential direction remains approximately unchanged and becomes the major axis of the new estimated error ellipse. The cooperative positioning algorithm for the beacon vehicle uses the newly estimated error ellipse and the estimated position of the survey AUV to plan its future motion. If the beacon vehicle can maneuver in such a way that the next range measurement occurs along the direction of the major axis of the error ellipse, the estimated position error of the survey AUV can be minimized. This is the key idea underlying the cooperative positioning for the beacon vehicle. Thus, in order for the single beacon range-only cooperative positioning to perform best, the absolute bearing between the beacon and the survey vehicles' positions should vary such that future range information transmission is along the direction of the major axis of the error ellipse. We term this change in absolute bearing between vehicles' positions as the change of *relative aspect*. This observation agrees with the work in [38] which claims that "ranging from the same relative direction" is one of the factors that results in the reduction of performance of their approach in AUV navigation using both the acoustic ranging and the side-scan sonar.

3.2 Problem Formulation

We assume that the beacon vehicle knows its position accurately and transmits a beacon signal every τ seconds. By measuring the time-of-flight and using either the OWTT or the TWTT of the signal, the survey AUVs' ranges from the beacon vehicle can be estimated. Since the beacon vehicle makes a navigation decision per beacon transmission period, we represent time using an index $t \in \{0 \dots \Gamma\}$. The elapsed time in seconds from the start of the mission to time instant t is simply $t\tau$.

Although the underwater environment is three dimensional, it is common that the depth of the beacon and survey vehicles is specified in a mission and may not be altered by our path planning algorithm. We therefore represent the position of each vehicle using a two dimensional position vector and the direction of travel of each vehicle by a yaw angle. Let $\mathbf{x}_t^B$ be the position and ϕ_t^B be the heading of the beacon vehicle B at time t. Let M be the number of survey AUVs supported by the beacon vehicle.

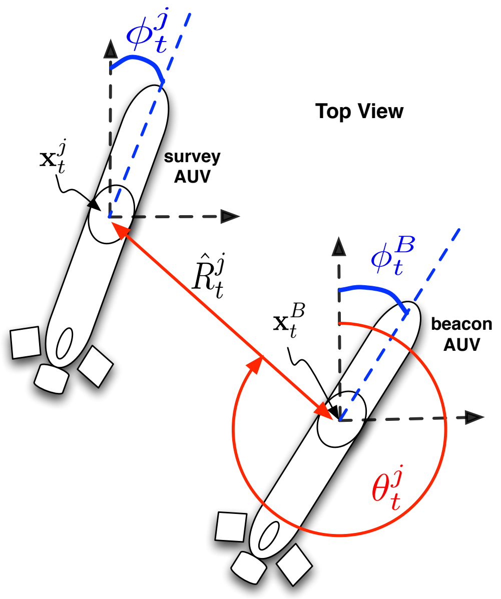


FIGURE 3.3: The beacon AUV's position $\mathbf{x}_t^B$ and heading ϕ_t^B, the j^{th} survey AUV's position $\mathbf{x}_t^j$ and heading ϕ_t^j, the direction θ_t^j of minimum error and the estimated range between the vehicles $\hat{R}_t^j$.

We index the survey AUVs by $j \in \{1 \dots M\}$. Let $\mathbf{x}_t^j$ represent the position of survey AUV j at time t. At every time index t, we have estimates $\hat{R}_t^j$ of the two-dimensional range (easily estimated from the measured range by taking into account the difference in depths between the vehicles) between the beacon vehicle and each of the survey AUVs (Fig. 3.3). We model the error in range estimation as a zero-mean Gaussian random variable with variance σ_R^2:

$$\hat{R}_t^j \sim \mathcal{N}(|\mathbf{x}_t^j - \mathbf{x}_t^B|, \sigma_R^2). \tag{3.1}$$

We further model the error in position estimation of the survey AUVs as a two dimensional zero-mean Gaussian random variable described by three parameters – the direction θ_t^j of minimum error, the error ε_t^j along direction θ_t^j, and the error $\bar{\varepsilon}_t^j$ in the tangential direction. Since ranging from a single beacon vehicle carries only information in the radial direction of the ranging circle centered at the beacon vehicle, the error model allows the beacon vehicle to keep track and minimize the error of the survey vehicle in all directions, regardless of the survey vehicle's pose, provided that they are within the communication range.

Assuming the error in range measurement is much smaller than prior error in survey AUVs' position estimate, the posterior error is minimum along the line joining the beacon and the survey vehicle (please refer to Appendix A. for detailed proof):

$$\theta^j_{t+1} = \angle(\mathbf{x}^B_{t+1} - \mathbf{x}^j_{t+1}) \tag{3.2}$$

$$(\varepsilon^j_{t+1})^2 = \sigma^2_R + \sigma^2_B + \alpha\tau \tag{3.3}$$

where σ^2_B is the variance of zero-mean Gaussian random variable, $\mathcal{N}(0, \sigma^2_B)$, describing the position error of the beacon vehicle and α is the constant of proportionality (determined by the accuracy of the velocity estimate of the survey AUV). The position error of the beacon vehicle is assumed to be isotropic and constant throughout the mission (other error models can easily be accommodated in the formulation). The error in ranging is independent of the error in position. When the distance between the beacon vehicle and the survey AUV is much larger than the positioning error of the survey AUV, the range measurement gives almost no information in the tangential direction and therefore the estimated position error grows in that direction. Assuming that the survey AUVs use velocity estimates (e.g. using DVL or thruster-induced speed) for dead-reckoning, the position error variance in the tangential direction will grow linearly with time (please refer to Appendix A. for detailed proof):

$$(\bar{\varepsilon}^j_{t+1})^2 = \frac{(\varepsilon^j_t \bar{\varepsilon}^j_t)^2}{(\varepsilon^j_t \cos\gamma^j_t)^2 + (\bar{\varepsilon}^j_t \sin\gamma^j_t)^2} + \alpha\tau \tag{3.4}$$

where $\gamma^j_t = \theta^j_{t+1} - \theta^j_t$.

The navigation decision made by the beacon vehicle at each time step t is δ^B_t, the turning angle during the time interval until the next decision. If $\dot{\phi}^B_{\max}$ is the maximum turning rate,

$$|\delta^B_t| \leq \dot{\phi}^B_{\max}\tau. \tag{3.5}$$

If s^B is the speed of the beacon vehicle, then the heading and position of the vehicle at time $t+1$ is approximately given by:

$$\phi_{t+1}^B = \phi_t^B + \delta_t^B \tag{3.6}$$

$$\mathbf{x}_{t+1}^B = \mathbf{x}_t^B + \tau s^B \begin{pmatrix} \cos\phi_{t+1}^B \\ \sin\phi_{t+1}^B \end{pmatrix}. \tag{3.7}$$

In order to ensure that the beacon and survey vehicles do not collide but are within communication range of each other, we require that:

$$D_{\min} \le |\mathbf{x}_{t+1}^j - \mathbf{x}_{t+1}^B| \le D_{\max} \ \ \forall\, j. \tag{3.8}$$

We assume that the position of each survey AUV is known at the start of the mission with an accuracy of ε_0 in all directions:

$$\varepsilon_0^j = \bar{\varepsilon}_0^j = \varepsilon_0 \tag{3.9}$$

$$\theta_0^j = 0 \ \ \text{(arbitrary choice)}. \tag{3.10}$$

Given the desired paths $\{\mathbf{x}_t^j\ \forall\, t\}$ of the survey AUVs and the initial position $\mathbf{x}_0^B$ and heading ϕ_0^B of the beacon vehicle, we wish to plan a path for the beacon vehicle such that we minimize the sum-square estimated position error across all survey AUVs for the entire mission duration. The path is fully determined by the sequence of decisions $\{\delta_t^B\}$ made during the mission:

$$\{\delta_t^B\} = \arg\min \sum_{j,t} \left[(\varepsilon_t^j)^2 + (\bar{\varepsilon}_t^j)^2 \right]. \tag{3.11}$$

3.3 Markov Decision Processes

In this section, we present the formulation of the beacon vehicle's path planning problem within the Markov Decision Process (MDP) framework [54]. Generally, an MDP is defined by four main components: the state and action sets, the state transition probability matrix, and the reward/cost function. From (3.1), $\hat{R}_t^j$ is the estimated distance between beacon vehicle and survey AUV, ϕ_t^{B} represents the beacon vehicle's current bearing at time t and ϕ_{t+1}^j being the survey AUV's bearing at time $t+1$ respectively, our state set is defined as a tuple: $z_t = \{\theta_t^j, \hat{R}_t^j, \phi_t^{\mathrm{B}},\ \phi_{t+1}^j\}$. Since we assume that ε_{t+1}^j in (3.3) is a constant, we need to minimize $\bar{\varepsilon}_{t+1}^j$ in (3.4) to obtain (3.11) for every time step t. This means having γ_t^j in (3.4) to be as close as possible to 90 deg. Thus, the ability of beacon vehicle B to achieve this with respect to survey AUV j will depend on its knowledge of the components in the state space as well as the actions that it can take. Both the $\hat{R}_t^j$ and θ_t^j can be obtained from the acoustic range measurements and communication between the AUVs while ϕ_{t+1}^j is usually pre-planned before the mission.

The action a_t is the turning angle from the beacon vehicle's current bearing (ϕ_t^{B}), $|a_t| \leq \dot{\phi}_{\max}^{\mathrm{B}}\tau$. At every time t, after a_t is selected, the corresponding $\mathbf{x}_{t+1}^{\mathrm{B}}$ can be calculated and the accumulated sum-square error can be estimated through (3.3) and (3.4). We model this accumulated error as the cost function, C, and we are interested in minimizing this cost over the entire mission path, which is equivalent to solving (3.11). An MDP policy is the state-action mapping that determines the probability distribution of action, a_t, when the process is in the state z_t at time step t. We discretize a_t into N_a action states, z_t into N_z states and define a policy matrix, $\mathbf{P}_{za} = (p_{za})$ with $z \in \{1 \ldots N_z\}$ and $a \in \{1 \ldots N_a\}$, such that for each state z, we choose action a with probability p_{za}. This requires that for all z rows in $\mathbf{P}_{za}$, the sum of each z^{th} row is equal to 1. In the case of cooperative path planning, this translates into the probability of choosing a particular turning angle from the beacon vehicle's current bearing (termed as desired heading in the rest of the paper) at time $t+1$, given the beacon vehicle's current bearing, survey AUV's next heading as well as distance and relative angle between the AUVs. As a result, the cost minimization problem reduces to determining the beacon vehicle's path planning policy.

3.4 Policy Learning

Policy learning, also known as policy search, is one of the approaches of the Reinforcement Learning (RL) adopted to "learn" the probability or reward of the state-action mapping in a MDP. Policy search approach is an iterative approach that updates an existing policy with policy changes that will increase the expected reward [57]. The learning process can be further divided into two different classes: gradient-based and gradient-free. A classic example of gradient-based approach is reported in [58], where the policy updating favors towards the direction that lies along the gradient of expected reward. Although suffers from a potential drawback of plateauing at a local optima, the work has received a lot of attention and motivated numerous, improved variants. A detailed account can be found in [59].

In contrast, the research presented in this thesis is based on gradient-free approach, in which the policy search process uses a search heuristic that maximizes the reward, while satisfying a number of constraints set by the problem [57, 60]. In this section, we briefly introduce two different policy learning approaches and illustrate the application as the beacon path planning policy.

3.4.1 Cross-Entropy Method

In this section, we briefly introduce the Cross-Entropy (CE) method and its application in learning the MDP policy. For convenience, we call this the *MDP-CE* method. The Cross-Entropy (CE) method was initially introduced for estimating the probability of rare events in complex stochastic networks [61]. Later, it was modified to solve the Combinatorial Optimization Problem (COP). The main idea behind the CE method in solving COP is the association of an estimation problem with the optimization problem which is called Associated Stochastic Problem (ASP). This ASP, once defined, can be tackled efficiently by iterative estimation procedure shown in Algorithm 1. In what follows, we present the simplified version of the CE method and refer the interested readers to [61] and [62] for its detailed development and formulation.

Suppose we wish to minimize some cost function C on space χ, where χ is the action space defined in the MDP shown in section 3.3. Let η^* denote the minimum of C on χ, $\eta \in \mathbb{R}^+$:

$$\eta^* = \min_{\mathbf{x}\in\chi} C(\mathbf{x}). \tag{3.12}$$

We define a collection of indicator functions $\{I_{\{C(\mathbf{x})\leq\eta\}}\}$ on χ for various thresholds or levels η. Let $\{f(\cdot;\mathbf{P}_{za}), \mathbf{P}_{za} \in V\}$ be a family of (discrete) probability density functions (pdfs) on χ, parameterized by a real-valued parameter $\mathbf{P}_{za}$. For a certain $p \in V$, we can associate with (3.12) the following estimation problem:

$$\begin{aligned} l(\eta) &= \mathbb{P}_p(C(\mathbf{x}) \leq \eta) \\ &= \sum_{\mathbf{x}} I_{\{C(\mathbf{x})\leq\eta\}} f(\mathbf{x};p) = \mathbb{E}_p I_{\{C(\mathbf{x})\leq\eta\}} \end{aligned} \tag{3.13}$$

where $\mathbb{P}_p$ is the probability measure under which the random vector $\mathbf{x}$ has pdf $f(\mathbf{x},p)$. The association comes from the fact that the probability $\mathbb{P}_p(C(\mathbf{x}) \leq \eta)$ will be very small (rare event) when η is close to η^*. By the CE method, this rare event can be estimated by iteratively generating and updating a sequence of tuple $\{(\hat{\eta}_n, \hat{p}_n)\}$ such that it will converge to a small region of the optimal tuple (η^*, p^*). Let ψ be the stopping criteria, the tuple $(\hat{\eta}_n, \hat{p}_n)$ can be updated iteratively by Algorithm 1.

Algorithm 1 Iterative Estimation

```
- Let $\eta_0 = 0$ and $p_0 = 1/\mid\chi\mid$, set $n = 0$
repeat
    - Set $n = n+1$
    - Let $\eta_n$ be the (1-$\rho$)-quantile of $C(\mathbf{x})$ under $p_{n-1}$
    - Generate a set of $N$ random vector from $f(x, p_{n-1})$, denoted as $\mathbf{x}_k$ for $k \in \{1\dots N\}$

    - Estimate $\eta_n$, denoted as $\hat{\eta}_n$, by assigning it as the (1-$\rho$)-quantile of $C(\mathbf{x}_k)$
    where $C(\mathbf{x}_k) \in \{C(\mathbf{x}_1) \leq \cdots \leq C(\mathbf{x}_N)\}$
    - Estimate $p_n$, denoted as $\hat{p}_n$, with fixed $\hat{\eta}_n$ and $p_{n-1}$. The estimation can be derived
    from [61] as:
        $$\hat{p}_n = \arg\max_p \frac{1}{N}\sum_{k=1}^{N} I_{\{C(\mathbf{x}_k)\leq\hat{\eta}_n\}} \ln f(\mathbf{x},\mathbf{P}_{za}). \tag{3.14}$$
until $|\hat{\eta}_n - \hat{\eta}_{n-1}| \leq \psi$
```

To sum up, the CE method generally consists of two important phases:

i. Generate sample data $\mathbf{x}$, according to a specified random mechanism (pdf parameterized by the vector p). Score and rank the resultant sample data according to the cost function $C(\mathbf{x})$.

ii. Select the η and update the parameters of the pdfs on the basis of the data, to produce a "better" sample in the next iteration.

3.4.1.1 Beacon Vehicle's Path Planning Policy Learning

In order to apply the CE method for learning the path planning policy, we must specify the two important phases stated before, which in our case are: (a) how to generate the sample beacon path, and (b) how to update the policy matrix at each iteration.

Since we have formulated the path planning problem within the MDP framework, for a given survey AUV's path with arbitrary path length of t_{LA} steps, we can generate a set of beacon paths with the same path length via the Markov process with the policy matrix $\mathbf{P}_{za}$. Let N be the total number of paths generated in the set, each beacon vehicle's path, $\mathbf{x}_k$, $k \in \{0 \ldots N\}$, consists of a sequence of state-action pairs, $\mathbf{x}_k = (z_0, a_0, \ldots, z_{t_{LA}}, a_{t_{LA}})$. The cost of each resultant beacon vehicle's path can be estimated through (3.3) and (3.4) as shown in Section 3.2.

Let $C(\mathbf{x}_k)$ represent the total cost of path $\mathbf{x}_k$ generated for policy learning at every iteration, we sort the paths' cost in increasing order and evaluate the (1-ρ)-quantile, η. Once the η is selected, the policy matrix can be updated by solving (3.14) to obtain the formula (see [61, 62]):

$$p_{za} = \frac{\sum_{k=1}^{N} I_{\{C(\mathbf{x}_k) \leq \eta\}} I_{\{\mathbf{x}_k \in \chi_{za}\}}}{\sum_{k=1}^{N} I_{\{C(\mathbf{x}_k) \leq \eta\}} I_{\{\mathbf{x}_k \in \chi_z\}}} \tag{3.15}$$

where $C(\mathbf{x}_k) \leq \eta$ means the total cost of path $\mathbf{x}_k$ is less than the selection score, the event $\{\mathbf{x}_k \in \chi_z\}$ means that the trajectory $\mathbf{x}_k$ contains a visit to state z while the event $\{\mathbf{x}_k \in \chi_{za}\}$ means the trajectory corresponding to path $\mathbf{x}_k$ contains a visit to state z in which action a was taken. The learning process is repeated until η converges within the stopping criteria. Detailed steps are shown in Algorithm 2.

Algorithm 2 Policy Learning through Iterative Estimation

```
Require: $\mathbf{P}_{za}$ uniformly initialized with $(1/\mid N_a \mid)$
  - Let $\eta_0 = 0$, set $n = 0$
  repeat
    - Set $n = n + 1$
    for all $z^s$ in $\mathbf{P}_{za}$ do
      - Generate a random surveying path with the first path segment satisfies $z^s$ and
      path length of $t_{LA}$ steps.
      repeat
        - Start from the initial state $z_0 = z^s$, set $i = 0$.
        - Generate an action $a_i$ according to the $z_i$th row of $\mathbf{P}_{za}$, calculate the cost
        $C_i = c(z_i, a_i)$ and generate a new state $z_{i+1}$. Set $i = i + 1$. Repeat till $i = t_{LA}$.
        - Output the total cost $(C(\mathbf{x}_k))$ of the trajectory $(z_0, a_0, \ldots, z_{t_{LA}}, a_{t_{LA}})$.
      until $N$ trajectories
      - Sort the $N$ scores in descending order, take $\eta_n$ as the $(1\ \text{-}\rho)$-percentile of the
      score set.
      - Update the parameter matrix $\mathbf{P}_{za}$ according to equation (3.15).
    end for
  until $|\eta_n - \eta_{n-1}| \leq \psi$
```

Instead of updating the policy matrix $\mathbf{P}_{za}$ directly with equation (3.15), we apply a simple smoothing filter:

$$\hat{p}_{za,n} = \mu \tilde{p}_{za,n} + (1 - \mu)\hat{p}_{za,n-1} \tag{3.16}$$

where $\tilde{p}_{za,n}$ is the solution of equation (3.15) and μ is the smoothing parameter with $0.7 < \mu < 1$, as recommended by [61, 62]. The filter serves two purposes: (i) smoothing the policy matrix update, and (ii) preventing $\hat{p}_{za,n}$ from becoming zero especially during the initial stage of the learning process. This is crucial as to prevent the learning algorithm from finding a local minima and converging to an incorrect solution.

3.4.1.2 Policy Learning Setups and Results

The learning algorithm shown in section 3.4.1.1 was used with the the setup shown in Table 3.1.

In our approach, we do not need to discretize our map into a grid map since we are only concerned with the absolute bearings between the AUVs' positions. However,

TABLE 3.1: PARAMETERS FOR POLICY LEARNING

Parameter	Value
t_{LA}	20
τ	10s
σ_R	1 m
$\dot{\phi}^{B}_{max}$	0.07 rad/s
D_{min}	100 m
D_{max}	1000 m
ε_0	1 m
α	0.1 m^2/s
N	200
ρ	0.1
μ	0.9
ψ	0.1

we do discretize the angle between the AUVs and the AUVs' bearing into 36 states each representing an angle section of 10 deg spanning from 0 $\sim$ 360 deg. The AUVs are allowed to navigate between 100 m and 1000 m within each other, the distance is discretized into 3 states with first 2 zones having 300 m each while the last zone spanning 400 m to provide slightly higher resolution for areas closer to the survey vehicle, since higher change of the *aspect ratio* could be achieved when the vehicles are close to each other. Any distance closer than 100 m or more than 1000 m apart will be given a heavy penalty that will contribute to the accumulated error. This is necessary to prevent the AUVs from colliding if they are too close together while keeping the AUVs within the communication range (which in our case, assumed to be 1000 m). The maximum turning angle of the AUV is 40 deg and is discretized into 8 action states. Increasing the resolution of the state space could potentially improve the outcome of the policy training, but also increase the computational requirement of the training process exponentially. The state and action space formulated for the policy learning are summarized in Table 3.2.

Fig. 3.4(a) shows a example of survey path (red color) randomly generated for the survey AUV, together with N number of beacon vehicle's paths (blue color) generated using the uniformly initialized planning policy. Since all the state-action mappings have equal probabilities during the initial stage, the beacon paths generated were random,

TABLE 3.2: STATE AND ACTION SPACE DISCRETIZATION

State Space, N_z	Number of States
Beacon AUV's current bearing	36
Surveying AUV's next bearing	36
Relative angle between AUVs	36
Distance between AUVs	3
Total :	139968

Action Space, N_a	Number of Action States
Beacon AUV's desired turning angle	8

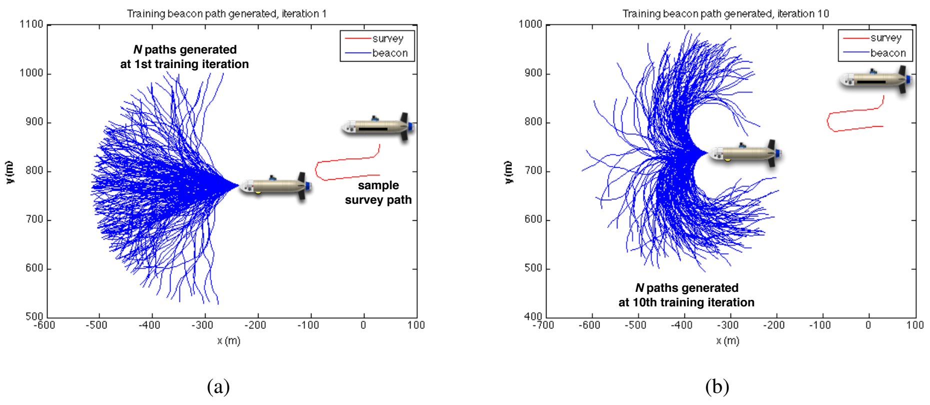


FIGURE 3.4: (a). At the first iteration of the policy learning process, the beacon vehicle's paths were random due to uniform probability when the policy is first initilaized. (b). At tenth iteration, the planning policy starts to converge and generates paths which favor the directions that will minimze the cost function (3.12).

but subject to the vehicle's dynamic constraints mentioned in Table 3.1. However, at the tenth iteration as shown in Fig. 3.4(b), the planning policy starts to converge as most of the beacon paths were generated along the directions that, according to the cost function defined in (3.12), would minimize the position error of the survey AUV. In contrast with the gradient-based approaches [21, 22] which favor only a single solution, the CE method is robust against problems that have multiple solutions, as can be seen in Fig. 3.4(b), where two opposite directions are equally favorable for the beacon vehicle to navigate to in order to minimize the survey AUV's position error.

3.4.1.3 Application to Cooperative Path Planning

Once the policy learning is completed, the path planning for the beacon vehicle supporting a single survey AUV reduces to a policy matrix lookup. At every planning step, the beacon vehicle determines its current state and decides on its next heading using the corresponding action row's probability distribution. This process is repeated until the survey AUV's mission is completed.

3.4.2 Variable-Length Genetic Algorithm

One of the main drawbacks of the approach described in Section 3.4.1 is the tradeoff between computational load against state-action representation: Overly fine discretization would produce a large policy and incurs a higher computational burden, while coarse policy might result in unrepresented states or actions. Besides, the policy learning requires exhaustive search of the value of being in a state, in respect to minimizing the cost function.

In this section, we propose a novel method for Direct Policy Search (DPS) of a MDP and employ the Variable-Length Genetic Algorithm (VLGA) to automatically discover the policy's state-action mapping. We call this the *MDP-GA* method. Given the beacon vehicle's current bearing, the survey AUV's next heading as well as distance and absolute bearing between the AUVs' positions, the cooperative path planning's policy determines the desired turning angle (action) from the beacon vehicle's current bearing (termed as desired heading) so that the position error of the survey AUV can be minimized during the next ranging event.

3.4.2.1 State Space approximation and Action Space Mapping

It is not always easy to design a good policy and predict the value of being in a state based on value function, as it is often computationally infeasible given the limited computational power that an AUV has. In order to alleviate this problem, various approximation techniques have been applied, and encouraging results have been reported in the

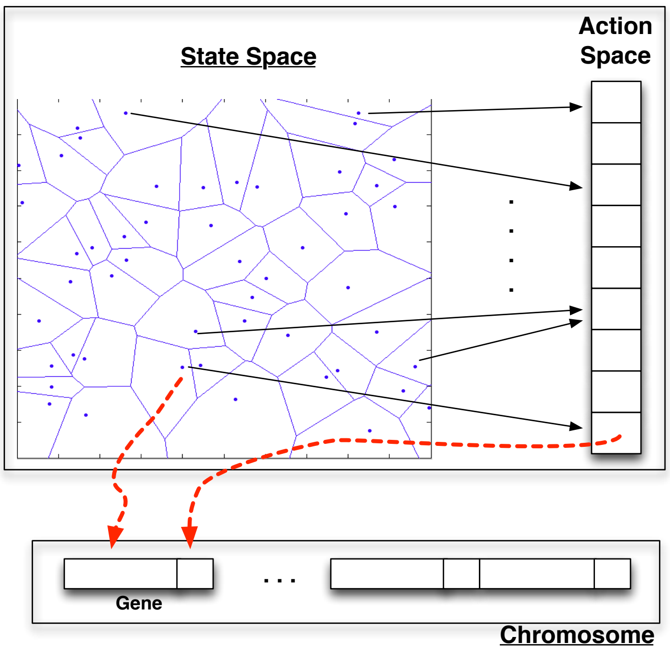


FIGURE 3.5: State-Action space mapping and chromosome representation.

literature [63].

We simplify the state space into the form of Voronoi Tessellation where states located within a Voronoi cell are represented by their Representative States (RStates) specified by their Voronoi seeds. Consequently, the path planning policy is the direct mapping of these RStates into the action space as shown in Fig. 3.5. During cooperative positioning, the beacon vehicle first determines the state using the latest ranging information. It then locates the closest RState in terms of normalized Euclidean distance in the state space. The normalization is to prevent domination of any state components in finding the closest RState. Since each of the RStates is deterministically mapped to a particular action, the decision making using the resultant policy is straightforward. Compared to the previous method [54] presented in the last section, this approximation technique greatly reduces both the size of the policy matrix and the computational load of the beacon vehicle.

3.4.2.2 Variable-length Genetic Algorithm Formulation

Genetic Algorithm (GA) was first designed to search and optimize solutions based on natural selection and natural genetics [64]. It is a class of global optimization technique based on simple yet powerful randomized search procedures. In GA based approaches, the variables are encoded as genes in a chromosome, which in turns represents a candidate solution for the problem at hand. A population of the chromosomes are spawned and allowed to evolve through natural selections and genetic operations so that the fittest chromosome (solution) can be found. The VLGA is an extension of the GA approach that allows the length of chromosomes to vary as the population evolves.

Three important parameters need to be tuned when solving the MDP formulated in Section 3.4.2.1: the number of RStates to fully represent the entire state space, the locations of each of the RStates and their corresponding action mapping in the action space. To search for the optimal parameters, we use a VLGA to automatically discover the number of RStates and their locations in the state space, as well as the RState-action mappings for the resultant policy.

Chromosome Representation

The chromosomes are encoded in binary form. Each of the continuous variables in the state and action space is discretized and encoded as a stream of binary numbers. They represent the locations of the state and action within the space domain. Fig. 3.6 shows an example of the chromosome represented using this scheme. Each of the genes in a chromosome consists of a RState-action pair which represents direct mapping relationship. The length of the chromosomes is variable during the process of evolution and represents the number of RStates for the resulting policy. This representation scheme is important to allow the VLGA to automatically discover the optimal number of the RStates, their locations within the state space, as well as their corresponding action mapping. Since the individual gene encodes the RState's location in the state space and its action mapping, the arrangement of the genes in the chromosome is irrelevant.

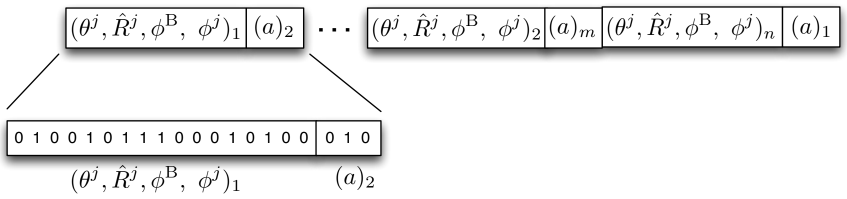

$(\theta^j, \hat{R}^j, \phi^B, \phi^j)_1$
$(a)_2$
$\cdots$
$(\theta^j, \hat{R}^j, \phi^B, \phi^j)_2$
$(a)_m$
$(\theta^j, \hat{R}^j, \phi^B, \phi^j)_n$
$(a)_1$
0 1 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0
0 1 0
$(\theta^j, \hat{R}^j, \phi^B, \phi^j)_1$
$(a)_2$

FIGURE 3.6: Gene representation in Chromosome. Each gene consists of RState-action pair; whenever the RState is selected, the corresponding action will be taken. Each Chromosome in the population represents a beacon vehicle's path planning policy.

Genetic Operations

Genetic operations found in traditional GA are used in this work for the process of evolution. They are described as follows:

Elitism selection and reproduction:

After each evolution process, the chromosomes in the population are sorted in decreasing order based on their fitness. Let P_s be the selection rate, the top P_s % of the population are selected and reintroduced into the new population. Besides that, the same proportion of new chromosomes are randomly generated and introduced into the new generation. The rest of the population are then randomly reproduced from the pool of best chromosomes. This approach ensures the exploitation of the best found solutions as well as exploration of the new solutions in the new population.

Crossover:

Two chromosomes are randomly selected from the population according to the P_c – the crossover rate. One-point crossover is performed between a pair of chromosomes and the new resultant chromosomes are re-introduced into the population. Physically, the crossover operation increases the probability of combining good genes from different parent chromosomes, thus, producing fitter offsprings.

Mutation:

Let P_m be the mutation rate. At every generation, P_m chromosomes are chosen from the new population to undergo mutation. In this work, we apply three different types of mutation operations to the selected sub-population:

- Growth mutation – randomly produces a new gene and appends it to the selected chromosome.
- Shrink mutation – randomly removes a gene from the selected chromosome.
- Flip mutation – applies flipping operation on the genes. The bit is flipped with the probability equal to the mutation rate.

Both the growth and shrink mutation help to introduce good new genes and remove bad genes from the chromosome. Besides, the flipping mutation aids to maintain the diversity of the new population in searching for an optimal solution.

Fitness Function

The fitness function of the chromosomes is evaluated based on the performance of the encoded policy through Monte Carlo simulation. Detailed setups of the simulation are presented in the following section. Since we are searching for a path planning policy that will minimize the cost function, C, of the MDP described in Section 3.3, the fitness function of the chromosomes is defined as follows:

$$f_i = \frac{1}{C_i} = \frac{1}{\sum_{t=1}^{t_{LA}} \left[(\varepsilon_t^{SA})^2 + (\bar{\varepsilon}_t^{SA})^2 \right]} \tag{3.17}$$

where f_i represents the fitness value of the ith chromosome, C_i is the cost incurred from the path planned by the beacon vehicle, which is calculated through the summation of the positioning errors (both the ε_t^{SA} and $\bar{\varepsilon}_t^{SA}$) accumulated by the survey AUV for a sample survey path of t_{LA} steps.

Fitness Evaluation through Monte Carlo Simulation

The fitness of each individual offspring is evaluated through Monte Carlo simulation between the beacon vehicle and a survey AUV. During the simulation, a survey path of t steps with lawn mowing pattern is randomly generated to simulate a survey mission. Starting from all the initial states in the state space, the beacon vehicle is deployed and plans its path to support the survey AUV using the encoded policy. Since acoustic ranging information is assumed to be available at each of the t_{LA} steps, the resultant beacon's path has the same length as the survey path. With both the beacon and survey paths, the sum of the positioning errors (3.17), which is equivalent to the cost, can be calculated. The same simulation is performed using the policies encoded in all the chromosomes in the population, and the resultant fitnesses are ranked in descending order for the selection operation. Detailed algorithm of the simulation is shown in Algorithm 3.

Algorithm 3 Fitness Evaluation through Monte Carlo Simulation

```
Require: $N_z$ – State Space
Require: $Pop$ – Policies represented by chromosomes in the population
  for all $z^s$ in $N_z$ do
    Generate a random surveying path with path length of $t_{LA}$ steps.
    for all $p_i$ in $Pop$ do
      Start from the initial state $z_0 = z^s$, set $j = 0$.
      Locate the RState in $p_i$ that is closest to $z_0$ in terms of Euclidian distance.
      Apply the corresponding action (encoded in the same gene as the selected
      RState) and generate a new state $z_{j+1}$. Set $j = j + 1$. Repeat until $j = t_{LA}$.

      Output the total cost ($C_{p_i}$) of the trajectory ($z_0, z_1, ..., z_{t_{LA}}$).
      Calculate the fitness $f_i$ of the policy $p_i$.
    end for
  end for
  return $f_i$ of all $p_i$ in $Pop$.
```

Policy Search Setup and Results

Instead of discretizing the map into grid map or graph nodes as is commonly done for the path planning problem of mobile robots [65, 66], we discretized both the state and action space of the beacon vehicle. For the convenience of binary encoding of the chromosome, we discretize the AUVs' bearing and the angle between the AUVs into 32 states (5 bits) spanning from $0 \sim 360$ deg. The distance between the AUVs are discretized into 4 zones (2 bits): two forbidden zones (less than $D_{\min}$ and more than $D_{\max}$) and two legal zones with each occupying half of the distance in between $D_{\min}$ and $D_{\max}$. Heavy penalty that will contribute to the accumulated errors is given whenever the vehicles are in the forbidden zones. This is necessary to prevent the vehicles from colliding if they are too close together while keeping them within the communication range. Due to the limitation of the turning radius achievable during navigation, the beacon vehicle's desired turning angle is constrained within [-20,20] deg (obtained

TABLE 3.3: STATE AND ACTION SPACE DISCRETIZATION

State Space, N_z	Number of States	Number of Bits
Beacon vehicle's current bearing	32	5
Surveying AUV's next bearing	32	5
Relatives angle between AUVs	32	5
Distance between AUVs	4	2

Action Space, N_a	Number of States	Number of Bits
Beacon vehicle's desired turning angle	8	3

TABLE 3.4: PARAMETERS FOR BEACON VEHICLE AND VLGA

(a) Beacon's Parameters

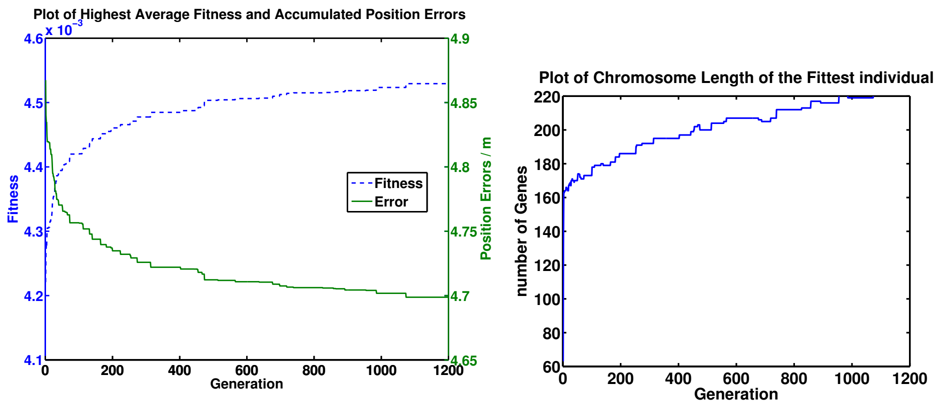
Parameter	Value
t_{LA}	20
τ	10 s
σ_R	1 m
$\dot{\phi}^{\mathrm{B}}_{\max}$	0.07 rad/s
$D_{\min}$	100 m
$D_{\max}$	1000 m
ε_0	1 m
α	0.1 m^2/s

(b) VLGA Parameters

Parameter	Value
P_s	0.1
P_c	0.6
P_m	0.15
Encoding scheme	Binary
Substring length	20
Population size (Pop)	200
Number of generations	1200

from $\tau\dot{\phi}^{\mathrm{B}}_{\max}$ in Table 3.4(a)) of the vehicle's current bearing and is divided into 8 zones. Detailed parameter setups are shown in Table 3.3. Table 3.4 shows the parameters used for the Monte Carlo simulation of the beacon vehicle and the DPS using the VLGA.

The fitness value and the length of the fittest chromosome in each generation of the VLGA are shown in Fig. 3.7. Even though the length of an individual chromosome in the population is allowed to evolve, it stabilizes at about 220 genes for the fittest chromosome. In some instances during the policy search, we observe that the length of the fittest chromosome is shortened (around generation 100, 500 and 700) while its fitness value continued to increase. This shows that the fitness of the chromosome (performance of the policy) does not only depend on the number of the RStates, but also the locations of the RStates and their action mapping.



(a) (b)

FIGURE 3.7: Result of the VLGA showing the fitness value and the length of the fittest chromosome in each generation. (a) The fitness of the chromosome with the highest fitness value. (b) The length of the fittest chromosome.

3.4.2.3 Application to Cooperative Path Planning

At the end of the training process, the fittest chromosome of the latest generation is used as the policy for beacon vehicle's path planning. Instead of policy matrix lookup, path planning using the policy trained with the VLGA method involves locating the closest RState encoded within the genes of the chromosome, and taking its deterministically mapped action. The process is repeated at every planning step, until the survey AUV's mission is completed.

3.5 Simulation

To evaluate the performance of the proposed beacon vehicle path planning algorithms, we simulate two cooperative positioning missions in this section. The first mission involves a single beacon vehicle supporting a survey AUV, while the second mission involves supporting a team of two survey AUVs. The parameters used in the beacon vehicle are listed in TABLE 3.5. We compare the simulation results obtained from both the proposed MDP-CE and MDP-GA algorithms with the DP [22] method to highlight their strengths and weaknesses. The DP method employs a recursive approach in generating an optimal path that would minimize the cost function (3.11). To alleviate the

TABLE 3.5: PARAMETERS USED IN BEACON VEHICLE

Parameter	Value
τ	10s
σ_R	1 m
$\dot{\phi}^{B}_{max}$	0.07 rad/s
$[D_{min}, D_{max}]$	$[100, 1000]$ m
ε_0	1 m
α	0.1 m^2/s

computational burden of the algorithm, the method resorts to decision space (turn angles) approximation and limits the level of value function computation. Four levels of look-ahead (LA-4) strategy and three discrete turn angles (A=3) are used for the DP method. We refer the reader to [22] for its detailed implementation. Throughout the simulations, we assume that the beacon vehicle's position uncertainty is isotropic and constant.

3.5.1 Supporting Single Survey AUV

In the first simulation scenario, a survey AUV was given a lawn-mower mission surveying an area of 500 m by 700 m as shown in the left column of Fig. 3.8. The survey AUVs' paths are pre-planned and are shared with the beacon vehicle. With this information, the beacon vehicle plans its path iteratively using the policy matrix until the survey AUVs' missions are completed. During the simulation, all the vehicles are assumed to be moving at the speed of 1.5 m/s. The survey AUV starts at point [0,0] while the beacon vehicle starts at point [100,100].

The dotted-lines in the left column plots of the Fig. 3.8 show the resultant paths planned by the beacon vehicle using the DP (top), MDP-CE (middle) and MDP-GA (bottom) methods, while the right column plots show the resultant error uncertainties of the survey AUV due to the ranging information broadcast by the beacon vehicle along its planned paths. Without the supporting beacon vehicle, the error uncertainty of the survey AUV is expected to grow linearly without bound. However, it can be seen that the position error uncertainty of the survey AUV is kept at around five meters throughout

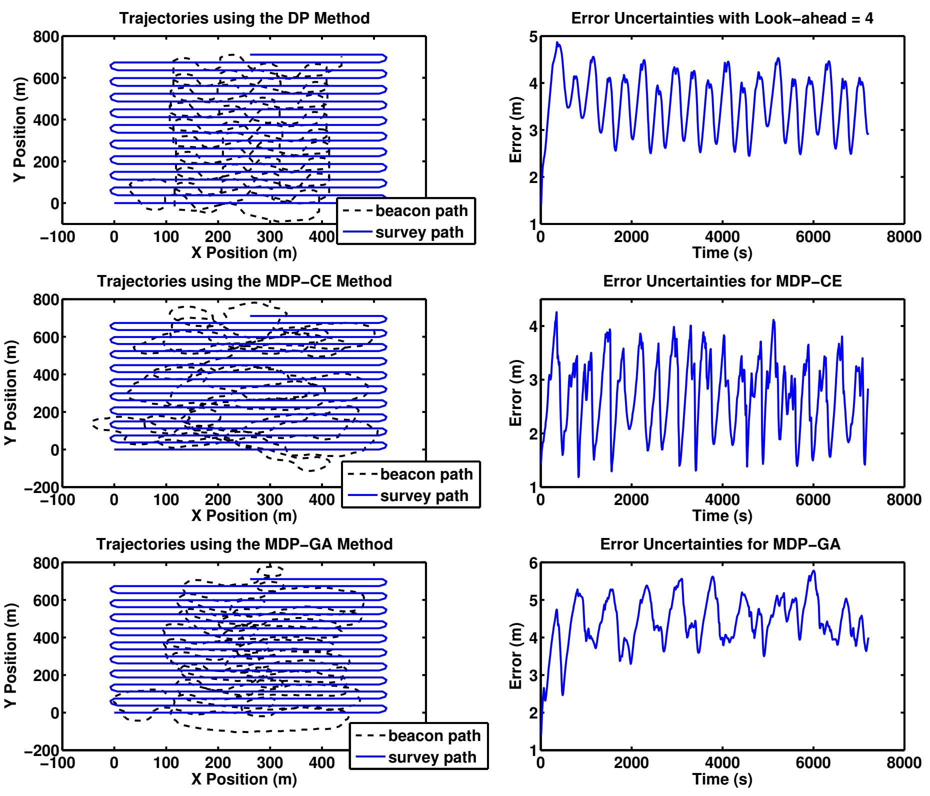


FIGURE 3.8: Simulated runs using the DP, MDP-CE and MDP-GA cooperative path planning algorithms. Left column plots: Lawn-mowing paths of the survey AUVs (blue lines) and the cooperative trajectories planned by the beacon vehicles (dotted lines) using the different planning algorithms. Right column plots: The error uncertainties of the survey AUVs tracked by the beacon vehicles. The beacon vehicle managed to minimize the error uncertainty of the supported survey AUV.

the simulation. Besides, the results are comparable using the paths generated by the different algorithms. We also observe that the beacon vehicle positions itself within the mission area during its course of supporting the survey AUV. The beacon vehicle seems to have "learned" that by keeping a close distance to the survey AUV, the chance for it to achieve maximum change in absolute angle with respect to the survey AUV's location, is higher.

3.5.2 Supporting Multiple Survey AUVs

In the second simulation scenario, we look into having a single beacon vehicle to support multiple survey AUVs. Two AUVs are put into a surveying mission where they are required to navigate in a lawn-mower pattern adjacent to each other as shown in the left column of Fig. 3.9, with an area of around 400 m by 700 m. Regardless of the number of survey AUVs, the DP method computes the cost function over all the supported survey AUVs and produces a single desired heading for the beacon vehicle. However, since the MDP methods were trained to support single survey AUV, the beacon vehicle generates one desired heading with respect to each of the survey AVUs using their respective policy table.

Since choosing one of the desired headings may reduce the accumulated error of one survey AUV while increasing the other, care has to be taken while making the final decision. One of the factors that affects the beacon vehicle's capability in achieving the maximum change of absolute bearing with respect to the survey AUVs' locations, is to maintain close distance with all the vehicles it is supporting. However, this is impossible for the case of multiple survey AUVs where during a surveying mission, survey AUVs may navigate far apart from each other. For the case of supporting two survey AUVs, the beacon vehicle generates two desired headings, one for each of the survey AUVs, using the policy table. The best strategy for the beacon vehicle to maintain close distance to all the supported survey AUVs is to choose, between the two desired headings, the heading that will navigate it in the proximity of the centroid location among survey AUV team. Different strategies can be applied to produce different results [54, 55]. Thorough studies of the strategies' performance are beyond the scope of the thesis.

Fig. 3.9 shows the resultant trajectories (dotted lines) planned by the beacon vehicle (left column) and the corresponding error uncertainties of supported survey AUVs (right column), using the DP, MDP-CE and MDP-GA methods. Although having to support more than one AUV, the beacon vehicle still managed to minimize the error uncertainties of the survey AUVs at around five meters. The results seem to show all three planning methods alternately minimize the error uncertainties of the survey AUVs, even though this "behavior" was not explicitly implemented in the planning algorithm.

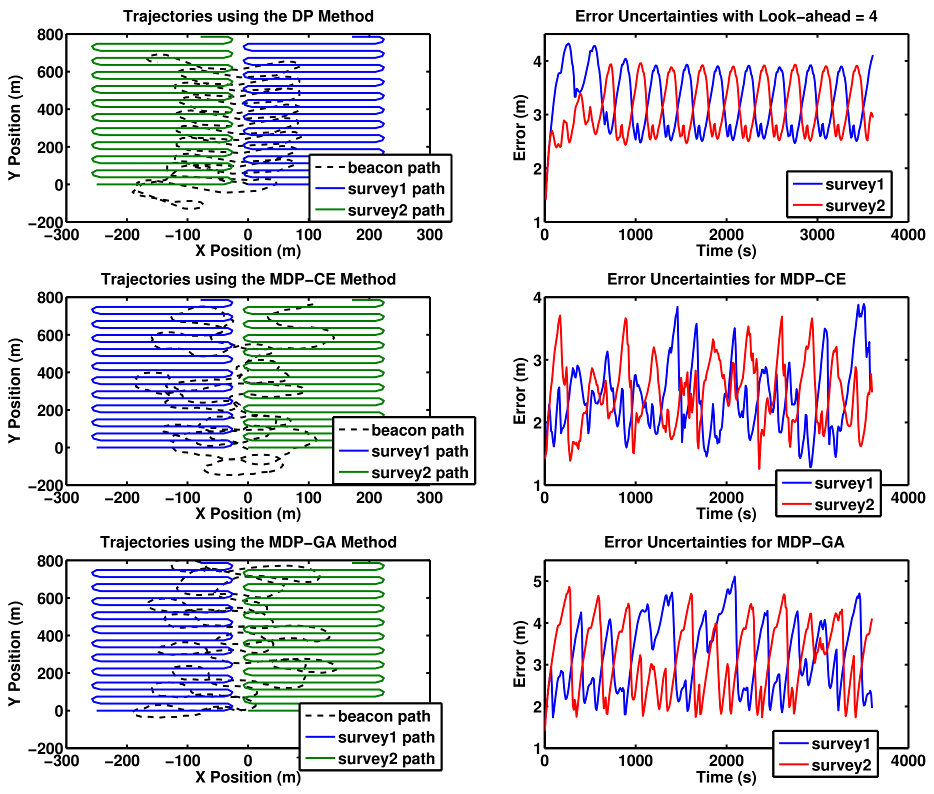
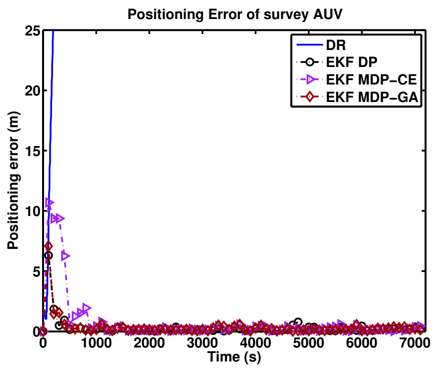


FIGURE 3.9: Simulated runs of single beacon vehicle supporting multiple survey AUVs using the DP, MDP-CE and MDP-GA cooperative path planning algorithms. Left column plots: Trajectories of beacon vehicles in supporting multiple survey AUVs. Right column plots: Error uncertainties of the supported survey AUVs. Similar to the results obtained via the DP method, the MDP-CE and MDP-GA methods alternately minimize the error uncertainties of the supported survey AUVs.

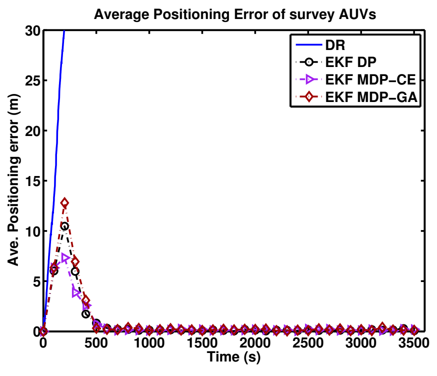
Although the policies of both the MDP methods were trained through stochastic optimization techniques and require much lower computational load when applied online, their performance in minimizing the error uncertainties of the supported survey AUVs are comparable with that of the DP method.

3.5.3 Position Estimation of the Survey AUV

The error uncertainties of the supported survey AUVs presented in the previous section were tracked by the beacon vehicle. For position estimation, the survey AUV must take into account the range measurements as well as the received position of the beacon



(a)



(b)

FIGURE 3.10: The results of the survey AUV's position estimation using the EKF, based on the beacon vehicle's paths planned with the DP, MDP-CE and MDP-GA algorithms. (a) Positioning errors of the survey AUV using beacon trajectories of Fig. 3.8. (b) Average positioning errors of the two supported survey AUVs using beacon trajectories of Fig. 3.9.

vehicle to estimate its position. Using the trajectories planned by the beacon vehicle, we perform position estimation of the survey AUV using the Extended Kalman Filter (EKF) presented in [56]. A tidal current is simulated and is randomly selected between 0.1 and 0.2 m/s. The EKF's process model tracks the vehicle's position and tidal current offset, while the measurement model fuses the range measurement whenever it is broadcast by the beacon vehicle, to improve the tracking accuracy. The survey AUVs are assumed to not be equipped with DVL which measures the vehicles' velocities. The main objective of this section is to assess the performance of the beacon trajectories in minimizing the positioning errors of the supported survey AUVs. For detailed derivation of the EKF, we refer the interested reader to [56] .

Fig. 3.10(a) shows the positioning error of the survey AUV based the beacon trajectories shown in Fig. 3.8 while Fig. 3.10(b) shows the average positioning errors over the two survey AUVs based the beacon trajectories shown in Fig. 3.9. As can be seen, without the velocity and range measurement, the positioning error of the survey AUV using the DR method grows unbounded. On the contrary, the range measurements broadcast by the beacon vehicle along the trajectories planned by the proposed algorithms managed to keep the positioning error of the survey AUVs low throughout the mission. More importantly, it also shows that both the MDP-CE and MDP-GA algorithms performed equally well as the DP method. Since there was no tidal current information at the beginning of the simulation, the positioning error of the survey AUV increased slightly. However, the EKF started to track the tidal offset once the range measurements were received from the beacon vehicle.

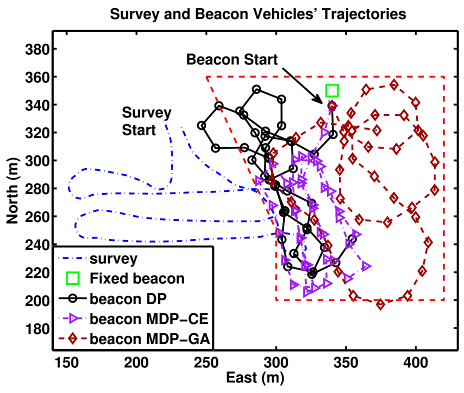
3.6 Field Experiments

We next present performance estimates based on the survey AUV's data obtained during field experiments with a simulated beacon vehicle and range measurements. The field experiment provided us with valuable navigational data collected from the survey AUV's proprioceptive sensors that could not otherwise be reproduced in the simulation environment. In addition, the environmental uncertainties due to tidal current around the coastal waters also allowed us to test the robustness of the algorithms in handling unexpected natural events.

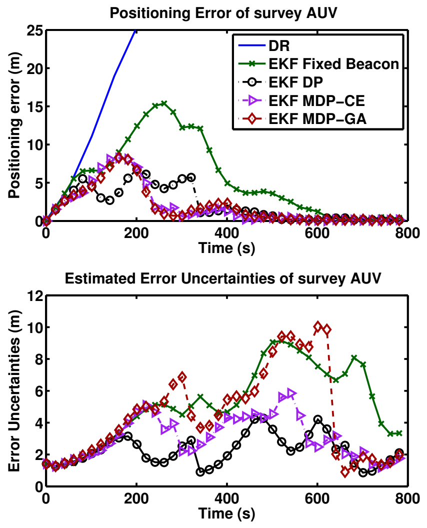
3.6.1 Cooperative Positioning with Geo-fence

Often time the AUVs have to operate within a geo-fence due to safety reason or operational constraint. A geo-fence is a confined region marked within a mission area where the AUVs are allowed to carry out their missions. In this field experiment, two STARFISH AUVs [3] were deployed in the Pandan Reservoir, Singapore to perform cooperative positioning, where one assumed the role of beacon vehicle and the other one as survey AUV. Both the AUVs were executing their missions on the surface so that their GPS position logs can be used as ground truth for offline tracking. Due to the close proximity to the launching platform and to avoid any potential collision between the AUVs, a geo-fence is defined for the beacon vehicle as shown in Fig. 3.11(a).

During the experiment, we managed to execute a number of cooperative missions using the MDP-CE method, with one of them shown in Fig. 3.11(a). To investigate the performance differences, we also simulated the mission with three different types of



(a)



(b)

FIGURE 3.11: (a) The resultant beacon paths planned within the geo-fence boundary (red dotted-box). (b) Positioning errors (top) and estimated error uncertainties (bottom) tracked by the EKF, using the resultant beacon paths. Compared to the result of a fixed beacon, the positioning errors were lower based on the offline simulations. However, the estimated error uncertainties were higher at some points, especially for the MDP-GA method.

beacon aids, by using the same survey vehicle's navigational data collected during the field experiment. The same EKF presented in [56] was used for the offline tracking of the survey AUV. The positioning errors of the survey AUV were comparable for all the beacon aids since the mission is relatively short and there is no tidal current in the reservoir. To illustrate the advantage of having mobile beacons, we repeated the offline tracking with a simulated tidal current offset of 0.1 to 0.2 m/s, randomly selected for each simulation run. A total of 100 simulations were performed for the same beacon paths, but with different realization of the measurement noise.

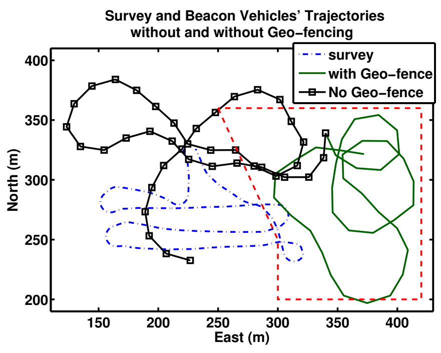
3.6.1.1 Experimental Results

The resultant positioning errors of the survey AUV were shown in Fig. 3.11(b) (top). Again, without any range measurement, the survey AUV with the DR method grew unbounded. Even though confined within the geo-fence, the beacon vehicle's trajectories were able to minimize the survey AUV's positioning error, except at the initial stage of the mission where there was no tidal information and the filter was just beginning to track it through ranging measurements from the beacon vehicle. On the other hand, the range measurements from the fixed beacon allowed the positioning error to grow much higher before being reduced towards the end of the mission. The result suggests that having range information from a fixed location is only able to help in reducing the positioning error opportunistically, depending on the survey AUV's path pattern; if the range measurements are received along the same absolute bearing between the survey AUV and fixed beacon's locations, the positioning error could grow in the tangential direction.

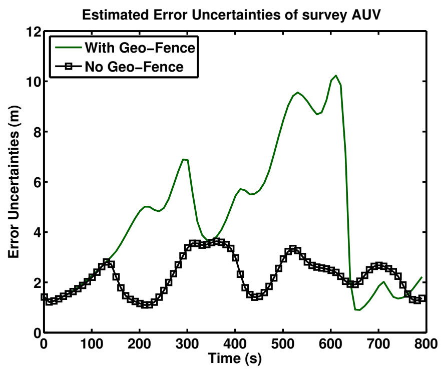
However, the plot of estimated error uncertainties (Fig. 3.11(b) (bottom)) shows relatively poor performance when the beacon vehicles are confined within the geo-fence, especially for the case of the MDP-GA method. This is because the beacon vehicle is forced to take sub-optimal actions that navigate itself within the geo-fencing box, potentially disregarding the best state-action mapping encoded in the planning policy. To further illustrate the effect of geo-fencing, we repeated the simulation with the MDP-GA method, but without the geo-fence constraint. Fig. 3.12(a) shows the resultant path of the beacon vehicle without the constraint of the geo-fence, with the previous geo-fenced beacon path overlaid for comparison purposes. It can be seen that the beacon vehicle could navigate more freely around the survey AUV in order to attain change of absolute bearing between the vehicles for every consecutive range information broadcast. The improvement due to this geo-fence free beacon trajectory can be clearly seen in Fig. 3.12(b) where estimated error uncertainty of the survey AUV is relatively lower than the case of geo-fenced for the most part of the mission.

3.6.2 Cooperative Positioning around Coastal Waters

On July 9, 2011, a field trial was conducted near Serangoon Island, Singapore using the STARFISH AUV (Fig. 3.13). The STARFISH AUV [3] was deployed to perform a simple surface surveying mission with GPS position available as ground truth. The navigational data were collected and used as the pre-planned path for the simulated beacon



(a)



(b)

FIGURE 3.12: (a) The trajectories of the beacon vehicle with and without geo-gence, using the MDP-GA method. (b) The corresponding estimated error uncertainties of the survey AUV tracked by the EKF.

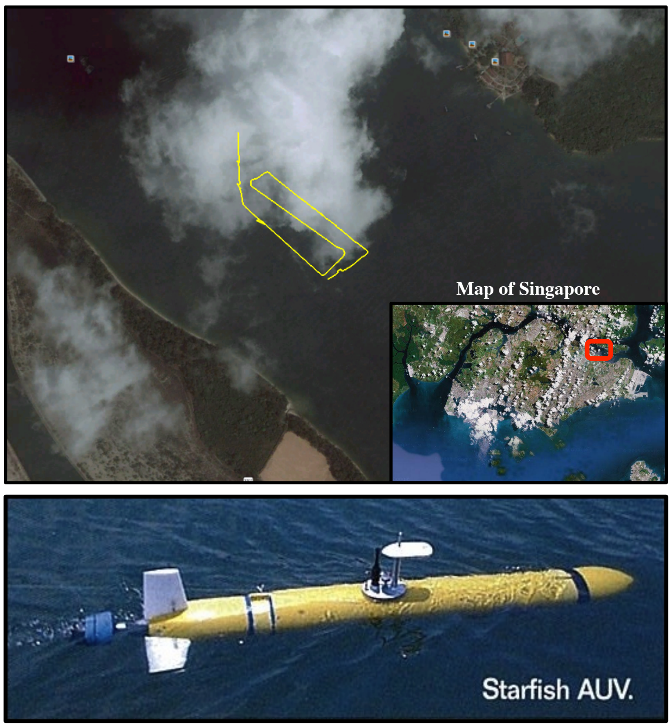


FIGURE 3.13: Top: Field trial near the Serangoon Island, Singapore. Bottom: The STARFISH AUV [3].

vehicle. In the simulation, the position of the survey AUV was estimated (assuming no GPS) using only compass measurements and simulated acoustic range updates. Only the first few GPS updates were used to initialize the position of the survey AUV.

The acoustic range updates were assumed to occur at a fixed interval $\tau = 10$ seconds for the simulation studies reported in Section 3.5. However, the updates may be sporadic in reality due to the communication packet loss and this may affect the computation of the position estimate and the error estimate of the survey AUV. From the measured acoustic ranging statistics during the field trial, the range updates occurred between 5 and 20 seconds with some exceptions due to packet loss. In this part of study, we simulated range updates received by the survey AUV to occur at any time uniformly distributed between 5 and 20 seconds, with a packet loss probability of 0.46, to match the statistics collected from a recent field trial. The difference in terms of the range update frequency allows us to investigate the robustness of the resultant path planning algorithms in handling the uncertainty associated with the acoustic communication.

For the comparison purposes, simulations were conducted with five different types of ranging aids each transmitted from a single beacon. The ranging aids used were: single fixed beacon, circular moving beacon (CMB), cooperative beacon (beacon vehicle) where its paths were planned using the DP method reported in [22], and both the MDP-CE and MDP-GA methods resulted from this work. The fixed beacon remained stationary throughout the mission, while the circular moving beacon maneuvered in a circular pattern around the center of the survey site. Finally, we also conducted a simulation where the survey AUV relied solely on the DR method for navigation to further illustrate the rate of position error growth without range measurement.

In total, 100 runs were conducted for the each simulation scenario. Throughout the simulation studies, we assumed the survey AUV was not equipped with a DVL. The only available measurements for position estimation were the compass and the acoustic ranges. The survey AUV's position for each of the simulated scenario was estimated with the same EKF as the previous sections. Detailed derivation of the filter can be found in [56]. Similar to the simulation study, four levels of look-ahead (LA-4) strategy and three discrete turn angles ($A = 3$) were used for the DP method while the policy table trained with the MDP-CE and MDP-GA methods were used for beacon vehicle's path planning.

3.6.2.1 Experimental Results

Fig. 3.14(a) and 3.14(b) shows the real path of the survey AUV and the resultant beacon paths generated with different beacon types while Fig. 3.14(c) shows the accumulated positioning errors of the supported survey AUV.

Throughout the mission, a total of 77 simulated acoustic range updates were received by the survey AUV (simulated with packet lost probability of 0.46). It can be seen that the DR method without range measurement produced the worst position estimation for the survey AUV. Since the GPS updates were only available at the beginning, the position estimation using the DR method started to drift uncontrollably throughout the rest of the mission. However, the error growth rate was different at different mission legs depending on the prevalent tidal current. The tidal current during the mission varied significantly in different mission legs. Fig. 3.15(a) shows the real trajectory of the survey AUV with time noted at every 200 seconds. The survey AUV was commanded to thrust at a constant level of 70% throughout the whole mission, which gives about 1.5 m/s relative speed with respect to the water. Since the observed displacement was not 300 m for every 200 seconds, it is clear that there was some tidal current slowing down or speeding up the AUV along its heading direction, in addition to the local non-tidal current variations along the channel, as reported in [67] and estimated in Fig. 3.15(b): in the first leg (200 to 400 seconds), there was a mild current stream (about 0.5 knots) against the AUV's direction; in the second leg (600 to 800 seconds), the effect of ocean current was along the AUV's heading direction, thus increased its effective speed; in the last leg (1000 to 1600 seconds), there was a strong current stream (up to 2 knots) slowing down the AUV and caused it to move only about 100 m for every 200 seconds interval.

In Fig. 3.14(c), the fixed beacon and the CMB performed poorly in correcting the positioning error of the supported survey AUV since the changes of relative aspect between the vehicles were small during the acoustic range updates. This caused the tidal current estimation in the state vector to become worse. The poor tidal current

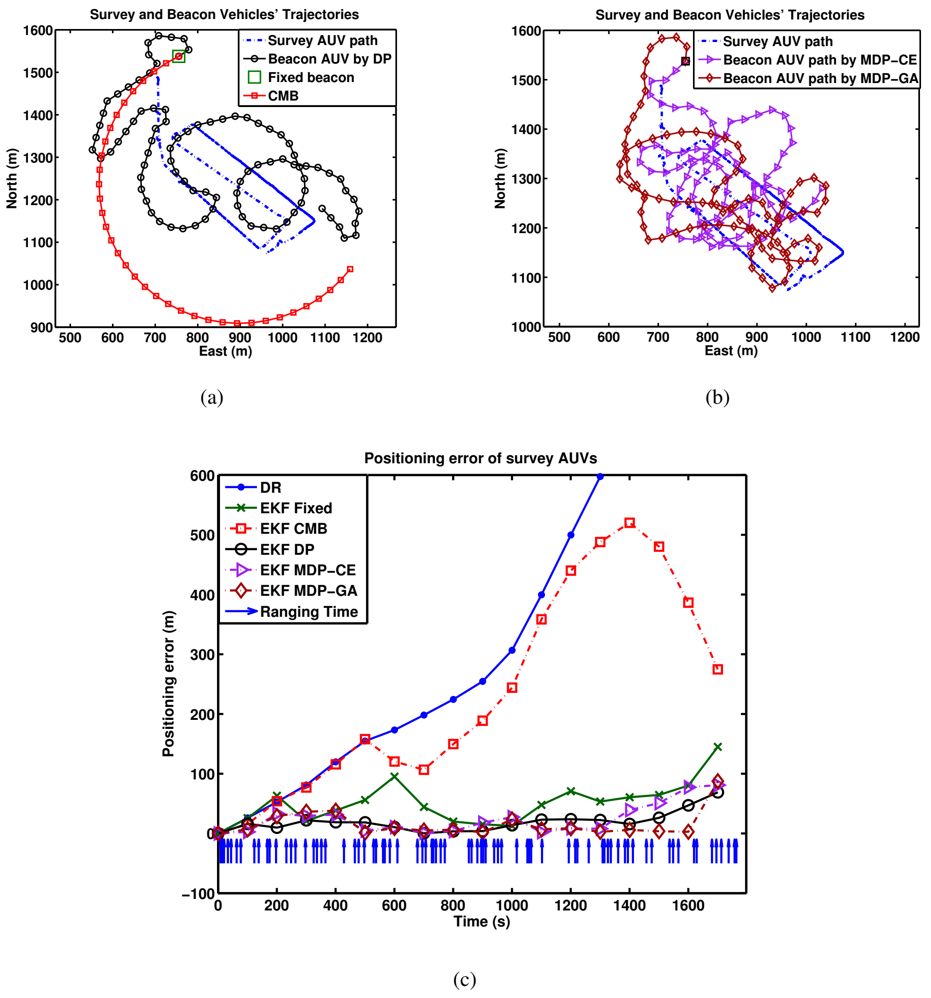


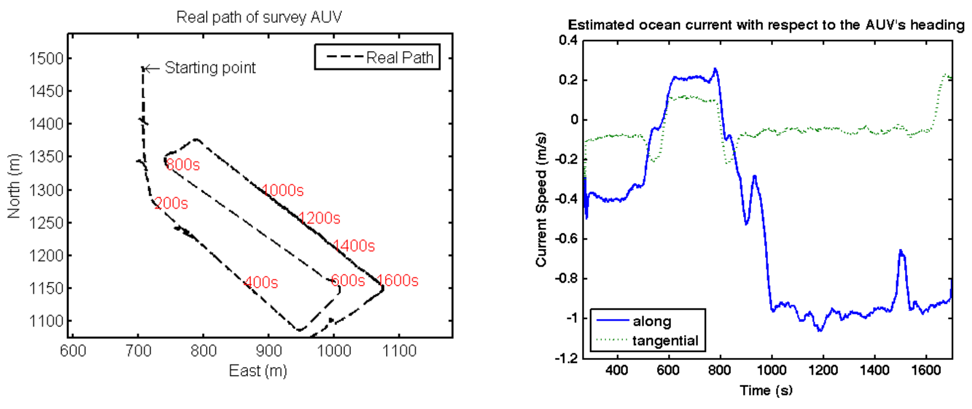
FIGURE 3.14: Performance estimate using the field data collected on July 9, 2011 near the Serangoon Island, Singapore. The beacon vehicle starts at an offset of [50,50] meters from the survey AUV. (a)-(b) Figure showing the planned paths by varies types of beacons overlaying the pre-planned path of the survey AUV. (c) Positioning errors of the survey AUV supported by different types of beacons. The vertical lines (blue) at the bottom of the plot show the time when there is an acoustic range update.

estimation in turn resulted in poor estimation of future positions. This feedback cycle escalated the growth of positioning error in the survey AUV. However, when the ocean current was almost zero and the survey AUV maneuvered in favor of the fixed beacon's location (600 to 800 seconds), the positioning errors were significantly reduced by its range updates. This observation further supported the claim that the location of the fixed beacon is one of the important factors that determines the performance of the beacon.

TABLE 3.6: POSITIONING ERRORS INCURRED BY VARIOUS METHODS.

Methods	DR	Fixed beacon	CMB	DP	MDP-CE	MDP-GA
Max Errors (m)	553.6	241.0	521.7	79.8	104.0	108.6
% o.d.t*	35.57	15.49	33.52	5.13	6.69	6.98
Ave. Errors (m)	343.7	54.7	233.4	19.1	26.5	16.1
% o.d.t*	22.09	3.51	15.00	1.23	1.71	1.03

* over the distance travelled of $\sim$ 1.5 km



(a) (b)

FIGURE 3.15: The real path of the survey AUV during the field trial on the July 9th, 2011 and the estimates of the ocean current in the AUV's body-frame. The AUV encountered a strong ocean current stream from time 1000^{th} seconds onwards. (a) survey AUV's executed path with time stamps of every 200 seconds. (b) Ocean current estimated in the AUV's body-frame.

Both the DP and MDP methods (both MDP-CE and MDP-GA) kept the positioning error of the survey AUV fairly small throughout the mission, even though they were under the effect of varying tidal currents. This demonstrated the robustness of both the DP and MDP methods in handling the environmental uncertainties. For the entire survey path of around 1.5 km, the survey AUV position error with both the fixed beacon and the CMB reached a maximum of around 16% and 34%, while the DP and MDP methods yielded a maximum error well below 7%. The average errors accumulated over the entire mission were around 3.5% and 15% for both the fixed beacon and the CMB, and 1.2%, 1.7% and 1.03% for the DP, MDP-CE and MDP-GA methods respectively. Detailed position error estimates using various methods are shown in Table 3.6.

3.7 Discussion

The solution to single beacon cooperative positioning problem consists of multiple local minima. Even though the paths generated based on different algorithms may look different, as can be seen from the results of both the simulation and field experiment sections, they are equally good in minimizing the estimated position error of the supported survey AUVs.

Although the DP method performed slightly better than the MDP methods, the differences are not significant. However, the MDP methods have an advantage over the DP method in terms of the computational complexity in planning the beacon vehicle's path. In the following, we present the comparisons of the algorithms' complexities in terms of their computational loads and the size of the resultant policy tables.

Computational Load and Policy Table

Let Γ be the length of the mission and A be the size of decision space. The computational load for the DP method in generating an optimal route using the greedy strategy to support a single survey AUV is $\mathcal{O}(\Gamma A)$ and only increases linearly with the length of the mission. However, its computational load increases exponentially to $\mathcal{O}(\Gamma A^{L+1})$ if L-level of look-ahead strategy is employed, which can be significantly higher than that of the greedy strategy (equivalent to 0-level look-ahead strategy). In supporting multiple survey AUVs, the computational load increases to $\mathcal{O}(\Gamma A^{L+1} M)$, where M is the number of survey AUVs.

On the other hand, the computational loads are only $\mathcal{O}(\Gamma)$ for the MDP-CE method and only $\mathcal{O}(\Gamma N_{Rstate})$ for the MDP-GA method, where N_{Rstate} is the number of RStates. In supporting multiple vehicles, the loads increase to $\mathcal{O}(\Gamma M)$ and $\mathcal{O}(\Gamma N_{Rstate} M)$ respectively. Although the computational loads are significantly lower than the DP method, the process of the decision making in the MDP methods is heuristic, and generates sub-optimal routes. Furthermore, the policy learning step can be time consuming even though it can be performed offline.

The DP method does not use a policy table as the optimal route is computed online at every τ seconds. Given the discretization shown in Table 3.2, the MDP-CE method carries a policy table with 1119744 states. In contrast, the MDP-GA method only carries a policy table of 7040 states (based on discretization shown in Table 3.3), yet it managed to achieve comparative performance.

3.8 Summary

In this chapter, we developed a cooperative path planning algorithm for a beacon vehicle to support one or more survey AUVs. The beacon vehicle utilizes acoustic range measurements to minimize the survey AUVs' accumulated positioning errors during underwater navigation. The algorithm planned the beacon vehicle's path around the survey AUVs such that when range information is exchanged, the position errors of the supported survey AUV can be kept small.

We formulated the path planning problem within a MDP framework and proposed two different gradient-free policy search techniques to learn the planning policy, taking into account the survey AUVs' accumulated position errors, relative geometries and inter-vehicle distances between the vehicles. The policy of the MDP-CE method was searched using a combinatorial optimization technique while the policy of the MDP-GA method was learned through natural evolution.

Simulation studies using vehicle data collected from field experiments showed that the proposed algorithm kept the position errors of the supported survey AUVs small throughout the mission runs. In addition, the algorithm was also shown to be robust in handling varying range update rates as well as environmental uncertainties. While both the techniques greatly reduced the computational load compared to previous published approaches, the MDP-GA method only requires a significantly smaller policy table, yet managed to perform comparatively well in terms of minimizing the survey AUVs' position errors.

Chapter 4

Cooperative Bathymetry-based Localization

Equipped with underwater modems, an AUV is able to estimate inter-vehicle range using the time-of-flight of an acoustic signal transmitted by another AUV. The range measurements among the AUVs provide relative geometrical constraints on their position estimates. However, such an approach requires at least a geo-referenced position information (e.g from GPS), and the information passing within the vehicle network to be acyclic. When the information passing within the network is allowed to be cyclic, the position estimates of the vehicles become correlated. The cross-correlations, if not taken into account, can cause the issue of overconfidence on the position estimation and result in filter divergence [68]. Even though the issue can prevented by maintaining the filter's consistency [69], it requires careful bookkeeping to track the origins of information broadcast.

Although underwater localization using bathymetry information has been reported in numerous literatures, its application is generally limited to a single sensor-rich AUV equipped with high accuracy navigational sensors and multi-beam echo sounders. In this chapter, we develop a cooperative bathymetry-based localization approach using a team of sensor-limited AUVs, equipped only with a single-beam altimeter, a depth sensor and an acoustic modem. The localization of the individual AUV is achieved via decentralized particle filtering, with the filter's measurement model driven by the AUV's altimeter measurements and ranging information obtained through inter-vehicle communication. We validate the feasibility of the decentralized filter through simulation studies, using randomly generated trajectories, as well as trajectories executed by the AUVs during field experiments. We also perform empirical analysis on the factors that affect the filter performance. Please refer to [70] for related publication.

4.1 The Concept of Cooperative Bathymetry-based Localization

An AUV that is capable of measuring only a single altitude measurement at every sampling time step can not localize itself effectively within a given bathymetry map, due to the multiple occurrences of similar terrain information in the map. However, a team of these AUVs that are also capable of estimating the inter-vehicle ranges may use this information to impose a geometrical constraint on the vehicles' altitude measurements. The set of geometry constrained measurements reduces, if not eliminates, the likelihood of multiple occurrences of similar terrain information in the map and allows each vehicle to estimate their individual positions. This is the main idea behind the cooperative bathymetry-based localization.

Cooperative bathymetry-based localization involves a team of low-cost, sensor-limited AUVs. The localization of the individual AUVs is based on bathymetry information measured along their trajectories, complemented with the ranging information obtained by the inter-vehicle communication among the vehicles in the team. This approach is inspired by the fact that given the vehicles' estimated locations, the relative geometry among the vehicles needs to be consistent with the bathymetry information measured at those locations. In contrast with the cooperative positioning, the bathymetry map acts as the source of geo-referenced position information, replacing the need for the vehicles to access a GPS signal.

Each of the vehicles in the team runs (locally) a decentralized particle filter to estimate their respective positions. The filter's process model is driven using only the

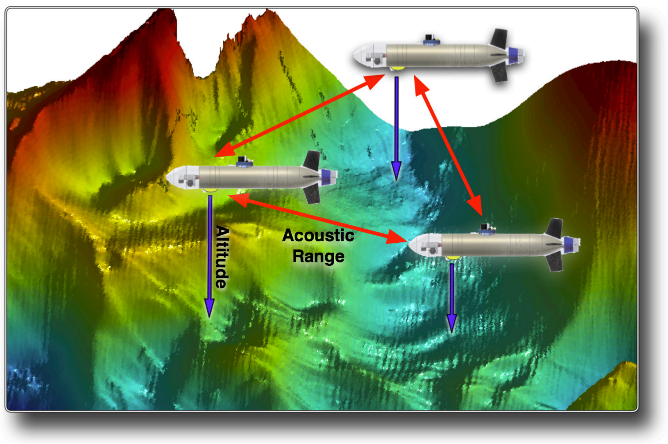


FIGURE 4.1: Multi-AUV cooperative localization using altimeter measurements and inter-vehicle acoustic communication.

AUV's control inputs and a model that predicts the AUV velocity based on the thruster control input and an onboard compass. The corresponding measurement model is updated by comparing the vehicle's water depth (altitude + depth) measurements against the bathymetry information. At every pre-scheduled period of time, the AUVs broadcast their filters' local sufficient statistics (belief) sequentially, via acoustic communication. Once received by other vehicles in the team, the information and the inter-vehicle range are fused into their respective measurement models to influence the filter's particle distribution.

Fig. 4.1 further illustrates the concept of the proposed cooperative localization. The decentralized formulation allows the approach to be scaled up with the number of vehicles without increasing the computational complexity. Since the individual vehicles' position and error uncertainties are estimated solely from their own bathymetry measurements between the times of acoustic communication, the proposed cooperative localization method alleviates the issue of overconfidence on the position estimates, even if the filters' belief broadcasts within the vehicle network are cyclic.

4.2 Problem Formulation

4.2.1 Process and Measurement Models

Let x, y be the easting and northing position of the vehicle, and c_x, c_y be the ocean current in the easting and northing direction. Furthermore, let t be the time step and the elapsed time between step t and $t+1$ be Δt. The discrete-time process model used for the vehicle is described by :

$$\pi_{t+1} = \mathbf{F}\pi_t + \mathbf{G}_u \mathbf{u}_t + \zeta_t \tag{4.1}$$

where $\pi = [x, y, c_x, c_y]^\top$ is the state vector, $\mathbf{F}$ and $\mathbf{G}_u$ are the state transition and control-input matrices respectively. $\mathbf{u}_t = [u_x, u_y]_t^\top$ is the control input that determines the AUV's motion. The control input (derived from commanded heading and thrust) is the commanded velocity at which the AUV should move in the easting and northing direction for the time step. The commanded heading is subjected to the same maximum turning rate as modeled in Equation (3.5) and (3.6). ζ_t is the process noise, modeled as an additive zero-mean Gaussian ($\zeta_t \sim \mathcal{N}(0, \sigma_\zeta^2)$). The corresponding discrete-time measurement model is

$$\mathbf{y}_t = \mathbf{h}(\pi_t) + \xi_t \tag{4.2}$$

where ξ_t is the measurement noise, modeled as an additive zero-mean Gaussian ($\xi_t \sim \mathcal{N}(0, \sigma_\xi^2)$). $\mathbf{y}_t$ represents the vehicle's measurement at time t while $\mathbf{h}(\pi_t)$ is the non-linear function that relates the bathymetric information at state π_t to the measurement.

4.2.2 Marginalized Particle Filter

Let N represent the number of particles used for the particle filter, π_t^i being the ith particle at time t. We adopt the marginalized particle filter (MPF) described in [71] and decompose the state vector into two parts:

$$\pi = \begin{bmatrix} \pi^{\text{pf}} \\ \pi^{\text{kf}} \end{bmatrix}. \tag{4.3}$$

where $\pi^{\text{pf}} = [x, y]^\top$ represents the position of the vehicle estimated by Particle Filter (PF) and $\pi^{\text{kf}} = [c_x, c_y]^\top$ represents the ocean current bias estimated by a Kalman Filter (KF). The resulting state-space model becomes:

$$\begin{bmatrix} \pi^{\text{pf}}_{t+1} \\ \pi^{\text{kf}}_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I}_{2\times2} & \mathbf{F}^{\text{pf}} \\ 0_{2\times2} & \mathbf{F}^{\text{kf}} \end{bmatrix}}_{\mathbf{F}} \begin{bmatrix} \pi^{\text{pf}}_{t} \\ \pi^{\text{kf}}_{t} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{G}^{\text{pf}}_u & 0_{2\times2} \\ 0_{2\times2} & 0_{2\times2} \end{bmatrix}}_{\mathbf{G}_u} \begin{bmatrix} \mathbf{u}_t \\ 0_{2\times1} \end{bmatrix} + \begin{bmatrix} \zeta^{\text{pf}} \\ \zeta^{\text{kf}} \end{bmatrix}. \tag{4.4}$$

where $\mathbf{F}^{\text{pf}} = \mathbf{G}^{\text{pf}}_u = \begin{bmatrix} \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$ and $\mathbf{F}^{\text{kf}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Since we assume that the vehicle does not equipped with any exteroceptive navigational sensor, tracking the ocean current will improve the accuracy of the vehicle's position propagation.

Prediction:

The decomposed state vectors are propagated from time t to time $t+1$ with :

$$\pi^{\text{pf},i}_{t+1} = \pi^{\text{pf},i}_{t} + \mathbf{F}^{\text{pf}}\pi^{\text{kf},i}_{t} + \mathbf{G}^{\text{pf}}_u \mathbf{u}_t + \zeta^{\text{pf}}_t. \tag{4.5}$$

where $\zeta^{\text{pf}}_t = \mathscr{N}(0, \mathbf{F}^{\text{pf}}\mathbf{P}^{\text{kf}}_{t|t-1}(\mathbf{F}^{\text{pf}})^\top + \mathbf{Q}^{\text{pf}})$ with $\mathbf{Q}^{\text{pf}}$ being the process noise intensity matrix and $\mathbf{P}^{\text{kf}}_{t|t-1}$ denotes the prediction of the estimate covariance of the ocean current from time $t-1$ to t.

The ocean current is estimated through the following process and measurement equations:

$$\begin{aligned}\pi^{\mathrm{kf}}_{t+1} &= \mathbf{F}^{\mathrm{kf}}\pi^{\mathrm{kf}}_{t} + \zeta^{\mathrm{kf}}_{t}. \\ \mathbf{Z}_t &= \mathbf{F}^{\mathrm{pf}}\pi^{\mathrm{kf}}_{t} + \mathbf{G}^{\mathrm{pf}}_{u}\mathbf{u}_t + \zeta^{\mathrm{pf}}_{t}.\end{aligned} \tag{4.6}$$

For each of the particles, the estimation $\mathbf{Z}^i_t$ corresponds to the Euclidean distance between the particle's position, $\pi^{\mathrm{pf},i}_{t}$ and its immediate successor, $\pi^{\mathrm{pf},i}_{t+1}$, as predicted in (4.5). Thus, the innovation $\mathbf{Z}^i_t - (\mathbf{F}^{\mathrm{pf}}\hat{\pi}^{\mathrm{kf},i}_{t|t-1} + \mathbf{G}^{\mathrm{pf}}_{u}\mathbf{u}_t)$ is the difference between the distance measured, $\mathbf{Z}^i_t$ and the predicted distance traveled by the vehicle due to ocean current bias, $\mathbf{F}^{\mathrm{pf}}\hat{\pi}^{\mathrm{kf},i}_{t|t-1}$ and vehicle's control input, $\mathbf{G}^{\mathrm{pf}}_{u}\mathbf{u}_t$. Within a KF's formulation, the ocean current bias's state vector of each particle is propagated with:

$$\begin{aligned}\pi^{\mathrm{kf},i}_{t+1} &= \hat{\pi}^{\mathrm{kf},i}_{t+1|t} \\ &= \mathbf{F}^{\mathrm{kf}}\hat{\pi}^{\mathrm{kf},i}_{t|t}.\end{aligned} \tag{4.7}$$

where the KF's update expression of the ocean current bias estimate and the corresponding covariance matrices are [71]:

$$\begin{aligned}\hat{\pi}^{\mathrm{kf},i}_{t|t} &= \hat{\pi}^{\mathrm{kf},i}_{t|t-1} + \mathbf{K}_t\mathbf{V}_t. \\ \mathbf{V}_t &= \mathbf{Z}^i_t - (\mathbf{F}^{\mathrm{pf}}\hat{\pi}^{\mathrm{kf},i}_{t|t-1} + \mathbf{G}^{\mathrm{pf}}_{u}\mathbf{u}_t). \\ \mathbf{K}_t &= \mathbf{P}_{t|t-1}(\mathbf{F}^{\mathrm{pf}})^{\top}\left[\mathbf{F}^{\mathrm{pf}}\mathbf{P}_{t|t-1}(\mathbf{F}^{\mathrm{pf}})^{\top} + \mathbf{Q}^{\mathrm{pf}}\right]^{-1}. \\ \mathbf{P}_{t|t} &= (\mathbf{I} - \mathbf{K}_t\mathbf{F}^{\mathrm{pf}})\mathbf{P}_{t|t-1}. \\ \mathbf{P}_{t+1|t} &= \mathbf{F}^{\mathrm{kf}}\mathbf{P}_{t|t}(\mathbf{F}^{\mathrm{kf}})^{\top} + \mathbf{Q}^{\mathrm{kf}}.\end{aligned} \tag{4.8}$$

Update:

The update step consists of updating the particle's relative weight (importance) based on its observation. Let w^i_t be the relative weight associated with ith particle at time t, the weight of a particle is updated according to [71] as:

$$w_t^i = w_{t-1}^i . p(\mathbf{y}_t \mid \pi_t^i) \tag{4.9}$$

where $p(.)$ is the likelihood function of the observation $\mathbf{y}_t$ given the particles' predicted states π_t^i and w_0^i is initialized to $1/N$. With the updated weights, a point estimate of the current state $\hat{\pi}_t$ can be estimated through:

$$\hat{\pi}_t^{\text{MMS}} \simeq \sum_i^N w_t^i \pi_t^i \tag{4.10}$$

while the PF's covariance is approximated by:

$$\mathbf{P}_t^{\text{pf}} = \sum_i^N w_t^i (\pi_t^{\text{pf},i} - \hat{\pi}_t^{\text{pf,MMS}}) \cdot (\pi_t^{\text{pf},i} - \hat{\pi}_t^{\text{pf,MMS}})^\top \tag{4.11}$$

and the corresponding Kalman part of the state vector has a covariance estimated by:

$$\mathbf{P}_t^{\text{kf}} = \mathbf{P}_{t|t}^{\text{kf}} + \sum_i^N w_t^i (\hat{\pi}_{t|t}^{\text{kf},i} - \hat{\pi}_t^{\text{kf,MMS}}) \cdot (\hat{\pi}_{t|t}^{\text{kf},i} - \hat{\pi}_t^{\text{kf,MMS}})^\top. \tag{4.12}$$

4.2.2.1 Sampling Importance Resampling

One of the problems with particle filters is degeneracy of particles where only a small percentage of the particles are contributing to the estimation. This happens because as the filter propagates, most of the particles will have small weights as they drift apart. One way to detect the degeneracy problem is to estimate the number of effective samples (N^{eff}) that are currently in the particle set. This N^{eff} indicates how well the current particle set represents the target distribution:

$$N^{\text{eff}} = \frac{1}{\sum_{i=1}^N (w_t^i)^2}. \tag{4.13}$$

Whenever N^{eff} is lower than the resampling threshold (N^{th}), resampling should be performed to generate a new set of particles. In this work, we adopt the sampling threshold value in [72]:

$$N^{\text{th}} = \frac{2N}{3}. \tag{4.14}$$

The resampling steps mentioned above are generally referred to as Sampling Importance Resampling (SIR) [28, 71, 73], and are summarized in Algorithm 4. The type of resampling method used affects the overall computational complexity of a particle filter, and is one of the important considerations because of the limited computational power that a low-cost AUV has onboard. We employed the Residual Resampling reported in [74] for its efficiency in terms of computation time and quality of variance reduction.

However, due to the discrete nature of the particle set, resampling over time leads to another problem called *sampling impoverishment* [75], where the newly generated particle set consists of only the offspring of a small number of particles and could not reflect the true density. To reduce the effect of this, we add randomly generated Gaussian noise (with variance equals to two times the map resolution) to every sample that was chosen more than once.

4.3 Measurement Model for Cooperative Localization

The evaluation of the likelihood function in (4.9) is according to the vehicle's measurement model. For the case of single vehicle localization, the measurement consists of the water depth estimate (AUV altitude measurement + AUV depth measurement) at the location of the AUV. Whenever acoustic communication is available among the vehicles, the measurement model also incorporates the localization information broadcast by other vehicles for the evaluation of the likelihood function.

4.3.1 Localization in Single-vehicle

At every time step, the vehicle performs altitude and depth measurement. Without acoustic ranging and information from peer vehicles, the measurement only consists of the vehicle's water depth information along its trajectory. The vehicle keeps a history of the previous ℓ (where $\ell \geq 1$) time step measurements and the segment of trajectory where these measurements were made. The differences in water depth within this trajectory can then be computed by subtracting each of the measurements from its previous time step's measurement. Fig. 4.2. shows an example of the altitude measurements along a vehicle's trajectory. This approach has the advantage of eliminating the tidal offsets between the time when the bathymetric map was generated and time of mission deployment.

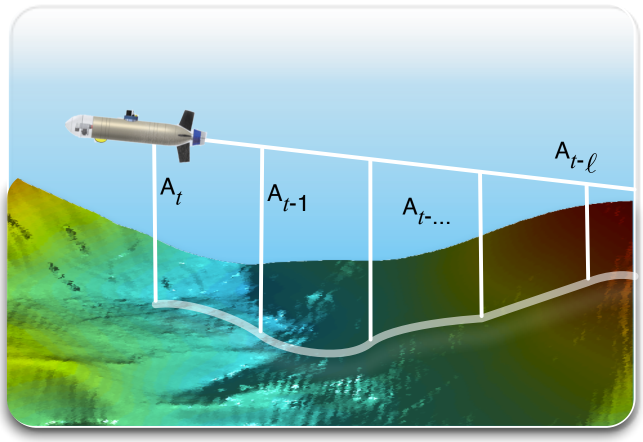
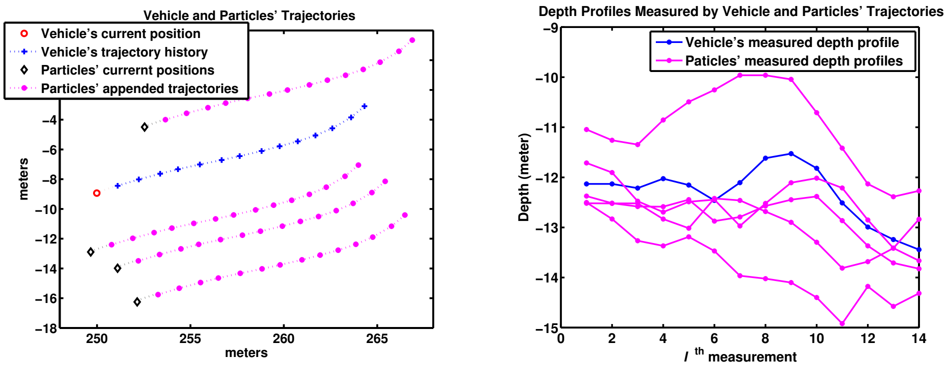


FIGURE 4.2: Altitudes measured along the vehicle's trajectory. The differences in water depth can be calculated by subtracting each of the measurements from its previous time step's measurement.

The weights of the particles are updated based on the likelihood function $p(.)$ of the measurement $\mathbf{y}_t$ given the predicted states π_t^i at every time step. In our case, the segment of trajectory history kept by the vehicle is appended to each of the particles. The corresponding water depth information of the appended particles is obtained from the bathymetric map, using the same measurement interval. As a result, each of the particles has an array of ℓ measurements. An example is shown in Fig. 4.3(a). The measurement model of the filter takes into account the variation between the differences in water depth measured at the particles' predicted locations (black diamonds, with



(a) (b)

FIGURE 4.3: (a) Examples of the vehicle's position (red circle) and its trajectory of length $\ell = 14$ (blue cross). The trajectory is appended to all the particles (black diamonds) forming the particles' trajectories (magenta asterisks). (b) Examples of depth profile measured by the vehicle and the particles' trajectories. The closer the depth profile measured by a particle's trajectory to that of the vehicle's, the higher the weight that is assigned to that particular particle.

measurement noise from section 4.2.1) and the true differences in water depth measured by the vehicle along the trajectory segment (red circle). The smaller the differences, the higher the weight that is assigned to the particular particle. An example of depth profile measured by the particles and the vehicle along a trajectory of ℓ measurements is shown in Fig. 4.3(b). Thus, the likelihood function of the i^{th} particle is:

$$p(\mathbf{y}_t \mid \pi_t^i) = p(\mathbf{y}_{t:t-\ell} - h(\pi_{t:t-\ell}^{\text{pf},i})) \tag{4.15}$$

where the subscript $t : t-\ell$ denotes an array of the differences in water depths measured from time $t-\ell$ to the current time t. This approach reduces the probability of multiple occurrence of sea bottom topology that occurs if the localization is performed using a single-beam altimeter. According to the study in [26], the higher the number of beams used, the lower the number of false-likelihood positions. However different from the multi-beam sonar where the distances between the measurements are fixed due to the fixed sensors layout, appending the individual measurements made along a vehicle's trajectory introduces accumulative errors in the distances between the measurements, due to interpolating errors and sensor noise.

4.3.2 Localization in Multiple Vehicles

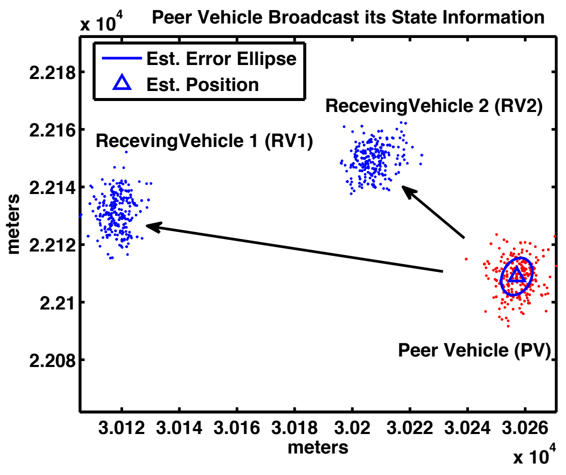
Fitted with an acoustic modem, the vehicles are able to communicate and share information with other vehicles within their communication range. The vehicles in the team are assumed to be capable of measuring the time-of-flight of acoustic signals and therefore can estimate their range from broadcasting vehicle using either the OWTT or the TWTT of the acoustic signal, with an assumed constant sound speed profile. A simple round-robin scheduling is adopted such that each vehicle, termed as Peer Vehicle (PV), in the team broadcasts its local state information, sequentially using acoustic communication. Round-robin scheduling for ranging among the vehicles has the advantage of eliminating the probability of collision, thus increasing the throughput of the network. Due to the extremely limited communication bandwidth, the broadcast information includes only the vehicle's current position point estimate, $\hat{\pi}_t^{\mathrm{PV}}$ and its filter's estimated covariance matrix, $\mathbf{P}_t^{\mathrm{PV}}$, as estimated in equations (4.10) and (4.11), even though its local particle filter may be tracking a multi-modal particle distribution. Sharing multiple modes of the distribution, if they exist, may improve the performance of the proposed decentralized filter. However, this will increase the requirement of the communication bandwidth and potentially decrease the robustness of the communication link.

When the acoustic signal is received by other vehicles, termed as Receiving Vehicle (RV), the range, $\hat{R}_t$, between the two vehicles can be estimated. Since the range is part of the measurements, we model its measurement error as a zero-mean Gaussian random variable with variance σ_R^2 :

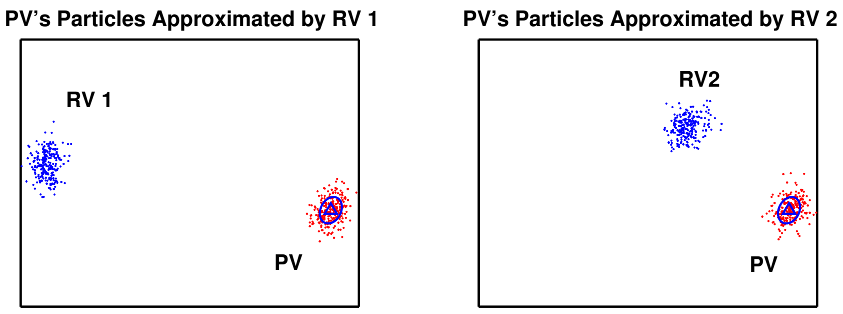
$$\hat{R}_t \sim \mathcal{N}(\mid \pi_t^{\mathrm{PV}} - \pi_t^{\mathrm{RV}} \mid, \sigma_R^2) \tag{4.16}$$

where π_t^{PV} and π_t^{RV} are PV and RV's ground truth positions, respectively.

The information received cannot be used directly to influence the measurement model, as presented in [40]. This is because none of the vehicles in the team is equipped with high accuracy navigational sensors, and the PV may have accumulated significant error by the time the information is broadcast. Instead, PV's information is used to



(a)



(b) (c)

FIGURE 4.4: (a) Local sufficient statistic information (both the estimated position and error covariance) is broadcast by the PV during Acomms. (b) N particles (Red dots) approximated by RV1 using the received information. (c) N particles (red dots) approximated by RV2 using the received information.

influence RV's particle distribution, and affects the corresponding likelihood computation. In the following sections, we present two different approaches for incorporating the PV's information in the RV's measurement model.

4.3.2.1 Approximation of the Peer Vehicle's Particles

The first approach approximates PV's particle set and utilizes the inter-vehicle ranging information to influence the particles' likelihood computation. Given $\hat{R}_t$, $\mathbf{P}_t^{\mathrm{PV}}$ and $\hat{\pi}_t^{\mathrm{PV}}$, we assume that the probability of RV's particle representing the vehicle's true position is directly proportional to the probability of the particle located at $\hat{R}_t$ meters away from the $\hat{\pi}_t^{\mathrm{PV}}$, taking into account the $\hat{\pi}_t^{\mathrm{PV}}$'s uncertainty covariance, $\mathbf{P}_t^{\mathrm{PV}}$. The likelihood evaluation for each of the RV particles ($\pi_t^{\mathrm{pf},i}, i \in 1 \ldots N$) is as follows:

1. Whenever the PV's information is received via acoustic ranging, a set of N particles ($\varpi_j^{\mathrm{PV}}, j = 1 \ldots N$) are normally distributed around $\hat{\pi}_t^{\mathrm{PV}}$ with covariance $\mathbf{P}_t^{\mathrm{PV}}$.

2. The Euclidean distances from $\pi_t^{\mathrm{pf},i}$ to all the particles generated in step 1 are computed, resulting in N distance measurements.

3. The likelihood of $\pi_t^{\mathrm{pf},i}$ is evaluated by taking the sum of the differences between the N measurements against the estimated range, $\hat{R}_t$. The smaller the differences, the higher the likelihood of $\pi_t^{\mathrm{pf},i}$:

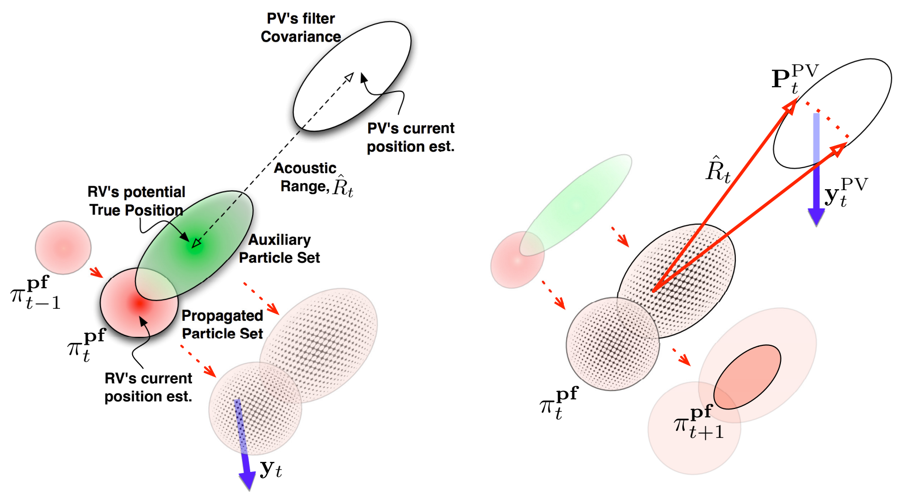
$$p(\pi_t^{\mathrm{pf},i}, \hat{R}_t, \mathbf{P}_t^{\mathrm{PV}}, \hat{\pi}_t^{\mathrm{PV}}) \propto 1/(\sum_{j=1}^{N} |\|\pi_t^{\mathrm{pf},i} - \varpi_j^{\mathrm{PV}}\| - \hat{R}_t|) \tag{4.17}$$

The assumption makes use of the PV's estimated state information as well as ranging information to further influence the RV particle's distribution. An example of state information approximation and sharing among the vehicles is illustrated in Fig. 4.4. As a result, the RV particles' likelihood evaluation consists of an extra likelihood function, fusing the information received via acoustic communication:

$$p(\mathbf{y}_t \mid \pi_t^i) = p(\mathbf{y}_{t:t-\ell} - h(\pi_{t:t-\ell}^{\mathrm{pf},i})) \times p(\pi_t^{\mathrm{pf},i}, \hat{R}_t, \mathbf{P}_t^{\mathrm{PV}}, \hat{\pi}_t^{\mathrm{PV}}) \tag{4.18}$$

4.3.2.2 Introduction of Auxiliary Particles for the Receiving Vehicle

The second approach approximates the RV's particle set based the PV's state information, and uses the PV's water depth measurement, in addition to the inter-vehicle ranging information, to influence the likelihood computation. Let $\mathbf{y}_t^{\mathrm{PV}}$ be the PV's latest water depth measurement. At every communication period, the $\mathbf{y}_t^{\mathrm{PV}}$ is first fused into the PV's local filter, before being broadcast together with the resulted $\mathbf{P}_t^{\mathrm{PV}}$ and $\hat{\pi}_t^{\mathrm{PV}}$ to all the RV. Once received, the likelihood of RV's particle set is computed in two separate stages:


PV's filter Covariance
PV's current position est.
Acoustic Range, $\hat{R}_t$
RV's potential True Position
Auxiliary Particle Set
$\pi^{\mathbf{pf}}_{t-1}$
$\pi^{\mathbf{pf}}_{t}$
Propagated Particle Set
RV's current position est.
$\mathbf{y}_t$
$\mathbf{P}^{\mathrm{PV}}_t$
$\hat{R}_t$
$\mathbf{y}^{\mathrm{PV}}_t$
$\pi^{\mathbf{pf}}_{t}$
$\pi^{\mathbf{pf}}_{t+1}$

(a) (b)

FIGURE 4.5: (a) Illustration shows the PV broadcast its current position estimate and error covariance via acoustic communication. Upon receiving it, the RV determines the distance (acoustic range) from the PV, and uses the PV's information to introduce new particle set (green ellipse) into its own particle set (red circle). (b) Illustration shows information from PV is used by the RV's particles for the second stage likelihood evaluation. N particles are resampled with replacement from the pool of $N+M$ particles according to their relative normalized weights.

1. Introduction of Auxiliary Particle Set

A set of M auxiliary particles is added to the RV's original particle pool. These particles are normally distributed around a position that is at a distance $\hat{R}_t$ away from $\hat{\pi}^{\mathrm{PV}}_t$ and located along a straight line between PV and RV, with the covariance of the distribution being $\mathbf{P}^{\mathrm{PV}}_t$ (Fig. 4.5(a)). The resultant $N+M$ particles then are weighted using the same likelihood function (4.15) as other particles. Intuitively, the introduction of the auxiliary particles modifies the distribution through the inter-vehicle constraints from ranging. The new distribution has the potential to alleviate divergence when the vehicle navigates over a flat terrain, until it enters another area that has more terrain variability.

2. Utilizing PV's Water Depth Measurement

Given $\hat{R}_t$, $\mathbf{y}^{\mathrm{PV}}_t$ and $\mathbf{P}^{\mathrm{PV}}_t$, we assume that the probability of an RV particle representing the vehicle's true position is directly proportional to the probability of measuring

$\mathbf{y}_t^{\text{PV}}$ within the ellipse described by the $\mathbf{P}_t^{\text{PV}}$, and at a distance of $\hat{R}_t$ away from the particle's current position. For each of the $N+M$ particles, ϖ_i^{RV} where $i = 1 \ldots N+M$, resulting from the first stage, a new set of particles, ϖ^{int}, is randomly generated along the arc formed by the intersection of a circle having radius $\hat{R}_t$ and centered at ϖ_i^{RV}, with $\mathbf{P}_t^{\text{PV}}$. The average likelihood of ϖ^{int} evaluated against $\mathbf{y}_t^{\text{PV}}$ contributes to the likelihood of ϖ_i^{RV}. This assumption makes use of PV's water depth measurement as well as the derived ranging information to further influence the local particles' distribution. This second stage likelihood evaluation is further illustrated in Fig. 4.5(b).

As a result, the particle's likelihood evaluation is similar to that of (4.18), but with an extra term incorporating the water depth measurement:

$$p(\mathbf{y}_t \mid \pi_t^i) = p(\mathbf{y}_{t:t-\ell} - h(\pi_{t:t-\ell}^{\text{pf},i})) \times p(\pi_t^{\text{pf},i}, \hat{R}_t, \mathbf{P}_t^{\text{PV}}, \hat{\pi}_t^{\text{PV}}, \mathbf{y}_t^{\text{PV}}) \tag{4.19}$$

Once all the particles undergo the likelihood evaluation, the original N particles are resampled with replacement, from the pool of $N+M$ particles, according to their relative normalized weights.

4.4 Simualtions and Results

A series of simulation studies were carried out to assess the feasibility of the proposed decentralized MPF, and to evaluate its performance by varying different parameters used in the filter. The parameters shown in Table 4.1 were kept the same throughout the simulation runs, except for studies that involved varying the specified parameters. The process and measurement noises were assumed independent and drawn randomly at every propagation and measurement steps, from Gaussian distribution characterized by the noise matrices. We assumed that all the vehicles have a GPS fix before submerging. Thus, each of their local filters were initialized to cover a search area of 20×20 m^2 centering at the individual fixes.

The bathymetry map was obtained from the water near the St. John Island, Singapore in year 2012 using a *Reson* 8125 multi-beam echo-sounder. The equipment was

Algorithm 4 Marginalized Particle Filtering for cooperative localization

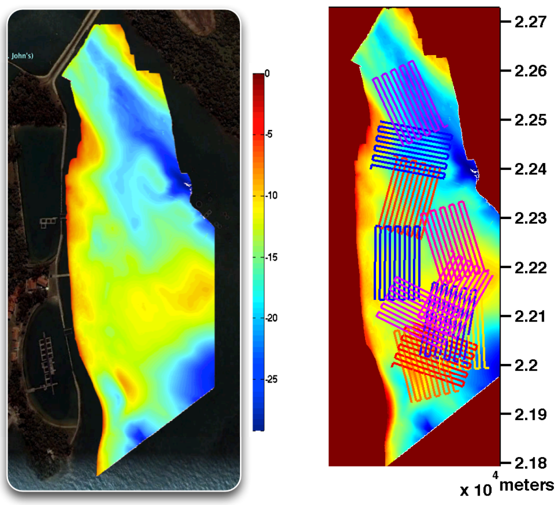
```
1. Initialization:
$t \leftarrow 0$ (time steps);
let $V_m$ be the number of vehicles in the team.
Draw $N$ particles: $\pi_0^{i,j} \sim p_{x_0}$; $i = 1:N$ and $j = 1:V_m$.
$\mathbf{P}_{0|-1}^{\text{kf},j} \leftarrow \mathbf{P}_0^{\text{kf}}$; $j = 1:V_m$.
$w_0^{i,j} \leftarrow \frac{1}{N}$; $i = 1:N$ and $j = 1:V_m$.

2. Time update:
for $j = 1 \rightarrow V_m$ do
  for $i = 1 \rightarrow N$ do
    Compute $\pi_{t+1}^{\text{pf},i,j}$ according to (4.5) and $\pi_{t+1}^{\text{kf},i,j}$ according to (4.7)
  end for
end for

3. Measurement update:
for $j = 1 \rightarrow V_m$ do
  if PV's information received then
    Evaluate $p(\mathbf{y}_t \mid \pi_t^{\text{pf},i,j})$ according to section 4.3.2.1 or section 4.3.2.2
  else
    Evaluate $p(\mathbf{y}_t \mid \pi_t^{\text{pf},i,j})$ according to (4.15)
  end if
  Compute the weights: $w_t^{i,j} \leftarrow w_{t-1}^{i,j} p(\mathbf{y}_t \mid \pi_t^{\text{pf},i,j})$; $i = 1:N$
  Normalize the weights : $w_t^{i,j} = \frac{w_t^{i,j}}{\sum_{i=1}^N w_t^{i,j}}$
end for

4. State estimation:
for $j = 1 \rightarrow V_m$ do
  $\hat{\pi}_t^{\text{MMS},j} \leftarrow \sum_{i=1}^N w_t^{i,j} \pi_t^{i,j}$
end for

5. Resampling :
for $j = 1 \rightarrow V_m$ do
  Let $N^{\text{eff}} = \frac{1}{\sum_{i=1}^N (w_t^{i,j})^2}$
  if $N^{\text{eff}} \leq N^{\text{th}}$ then
    $\pi_t^{i,j} = \text{Resample}(\{w_t^j\}, \{\pi_t^j\})$; $i = 1:N$
    $w_t^{i,j} \leftarrow \frac{1}{N}$; $i = 1:N$
  end if
end for

6. Iterate :
$t \leftarrow t + 1$
Goto Step 2.
```

TABLE 4.1: SIMULATION PARAMETERS

Parameter	Value
No. of vehicles	3
No. of particles	600
Filter sampling time	1 s
Vehicles velocity	1.5 m/s
Ranging Period per vehicle	9 s
Ranging scheduling	Round Robin
Process noise std. div., (σ_ζ)	$\begin{bmatrix} 0.05 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$ m
Altimeter measurement noise std. div., (σ_ξ)	0.05 m
Ranging measurement noise std. div., (σ_R)	1 m

operated with 240 beams at 455 kHz with a combined swathe width of 120 deg. The vertical resolution of the data is 0.01 m while the horizontal resolution is down-sampled to 1 m grid cells. The water depth is from a few meters to around 30 m depth.

The feasibility of bathymetry-based localization depends on the amount of information contained within a bathymetry map. Besides varying different parameters for the filter in the simulation runs, we also investigate, in general, if a given bathymetry map contain sufficient amount of information for multi-vehicle localization. Thus, 300 different lawn-mowing paths (100 paths for each of the vehicles) were randomly generated within the map. This allows us to conduct 100 different simulated runs (with one trial per path set) using those paths. The results of the simulations are shown in the form of position estimation errors of each vehicle at the end of all the simulated runs. If a high percentage of the simulated runs achieve good localization performance (low position estimation errors), we conclude that the map indeed has sufficient terrain information. Using the results from the simulation, we perform analysis on the best and worst performing cases to further investigate the influence various parameter settings have on the performance of the filter. Fig. 4.6 shows the bathymetry map used in the simulation studies and examples of randomly generated lawn-mowing paths for the simulation studies.



(a) (b)

FIGURE 4.6: (a) Bathymetry map of St. John Island, Singapore obtained in year 2012. Terrain variation ranging from a few meters to 30 m depth. (b) Examples of randomly generated paths within the bathymetry map.

4.4.1 Measurement Models

The first simulation study was conducted to compare the performance of the measurement models presented in section 4.3.2. We term the model in section 4.3.2.1 as the Model with Particle Approximation (MPA), and the model in section 4.3.2.2 as the Model with Auxiliary Particles (MAP). Whenever inter-vehicle communication happened, a total of 300 auxiliary particles ($M = 300$) were generated for stage 1 of MAP, resulting in 900 particles for the evaluation against the water depth measurement in stage 2.

The boxplots in Fig. 4.7 show that the MPA perform significantly better than the MAP in estimating the vehicles' positions. Even though MAP manage to perform well in some cases as illustrated by the low position error of its boxplot's smallest value, as well as by the results reported in [70], its performance is less consistent than the MPA in handling different terrain profiles and path configurations. Nevertheless, MAP still outperformed the dead-reckoning in most cases.

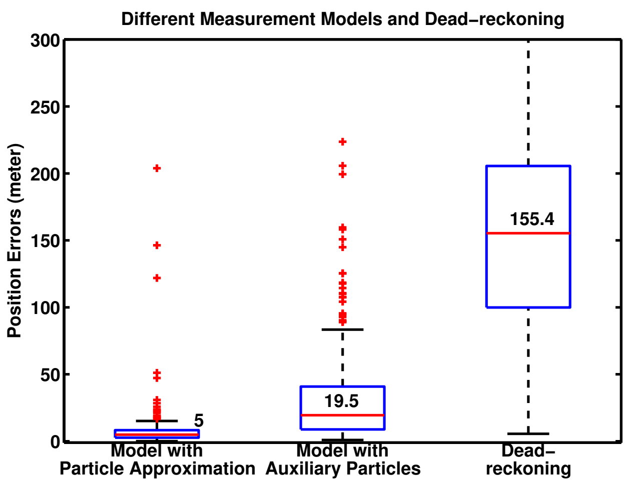


FIGURE 4.7: Distribution of position estimation errors for the decentralized MPF using different measurement models. Boxplots show median (numeric) and 25% - 75% quartiles while the whiskers are the smallest and greatest values, and the red crosses are the outliers.

Besides, the MPA is also more computationally efficient than MAP. During likelihood computation, the MAP has M extra particles and $(N+M)\times\varpi^{\text{int}}$ bathymetry map look-ups for evaluations against $\mathbf{y}_t^{\text{PV}}$. Due to the advantages, we chose the MPA for the simulation studies in the following sections.

4.4.2 Influence of Communication Bandwidth

Due to the limited underwater communication bandwidth, it is impractical to share all the PV's particle information with the RV during underwater cooperative localization. Nevertheless, the simulation can be used as a good benchmark to compare the performance of the decentralized filters, when it is performed offline. In this section, we undertake a simulation study whereby all the vehicles have unlimited communication bandwidth during the filter information broadcast step. As in the decentralized version, each vehicle still runs a local filter. However, instead of approximating the PV's particle distribution, it is assumed that all the vehicles have access to all the other vehicles' filter information. This can be seen as the unconstrained version of the filter with MPA presented in the previous section, where the vehicles have access to other vehicles' particle sets.

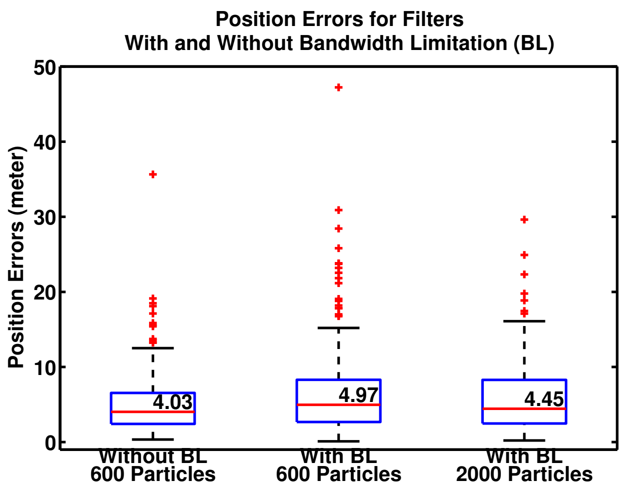


FIGURE 4.8: Distribution of position estimation errors for decentralized filters with and without communcation bandwidth limitation.

During the filter's measurement update, instead of approximating the PV's particle distribution as shown in Fig. 4.4, and using them for the computation of the likelihood of the RV's particles, the exact locations of PV's particles are used. This approach makes sure the filters use same process and measurement models, and provides a fair comparison that the only factor which affects the performance of the filter is the amount of information being exchanged among the vehicles in the team.

The first boxplot on the left of Fig. 4.8 shows the distribution of position estimation errors where the filters information was broadcast assuming unlimited communication bandwidth. By allowing full access to other filter's information during the measurement update step, the decentralized filters achieve the best performance. More importantly, the performance achievable by the filter with MPA is comparable (middle boxplot in Fig. 4.8), even though the filter's information sharing is based on distribution aggregation. The results demonstrate the feasibility of the decentralized filter to be used for underwater multi-vehicle cooperative localization, where only the sufficient statistic of the particle distribution need to be shared via the bandwidth limited acoustic communication .

The ability of a particle filter in estimating a vehicle's position depends on how accurate the position's probability distribution is represented by a set of particles. Apart

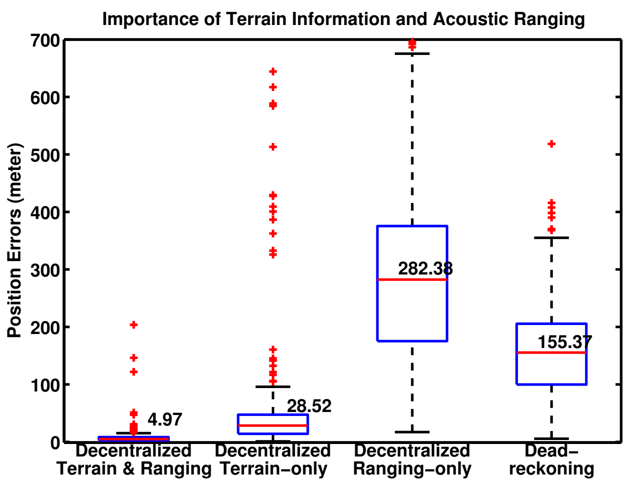


FIGURE 4.9: Distribution of position estimation errors of various decentralized MPFs against dead-reckoning. The results show the importance of having both the terrain and ranging information in the filter's performance.

from the issue of *sample impoverishment* mentioned in Section 4.2.2.1, the number of particles used also plays an important role. Higher number of particles may increase the accuracy, but also incurs higher computational cost, while insufficient number of particles may result in the true density not being encompassed by the sample set. We repeated the simulations and increased the number of particles in the filter from 600 to 2000, while keeping the other parameters mentioned in Table 4.1. The result showed only a slight improvement (right-most boxplot in Fig. 4.8) and that a set of 600 particles appeared to be sufficient for representing the distribution, and is used for subsequent studies.

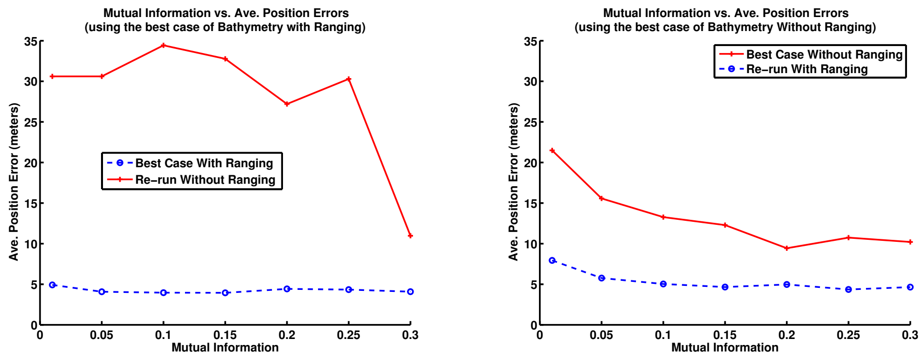
4.4.3 Importance of Acoustic Communication and Bathymetry Information

As mentioned in the Section 4.3.2, multi-vehicle cooperative localization is achieved by incorporating both the vehicle's water depth and inter-vehicle range measurements into the decentralized MPF's measurement model, to estimate the vehicle's position. In this section, we perform simulation studies to investigate the importance of having both pieces the information (terrain & ranging) on the filter's performance, as against having only a single piece of information: either using the terrain information (terrain-only) or the ranging information (ranging-only). The results were compared with the position errors accumulated by the dead-reckoning method, to illustrate the potential benefit, if any, of having terrain and/or ranging information (Fig. 4.9).

The position estimation errors of the decentralized MPF with both the terrain and ranging information were the same as the previous case shown in Fig. 4.8. However, the estimation errors increase significantly in the absence of acoustic communications among the vehicles, as shown by the terrain-only boxplot in Fig. 4.9. Since the filter's performance in this case depends solely on the terrain information within the area where the paths were generated, the wider spread of position errors showed there was a good mixture of areas, each containing different amount of terrain information, that was randomly selected within the bathymetry map for planning the vehicles' paths. The mixture provided suitable scenario to illustrate the benefits of incorporating acoustic communication, as can be seen in terrain & ranging boxplot. Nevertheless, the resulting terrain-only filter still outperformed the dead-reckoning method in most cases, except some outlier cases where the filter diverged due to insufficient terrain information. On the other hand, cooperative localization with ranging-only performed poorly and in most cases, worse than the dead-reckoning method. This is due to overconfidence of the filter's estimations mentioned in the previous sections, and made worse especially when none of the vehicles have a geo-referenced position information. The results will be further explained in the following sections.

4.4.3.1 Influence of Inter-vehicle Acoustic Communication

In this section, we turn to the concept of information gain to illustrate the benefits of having acoustic communication in cooperative bathymetric-based localization. As described in [76], the mutual information $I(X;Y)$ is the reduction in the uncertainty of the random variable, X, due to the knowledge of random variable, Y. In the case of cooperative localization, this translates to the reduction of uncertainty in the vehicle's position estimate, due to the information gained from the terrain and/or ranging information. This measure can be treated as an indicator for the effectiveness of the decentralized



(a) (b)

FIGURE 4.10: Mutual information against the average position estimation erros of the best performing cases. The decentralized MPF is more effective whenever both the ranging and bathymetry information are incorporated in the filter's measurements. (a) Comparison using the best performing case of terrain & ranging (circle-dashed line). The simulation was re-ran without ranging information (plus-solid line). (b) Comparison using the best performing case of terrain-only (plus-solid line). The simulation was re-ran with the addition of inter-vehicle ranging information (circle-dashed line).

MPF in estimating the vehicle's position: if the filter is effective and converges to the correct estimate, large information gain should yield lower position error.

Mutual information is defined as:

$$I(X;Y) = H(p(X)) - H(p(X) \mid p(Y)) \tag{4.20}$$

where $H(X)$ is the entropy of the *prior* distribution and $H(X \mid Y)$ is the *posterior* distribution. We adopt the approach in [77] to approximate the entropy of a probability distribution, p, using:

$$H(p) \approx \sum_i w^i \ln \sum_j w^j \mathrm{K}(\pi^{\mathrm{pf},i} \mid \pi^{\mathrm{pf},j}) \tag{4.21}$$

where $\mathrm{K}(\pi^{\mathrm{pf},i} \mid \pi^{\mathrm{pf},j})$ is a Gaussian radial kernel approximated by:

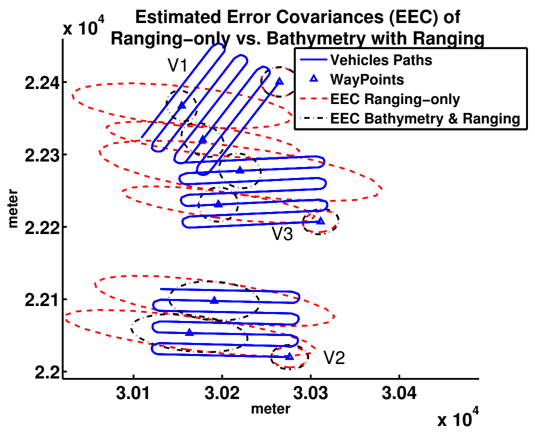
$$\mathrm{K}(\pi^{\mathrm{pf},i} \mid \pi^{\mathrm{pf},j}) \approx e^{-\frac{1}{2}\|\pi^{\mathrm{pf},i} - \pi^{\mathrm{pf},j}\|^2/\sigma^2}. \tag{4.22}$$

For comparison, the best performing case of the terrain & ranging, and the terrain-only simulation results shown in Fig. 4.9 were used to investigate the benefits of having inter-vehicle ranging during cooperative localization. The results from the first study is shown in Fig. 4.10(a) where the simulation using terrain & ranging (circle-dashed line) was re-run without the ranging information (plus-solid line). The results from the second study are shown in Fig. 4.10(b) where the simulation using terrain-only (plus-solid line) was re-run with the addition of inter-vehicle ranging information (circle-dashed line). The large differences in the average position errors, across all levels of mutual information, clearly show that the decentralized MPF is more effective when the ranging information is incorporated in the measurement model.

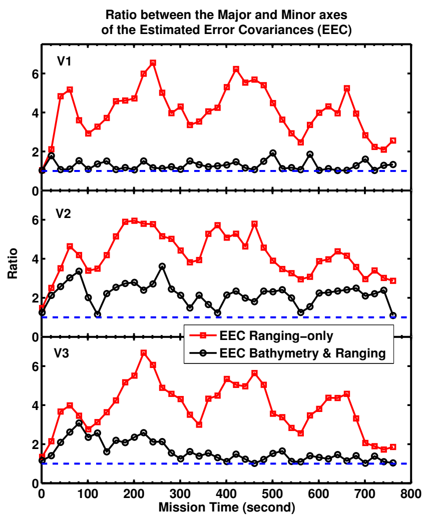
4.4.3.2 Influence of Bathymetry Information

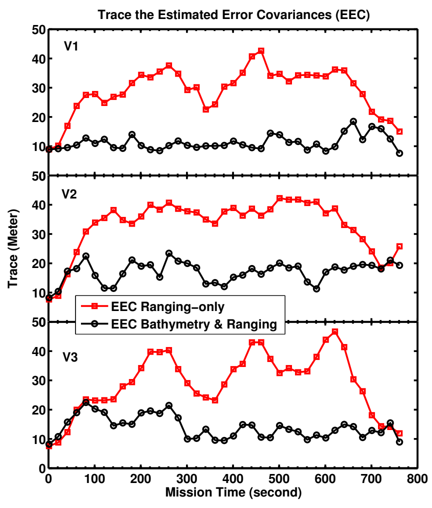
Without geo-referenced position information such as the beacon vehicle mentioned in [56, 78], the vehicles only rely on relative range measurements to estimate their positions. Over time, this causes the uncertainty of the vehicles' position to increase, due to the accumulation of the process and measurement noises. Furthermore, as illustrated in section 3.1, acoustic signal broadcast by a PV, at position π^{PV}, only contains ranging information in the radial direction of the ranging circle centered at π^{PV}. Consequently, if the ranging is measured consecutively from about the same *relative aspect* (see section 3.1), with respect to the PV, the position uncertainty of the RV in the tangential direction continues to grow. In this section, we analyze one of the cases from the ranging-only simulation results shown in Fig. 4.9. The simulation is repeated with the addition of bathymetry information in the filter's measurement model. The vehicles' paths used for the simulation, as well as the Estimated Error Covariance (ECC) at different waypoints are shown in Fig. 4.11(a).

Fig. 4.11(b) shows the aspect ratio of the error ellipsoid described by the EEC for all the three vehicles (V1,V2 and V3), while Fig. 4.11(c) shows the trace (sum of diagonal elements) of the ECC matrix throughout the mission time. The aspect ratio denotes the skewness of the error uncertainty, while the trace denotes the magnitude of the error uncertainty. An effective filter typically has an aspect ratio as close to 1 as



(a)





(b) (c)

FIGURE 4.11: (a) The Estimated Error Covariance (EEC) at different waypoints (blue triangles) along the vehicles' paths. (b) The ratio between the major and minor axes of the EEC throughout the mission time. (c) The trace of the EEC throughout the mission time. By incorporating both the bathymetry and ranging information in the measurement model, the vehicle's individual filter was able to achieve lower trace and keep the aspect ratio closer to 1 throughout mission time.

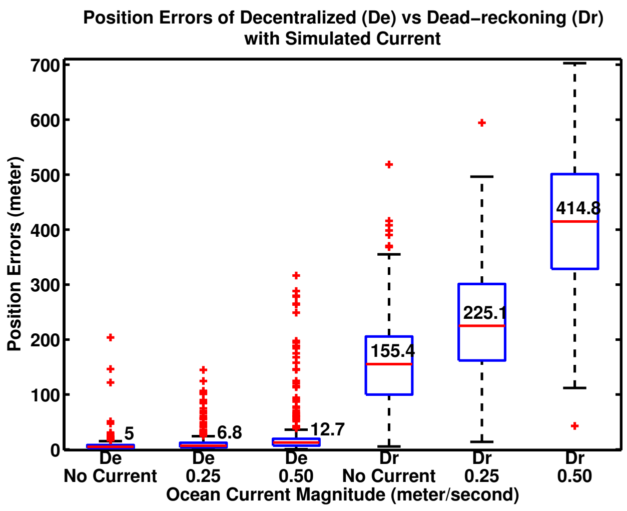


FIGURE 4.12: Position estimation errors of the decentralized (De) MPF and dead-reckoning (Dr) under the influence of simulated ocean currents with different magnitude.

possible, and keeps the trace minimum. The results in Fig. 4.11(b) and 4.11(c) clearly show that by incorporating bathymetry and ranging information in the measurement model, the filter is able to achieve lower trace and keep the aspect ratio closer to 1 throughout mission time.

The position estimation errors in some ranging-only cases can be worse than the dead-reckoning method (see Fig. 4.9). This may due to filter divergence in one of the vehicles, exacerbated by the wrong estimate feedbacks within the vehicle network, causing error reinforcement. However, if the cooperative localization is aided by bathymetry information, this issue can be potentially avoided, as the individual vehicles' positions and error covariances are estimated solely from their own bathymetry measurements between acoustic communications.

4.4.4 Influence of Simulated Ocean Current

Without an exteroceptive sensor, like the DVL, to measure the vehicle's ground speed, it is crucial for the filter to track the existence of an ocean current for more accurate propagation of the process model. In this section, we repeat the simulations with a southward simulated ocean current to investigate its influence on the performance of the

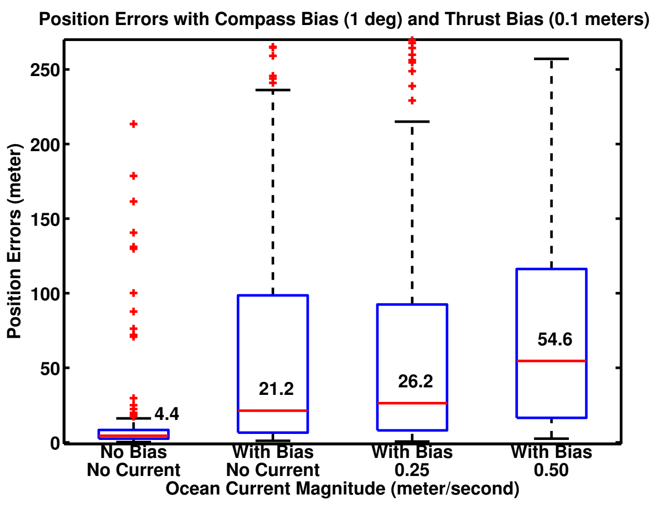


FIGURE 4.13: Position estimation errors of the decentralized MPFs under the influence of compass bias of 1 deg and thrust bias of 0.1 m/s. The filter shows some level of robustness against the biases and simulated ocean current.

filter. We also compare the results with the dead-reckoning method to further illustrate the benefits of employing cooperative localization. The parameters shown in Table. 4.1 remain the same.

Fig. 4.12 shows the results from the simulations with different magnitudes of simulated ocean current. The decentralized MPF is able to retain its performance under the influence of a southward ocean current with a magnitude of 0.25 m/s, and degrades slightly when the magnitude was increased to 0.50 m/s. However, all three filters' performance still outperform the results produced by the dead-reckoning method, with or without the ocean current.

4.4.5 Influence of Compass and Thruster Biases

Often a vehicle's sensors or actuators produce readings with a constant offset (also known as bias), if left uncalibrated. Although the biases can be modeled as part of the system state and tracked by the decentralized MPF, it increases the dimensionality of the search space, thus requiring a higher number of particles for the filter to converge. Since we do not model any bias in the system state, it is worthwhile to investigate the robustness of the filter when there are indeed biases that exist in either the sensors or actuators of the vehicle.

In this simulation study, we repeated the simulation runs with a compass bias of 0.1 deg, and a thruster model bias of 0.1 m/s. The results of the simulation runs are shown in Fig. 4.13. As can be seen, the position estimation errors increased significantly due to the existence of the biases. Since the biases were not taken into account in the process model, they were treated as the ocean current by the filters and misled the propagation process. However, in areas with sufficient bathymetry information, the decentralized MPF managed to keep the position estimation errors low, as shown by the lower whisker and smallest value of the boxplots. The simulation results showed that given sufficient amount of bathymetry information in a mission area, the decentralized MPF is robust against some levels of compass and thruster biases, even under the influence of simulated ocean currents.

4.4.6 Influence of Bathymetry Map Resolution

The bathymetry maps used for underwater localization missions may come in different resolutions. The performance of the bathymetry-based localization not only depends on the sampling efficiency of vehicle's sensors, but also depends on the resolution of the bathymetry map used. A low resolution bathymetry map resembles a topography that contains little excitation where the depth variation within a large grid cell is assumed constant. Regardless of the location of the particles within a grid cell, the best depth information that a particle can deduce from the bathymetry map is from the closest vertex of the containing grid cell. Even though interpolation techniques can be used to improve the result, their accuracies are still subjected to the assumption made within the interpolant, and may not depict the true depth reading.

To investigate the impact of the bathymetry map resolution against the performance of the proposed filter, we repeated the simulation runs using bathymetry maps with different map resolutions. For the purpose of this study, the original bathymetry map with 1 meter resolution is downsampled to 5 meters and 10 meters respectively. During the depth measurement step, the depth information of the closest grid vertex,

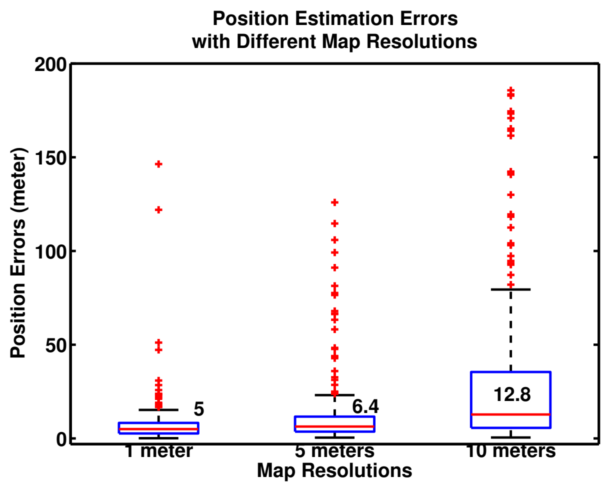


FIGURE 4.14: Position estimation errors of the decentralized MPFs when bathymetry maps with different resolutions were used. The performance of the filters decreased as the resolution of the bathymatry maps decreased.

with respect to the particles' latest propagated locations, were used as their depth measurements.

The middle and right-most boxplots in the Fig. 4.14 show the distributions of position errors of the decentralized MPFs when the bathymetry map of 5 meters and 10 meters resolution were used. Compared with the result where bathymetry map of 1 meter resolution was used (left-most boxplot), it can be seen that the resolution of the bathymetry map used has an impact on the decentralized MPF's performance. As the resolution of the map decreases, so does its information content. Thus, the position errors of the filters also increase.

4.5 Field Experiments

In this section, we present the results obtained from field experiments using bathymetric maps from two areas with distinct terrains. The first map is from waters Charles River Basin, Boston (Fig. 4.15) where the terrain is flat and patchy in places. The second map is from the water near the St. John Island, Singapore, as shown in Fig. 4.6(a). In the following experiments, we collected necessary data using vehicles fitted with single-beam echo-sounder, and ran the decentralized MPF in post processing to evaluate

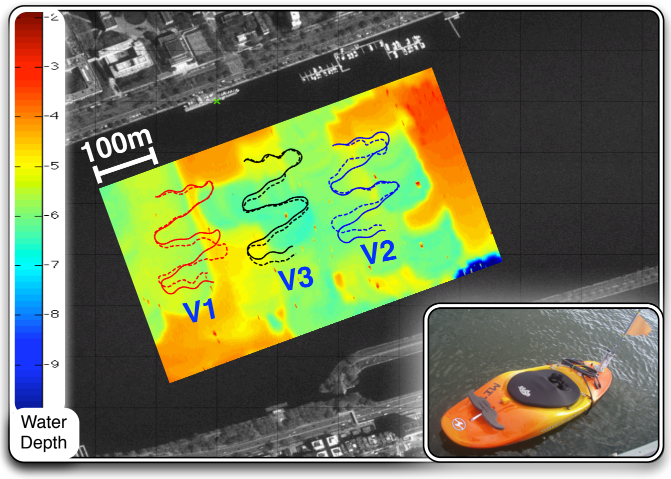


FIGURE 4.15: Bathymetry map of Charles River, paths executed (solid-line) by the autonomous surface vehicle (insert) and the trajectories tracked (dotted-line) by the decentralized MPF. The vehicle was fitted with a single-beam altimeter.

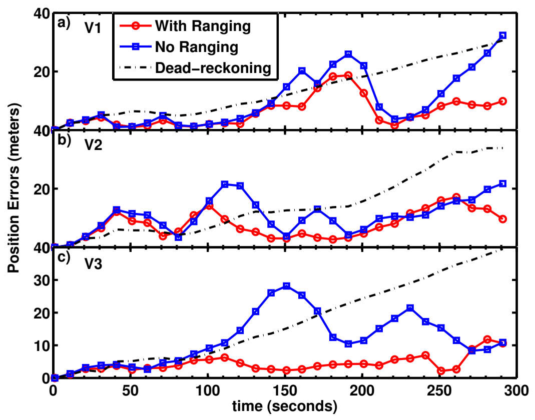
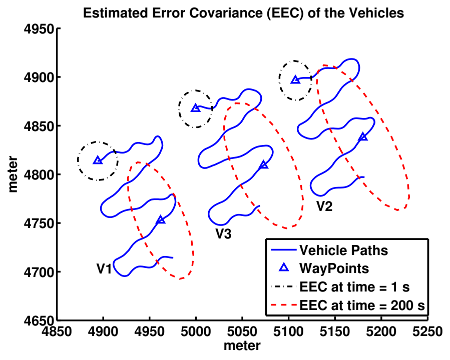
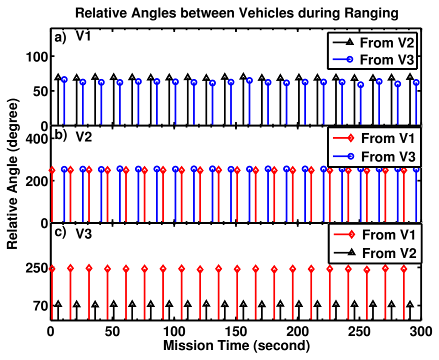


FIGURE 4.16: The average position estimation errors for all three vehicles (V1...V3) over 10 localization runs using the same paths. The position errors of the vehicles are lower when both the ranging and bathymetry information are incooperated in the decentralized MPF for cooperative localization.

the localization performance. The MPA (Section 4.4.1) measurement model and the parameters shown in Table 4.1 were used throughout all the runs. The process and measurement noises are assumed Gaussian independent and drawn randomly at every propagation and measurement step.



(a)



(b)

FIGURE 4.17: The inter-vehicle ranging at about the same *relative aspects* cause the estimated error covariance (EEC) of the vehicles to grow at tangential direction with respect to the direction of ranging from mission time $t = 1$ to $t = 200$ seconds. (a) The EEC for each of the vehicles (V1... V3) estimated at different waypoints (triangles) along the vehicle paths. (b) Relative angles between the vehicles (global frame) during inter-vehicle rangings.

4.5.1 Charles River Basin, Boston

The first set of tests was performed using the bathymetry map of the Charles River Basin, with altimeter measurements obtained from an autonomous surface vehicle (ASV). The ASV is fitted with a *Tritech*-PA500 single-beam altimeter, providing one-millimeter resolution when it operates in digital mode. A total of three lawnmowing-like paths were planned within the area where the bathymetry information is available. The ASV was commanded to follow these three paths using high-precision RTK GPS as a ground truth, while collecting depth data. The resultant paths are shown in Fig. 4.15. The oscillating patterns on the trajectories were due to the surface waves and the ASV's onboard control system, which were not modeled in the filter.

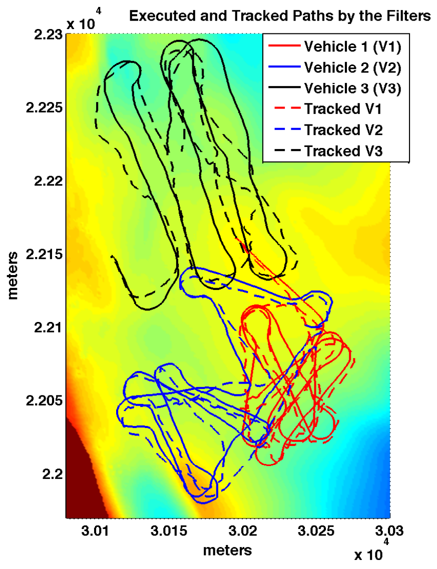
With control input to the filter's process model derived from the planned paths, we carried out cooperative localization using the depth data as if it has been obtained by three separate vehicles. Acoustic communication was simulated between the vehicles with a ranging period of 15 seconds per vehicle. The range between the vehicles was computed from the vehicle's GPS ground truth during acoustic communication, corrupted with measurement noise in (4.16). Fig. 4.15 shows the trajectories tracked (dotted-lines) by the decentralized MPF of each vehicle, while Fig. 4.16 shows the corresponding position errors accumulated by the filters. Throughout the mission execution, the decentralized MPF maintained the position errors within 20 m (under 10 m for most of the time). In comparison, the position errors are higher when there is no filter and ranging information sharing among the vehicles, and even higher when the vehicles depend solely on dead-reckoning for navigation.

Due to the layout of the vehicles' path, individual vehicle receives the ranging broadcasts from other vehicles at about the same *relative aspects* (or 180 deg successively from each of the vehicles in the case of V3) throughout the mission, as shown in Fig. 4.17(b). Such inter-vehicle ranging does not contain position information in the tangential direction (with respect to the direction of ranging) and causes the estimated error covariances (EEC) to grow unbounded in that direction. This can be clearly seen in Fig. 4.17(a) where the EEC of individual vehicle evolved from circular shape at beginning of the mission, $t = 1$ second, to elongated ellipses with their major axes almost parallel to each other, at time $t = 200$ seconds. such cases of inter-vehicle ranging, coupled with the patchy terrain (lack of terrain information) around the mission area, resulted in poor localization performance especially around 200 seconds mark into the mission for V1 and 260 seconds mark for V2 (see Fig. 4.16).

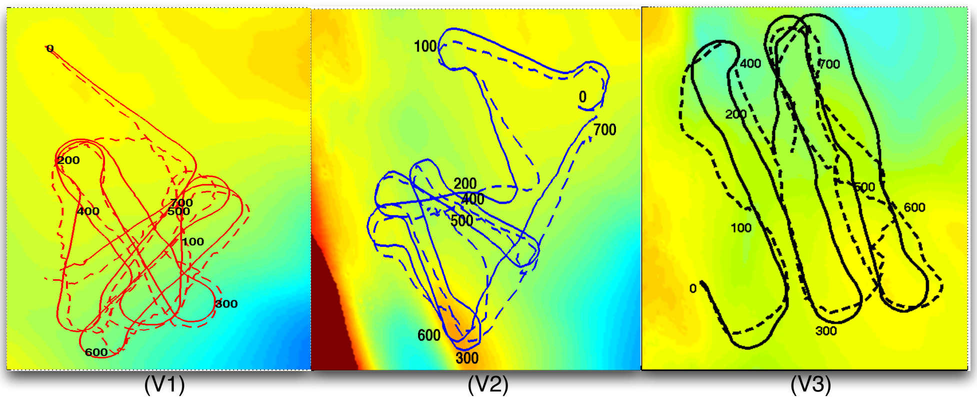
4.5.2 St. John Island, Singapore

In this section, we performed offline cooperative localization with field data collected by the STARFISH AUV [3] (Fig. 1.1(c)). The AUV is equipped with a *Tritec Micron* single-beam altimeter, providing five-millimeter accuracy, when operated in digital mode. All the tests were conducted as surface missions so that the GPS logs can be used as ground truth.

A total of three separate missions conducted during sea trials in year 2013 were used for the offline validation (see Fig. 4.18(a), and Fig. 4.18(b) for the individual paths). The vehicles were either performing lawn-mowing mission (V3) or cooperative positioning algorithms (V1 and V2) mentioned in [56]. For the offline cooperative localization, we make use of the altitude measurements collected during those missions to assess the decentralized MPF's performance. Acoustic communications were simulated



(a)



(b)

FIGURE 4.18: (a) Trajectories of the AUVs during the field trials. (b) Individual AUV's trajectory with mission time steps marked at 100 seconds span. Dotted lines are the trajectories traked by the decentralized MPFs.

so that filter information can be shared among the vehicles. Each vehicle was scheduled to broadcast its filter information at a period of 9 seconds. Similarly, the range between the vehicles was computed from the vehicle's GPS ground truth and corrupted with the measurement noise mentioned in (4.16).

Fig. 4.18 shows the trajectories executed by the vehicles (solid lines) during the sea trials, together with the corresponding trajectories tracked (dotted lines) by the decentralized MPFs. Using the vehicles' altitude measurements, ranging measurements

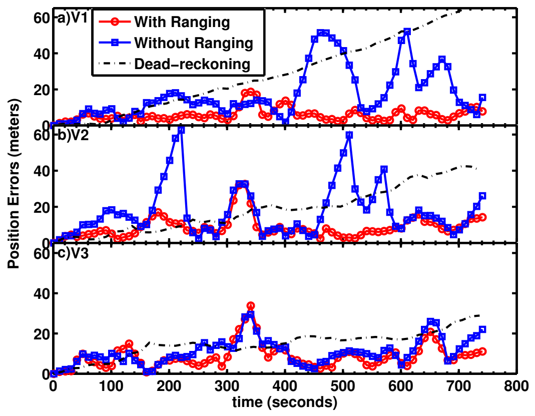
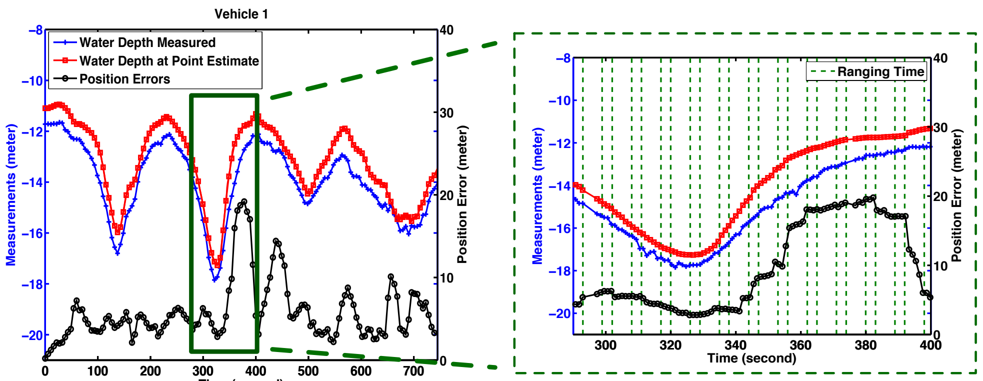


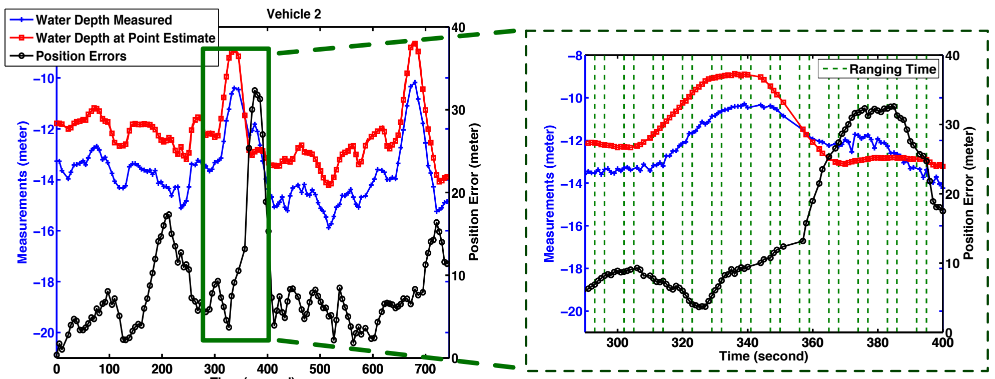
FIGURE 4.19: The average position estimation errors for all three vehicles over 100 localization runs using the same paths. The errors are lower when both the bathymetry information and acoustic communication are used for cooperative localization.

and filter information shared among the vehicles, the individual filter managed to closely track the executed trajectories, with the exception at around 300 - 400 seconds where the filter's estimation errors momentarily increased. However, without acoustic communications among the vehicles, V1 and V2's filter diverged on quite a few occasions as shown in Fig. 4.19.

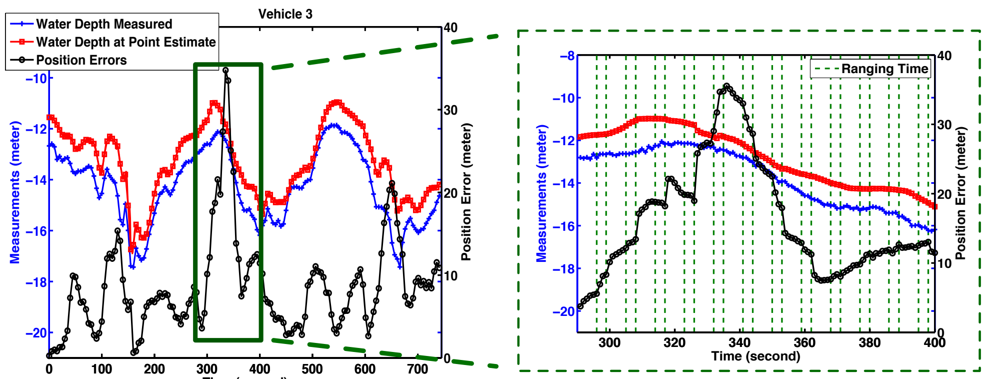
The detailed position estimation errors, water depths measured by the vehicles as well as the water depths at the point estimates (see (4.10)) of the particle sets are shown in Fig. 4.20. Close inspection of plots (right column of Fig. 4.20) at around 300 $\sim$ 400 seconds mark reveals that the divergence of the filter may be due to the similarity of terrain profiles along the vehicles' trajectories around that particular period of the mission times. This is illustrated by the similarities of terrain gradients from 290 $\sim$ 330 seconds marks of the enlarged plots. Besides, Fig. 4.19-(b,c) also show that both the V2 and V3's filter failed to track their positions at around the same time, using the bathymetry information alone (blue-squared line). When acoustic communications are available among the vehicles, the feedback of the wrong estimates by V2 and V3 cause the filter's estimation errors of V1 (Fig. 4.19-(a)) to increase too.



(a)



(b)



(c)

FIGURE 4.20: Position estimation errors and water depth measurements of all the AUVs are shown on the left while the enlarged plots between second 300 - 400 are shown on the right. The similarities of the terrain profiles caused the filter to diverged momentarily. (a) Position errors and water depth measurements of V1. (b) Position errors and water depth measurements of V2. (c) Position errors and water depth measurements of V3.

4.6 Sensitivity Analysis

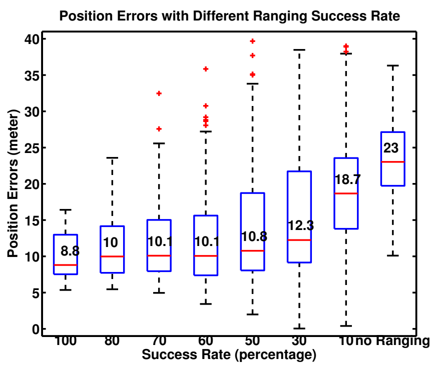
In an autonomous underwater mission, the underwater acoustic communication is often lossy and unreliable, while the sensors carried onboard the low-cost AUVs are often corrupted by noise, introduced by internal electro-mechanical systems or external environmental factors. Since the decentralized MPF mainly relies on the vehicles' abilities to measure bathymetry information and performing inter-vehicle acoustic communication, it is imperative to investigate their impacts on the decentralized MPF's performance.

In this section, we make use of the data from Section 4.5.2 to conduct sensitivity analysis on the performance of the resultant decentralized MPF. This is done by varying the ranging frequency and success rate among the vehicles in the team, as well as by introducing different levels of sensor noise to the vehicles' single-beam echo sounder. We present the results in terms of position estimation errors accumulated by the decentralized MPF at the end of each mission runs.

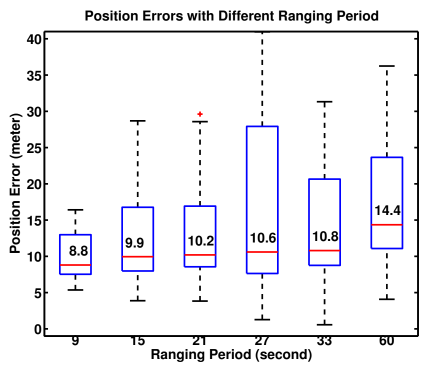
4.6.1 Influence of Ranging Frequency and Success Rate

The results presented in Section 4.5 illustrate the importance of acoustic communication in cooperative localization. However, the assumption of 100 % success rate for the simulated acoustic communication among the vehicles may be impractical especially in a lossy and unreliable underwater acoustic channel. Also, the ranging period of 9 seconds may be too short, as each vehicle has to broadcast a signal and decode two separate signals sent by the remaining two vehicles in the team. In this section, we repeated the offline localization runs with different acoustic ranging success rates, to investigate their impact on the performance of the decentralized MPF. We also carried out separate tests where the broadcast period of each vehicle in the team was increased from 9 seconds (as stated in Table 4.1) to 1 minute.

Fig. 4.21(a) shows the boxplots of the position estimation errors against acoustic communication success rate. Overall, the decentralized MPF is robust against the acoustic communication loss up to approximately 50 % success rate. Besides that, it is



(a)



(b)

FIGURE 4.21: (a) Position estimation erros against different success rate of the acoustic communications. The decentralized MPF is robust again communication loss up to around 50 % success rate. (b) Position estimation errors of the filters with difference ranging period. The increase in position errors are not significant when the communication period is increased to 1 minute.

also robust against slower update rate as shown in Fig. 4.21(b) where the position estimation errors only increase slightly when the communication period was lengthened to 1 minute for each of the vehicles.

4.6.2 Influence of Sensor Noise Level

The assumption of 0.05 m for the standard deviation of the sensor measurement noise may be too small in some cases, especially for AUVs operating in deeper water. Besides, low-cost AUVs may be fitted with lower accuracy sensors due to the limitation of budget. To study how the measurement noise may impact on the decentralized MPF's performance, we repeated the same set of offline localization runs, but with the standard deviation of the measurement noise increased from 0.05 m to 1 m. Fig. 4.22 shows the distribution of the position estimation errors when the measurement noises were increased. In this case, the decentralized MPF is robust against noisy sensors with the noise level up to 0.5 m. The errors increase significantly when the sensor noise was increased above that level. Other factors like the depth sensor's sampling frequency and terrain variability also affect the filter's localization performance, and will need to be considered in more detail.

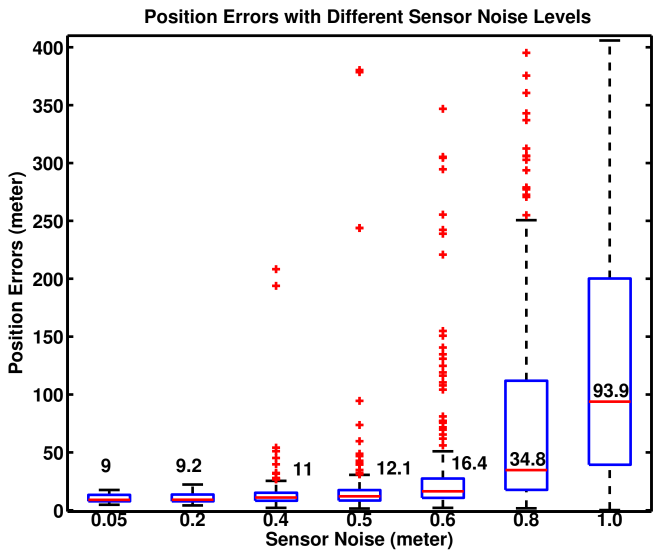


FIGURE 4.22: Position estimation errors of the decentralized MPF when the altitude sensor is corrupted with different Gaussian noise levels.

4.7 Discussion

Although the proposed cooperative filters do not take the correlation of the estimation into account, the issue of overconfidence is not be serious enough to cause the filter to diverge, as shown in Fig. 4.8 where the position errors for the cooperative filters remained low for most of the simulation runs. Tracking the correlation may further improve the performance of the cooperative filters, provided that there is sufficient communication bandwidth for the vehicles to share the correlation information.

During the PV's filter information broadcast, some information is lost because only a single mode of the particle's distribution is estimated (via equations (4.10) and (4.11)) and shared with other vehicles. Even though it is impractical to transmit all the particles' locations via the bandwidth-limited underwater communication link, it may be beneficial to estimate and transmit multiple modes of the distribution, if they exist. This will allow the RV to better approximate the PV's true particle distribution during the measurement update step, and potentially improve the filter's performance. However, the approach comes with a cost of longer packet size for the information broadcast, and may potentially decrease the transmission success rate and cause the channel throughput to be lower.

The filter's state space for tracking and localization applications also typically include the vehicle's velocity and heading estimates. Due to the limitation on the computation power and the absence of a velocity measurement in the sensor-limited AUVs, we do not consider them in this work. However, a natural extension for this work is to include these estimates in the future filter's state space. Although the increment in the dimension of the filter's state space will inevitably increases the requirement of the size of the particle set for the filter to converge, it would be instructive to assess the improvement in the localization accuracy attributed to the extension.

4.8 Summary

In this chapter, we showed that it is feasible for a team AUVs equipped with single-beam altimeter, depth sensor, and acoustic modem to perform cooperative localization. In particular, we employed the marginalized particle filtering in a distributed manner in each of the vehicles and extended the filter's measurement model to incorporate the information broadcast by other vehicles in the team.

We showed that both the bathymetry information and inter-vehicle acoustic communication among the vehicles are crucial for sensor-limited, low-cost AUVs to perform cooperative localization. Empirical studies using simulated data demonstrated the benefits of the decentralized filter against dead-reckoning navigation, as well as showcased its ability in estimating the vehicles' position under the influence of ocean current and sensor biases. Finally, offline localization using data from field experiments also validated the effectiveness of the cooperative localization algorithm at localizing a team of three vehicles, and showed its robustness against sensor noise and unreliable communication channels.

Chapter 5

Command and Control System for Autonomous Underwater Vehicles

In this chapter, we focus on the design and development of a C2 system for a single AUV that is robust and easily extensible to accommodate the requirements of multi-vehicle missions. We adopt a multi-agent approach where mission, navigation and vehicle control tasks are allocated to individual software agents that are arranged in a hierarchical order according to their corresponding control responsibilities. The agent-based modeling approach provides separation of concerns in developing agents that handle various C2 tasks, thus clearly defining each agent's responsibilities and interfaces. At the C2 system's Supervisory level, we adopt the Backseat Driver (BD) paradigm introduced in [52, 79], where the mission tasks are handled by individual DB agents that implement specific algorithms to satisfy the tasks' objective. We show that this approach enables the C2 system to cope with new mission requirements easily by adding new BD agents.

Next, we present the software implementation for the C2 system and showcase its robustness in executing various single and multi-vehicle missions via Software-In-The-Loop simulations. Finally, we illustrate the use of the C2 system in an AUV called STARFISH (Fig. 1.1(c)). The STARFISH AUV is a low-cost modular AUV developed as a research platform and is capable of being extended with various sensor payload

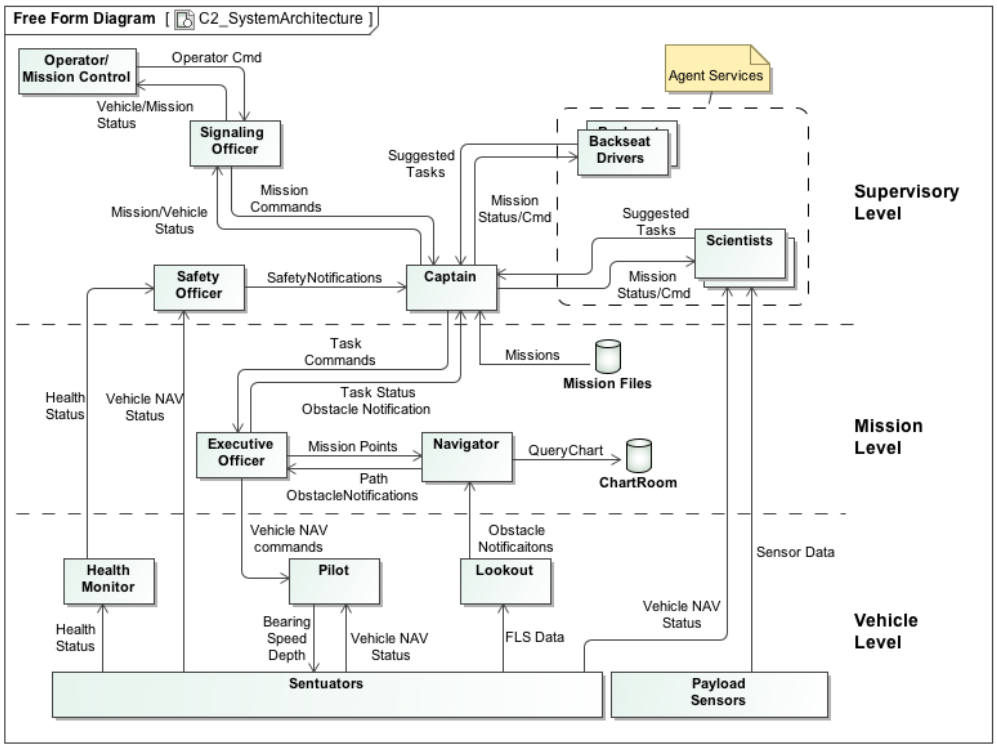


FIGURE 5.1: Overview of the Agent-based Command and Control (C2) System. The allocation of C2 responsibilities to different agents at different control hierarchy provides an explicit view of the control flow within the control architecture.

modules for sensing and monitoring missions. The availability of different payload modules on the STARFISH AUV allows us to demonstrate the extensibility of the C2 system in coping with different mission requirements, through various single and multiple vehicles mission scenarios. To further illustrate the portability of the C2 system, we also present the mission scenario where the C2 system was ported into a different robotic system for a collective localization mission, with minimum modifications to its software architecture. For related publications of the C2 system, we refer the reader to [3, 80, 81].

5.1 Heirarchical Agent-based Control Architecture

The C2 system is based on a hybrid-hierarchical model as shown in Fig. 5.1. It adopts a deliberative-reactive architecture that consists of a set of interacting agents (termed C2 agents) organized in three different levels within the control hierarchy: Supervisory level, Mission level and Vehicle level. Each of the control levels assumes different C2 responsibilities and defines the responsive requirements for an agent located within it. The Supervisory level is in charge of making high-level mission decisions, monitoring the vehicle status and handling the communication with the operator/mothership and/or other vehicles. The agents within this level maintain internal states and deliberate upon the state information for decision making. The Vehicle level is responsible for performing low-level vehicle control and interacts reactively with Sentuator (vehicle sensors and actuators) [82] components to generate the desired maneuvering behaviors. The agents in the Mission level act as the arbitrators among the agents in the Supervisory and Vehicle level by translating the mission goals into collision-free path waypoints for the vehicle to follow.

Each agent has its private data and implements its own algorithms depending on the assigned responsibilities. All the agents are self-contained and have a uniform software interface to facilitate inter-agent communication via a message-passing mechanism. The vehicle's C2 tasks are achieved through the interaction and cooperation among the involved agents. The agent-based design provides flexibility in terms of software implementation; rather than modify existing software components, new agents that adhere to the software interface, but implement different algorithms, can be built and loaded to replace the existing agent when necessary.

The C2 system design offers many benefits. The hybrid architecture allows high level mission control to behave deliberatively while decoupling the low level reactive vehicle control. It also defines the real-time requirements of the agents residing within each control level. The breaking down of the C2 tasks into individual agents presents an explicit view of the clearly defined control responsibilities at different levels of the control hierarchy. The architecture provides an important guideline for agent developers and ensures the resultant C2 system's integrity.

5.1.1 Agents Responsibilities

As mentioned in the previous section, different C2 responsibilities are handled by different C2 agents, and the collective interaction among the agents ensures the mission objectives are achieved. The C2 agents are shown in Fig. 5.1, and their C2 responsibilities are briefly described below:

- **Captain:**
  - Starts, coordinates, oversees and controls the execution of missions.
  - Listens to the safety notifications from the Safety Officer and aborts the mission if any abnormality is observed.
  - Executes Operator's commands delivered by the Signaling Officer and broadcasts mission planning requests to the Agent Services and assigns the mission point generator. More details are discussed in section 5.1.2.
- **Signaling Officer:**
  - Acts as the AUV's external communication node: uses WIFI or Global System for Mobile Communications (GSM) when the vehicle is at the surface; uses acoustic modem when the vehicle is submerged.
  - Updates the Operator with the latest mission and AUV status periodically.
  - Decodes the mission commands received and passes them to the Captain.
- **Safety Officer:**
  - Detects any abnormality reported by the Health Monitor and monitors vehicle navigational status (maximum pitch, roll, depth and minimum altitude) and system resources.
  - Ensures that the vehicle navigates only within the geofenced area defined by the Operator.
- **Backseat Drivers and Scientists:**
  - Generate mission points according to the mission specification.
  - Interrupts the Captain with new mission points when needed. The Scientist is a type of the Backseat Driver that controls and interacts with optional payload modules attached to the AUV. More details are discussed in section 5.1.2.

- **Executive Officer:**
  - Receives mission tasks in the form of mission points and passes them to the Navigator for waypoint planning. Once waypoints are received, sends them to the Pilot for execution.
  - Stops the execution of the Pilot and notifies the Captain if the vehicle fails to avoid obstacles that are detected in the mission path.
- **Navigator:**
  - Plans a collision-free path from one mission point to the next. Re-plans the waypoints if obstacles are detected in the path.
  - Notifies the Executive Officer if a collision free path to the next mission point is not found.
  - Marks the newly detected obstacles in the mission Chart Room.
  - Executes different waypoint planning algorithms according to the mission specifications.
- **Health Monitor:**
  - All the sensor and actuator (Sentuators) drivers in the vehicle implement a health reporting mechanism. The Health Monitor collects the information and analyzes the severity when Sentuators are found unhealthy.
  - Notifies the Safety Officer if the severity is high.
- **Pilot:**
  - Translates the waypoints into primitive vehicle control (bearing, speed, depth and altitude control set-points). This is done according to the mode of waypoint execution. Two modes are currently implemented: Waypoint Following and Path Following.
  - Sends the vehicle control to the vehicle's control system.
  - Broadcasts the waypoint's execution status periodically.
- **Lookout:**

  - Processes and analyzes the sensor data provided by the Forward Looking Sonar for obstacle detection.
  - Informs the Navigator with the range and bearing of newly detected obstacles.

- **Mission Files and Chart Room:**
  - Stores mission files and mission area bathymetry.

5.1.2 Backseat Driver Paradigm

To allow for different mission behaviors and cater to various payload modules with potentially different mission requirements, the Supervisory level adopts a backseat driver paradigm where mission decisions are made based on the input provided by a pool of BD agents. This pool is termed as Agent Services (AS). Each BD agent in the AS implements different algorithms and monitors various sensor data to generate input in the form of mission points, which when accepted for execution, achieve a specific mission task. Depending on the requirements of the mission, the BD agents can be tasked to generate mission point-sets in some pre-planned pattern, or to generate single mission point iteratively adapting to sensor data as the mission progresses.

The Scientist agents are special BD agents that interact exclusively with the payload modules. This enables the payload developers to implement algorithms that make use of the payload sensor data to adaptively generate mission points for sensing and tracking missions. A Scientist agent can be developed to handle each newly added payload module. Optionally, the payload module can also provide its own Scientist agent, which is separately loaded in the module's computing node, instead of the vehicle's main computing node. This option decouples the development of the payload modules to that of the main vehicle and increases the reconfigurability of the modular-based AUVs. Depending on the vehicle's final payload setup, different Scientist agents can be loaded to generate mission points to achieve the overall mission objectives. However, this does not prevent the introduction of payload modules like the DVL that provide only

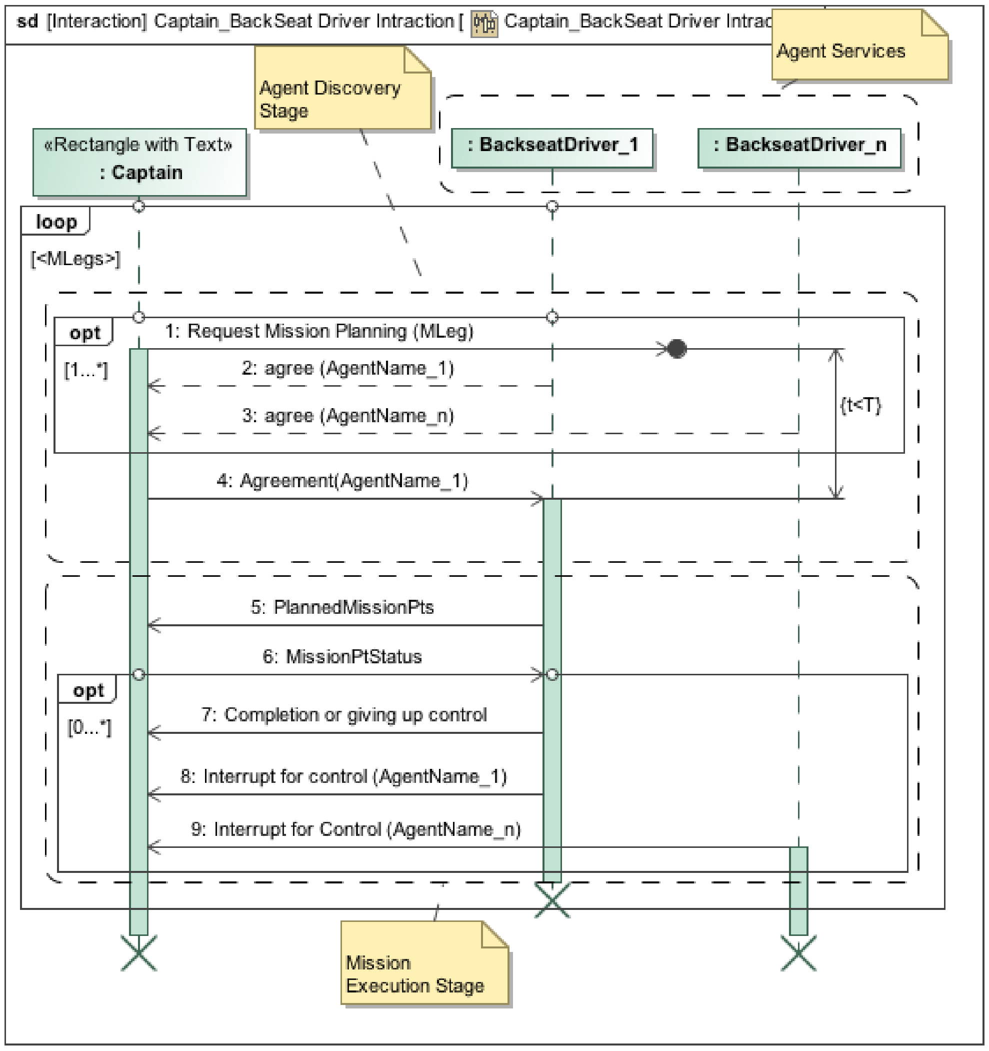

sd [Interaction] Captain_BackSeat Driver Intraction [Captain_BackSeat Driver Intrac
Agent Services
Agent Discovery Stage
«Rectangle with Text» : Captain
: BackseatDriver_1
: BackseatDriver_n
loop
[<MLegs>]
opt
[1...*]
1: Request Mission Planning (MLeg)
2: agree (AgentName_1)
3: agree (AgentName_n)
{t<T}
4: Agreement(AgentName_1)
5: PlannedMissionPts
6: MissionPtStatus
opt
[0...*]
7: Completion or giving up control
8: Interrupt for control (AgentName_1)
9: Interrupt for Control (AgentName_n)
Mission Execution Stage

FIGURE 5.2: Sequence diagram showing the interactions between the Captain and the Agent Services. During a mission, the interaction consists of two different stages: Agent Discovery Stage and Mission Execution Stage

navigational data for the onboard positioning system, and do not need a corresponding Scientist agent.

The BD design paradigm relies strongly on the interactions between the Captain agent and the pool of BD agents. Essentially, the interactions happen in two different stages during a mission: Agent Discovery stage and Mission Execution stage. The Agent Discovery stage takes place when the mission starts. A mission consists of at least one or more mission legs (`MLegs`). Each `MLeg` encodes the `MLeg_type`, position and parameters that define speed, waypoint radius, maximum depth and minimum altitude. When the start command is received from the operator, the Captain agent broadcasts a request for mission planning with the corresponding `MLeg`. BD agents that are programmed to handle the particular `MLeg_type` will respond.

The Captain agent waits for a time period, T, and collects all the BD agents' responses. If no response is received for the requested `MLeg_type`, the mission is aborted since there is no BD agent that is capable of translating the mission leg command into mission points. If only one response is received, the corresponding BD agent will be assigned for mission point generation. However, in the case where more than one response is received, the Captain agent identifies one of the BD agents as the mission point generator according to a specific preference. In this thesis, we adopted a simple priority based selection scheme. More complex scheme like the market-based approach [83] or automated planning approach like T-REX [84] can be adopted when needed. Once selected, the assigned BD agent is notified with an agreement and contracted as the mission point generator.

During the Mission Execution stage, the assigned BD agent provides the generated mission points to the Captain agent and monitors the mission status. This process is repeated until the completion of the mission leg or the BD agent gives up the control (due to failure in achieving the mission leg's objective). In the later case, the mission is aborted for safety reasons.

Depending on the mission requirements and the BD agents' configuration, the current executing mission point can be aborted and replaced with new mission points by the same BD agent or by another BD agent to pursue tasks of interest with higher priority. This approach allows the C2 system to adapt the mission during execution. A mission that requires the vehicle to survey an area and track a feature of interest, if and when it is detected, benefits from the C2 system's adaptive capability. Detailed interactions between the Captain agent and the BD agents during a sample mission are shown in Fig. 5.2. The chosen mission scenario presented in section 5.3 showcases the usefulness of the adaptive capability.

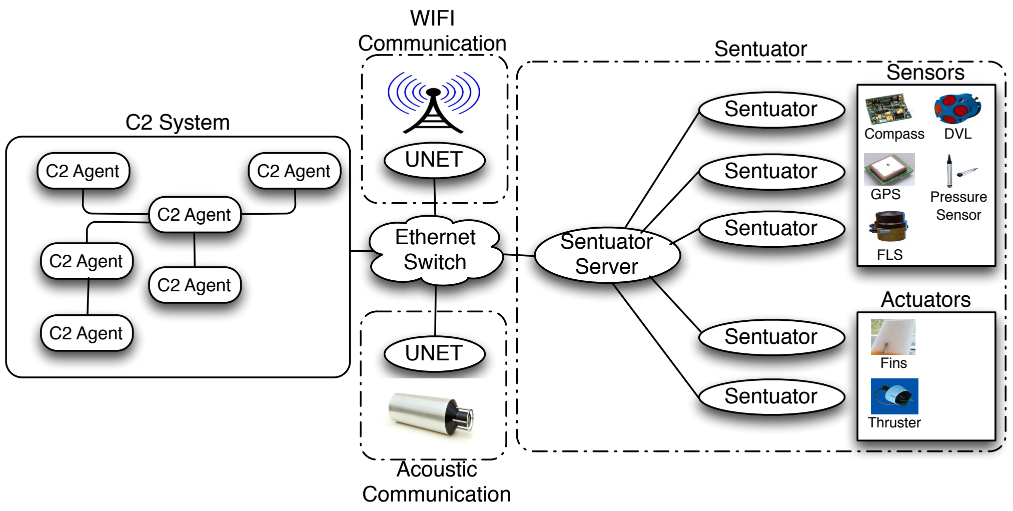


FIGURE 5.3: The overview of the vehicle's software architecture. The C2 agents receive vehicle's sensor data and send actuctor commands via the Sentuator [82]. The communication with the operator or another AUV (WIFI and acoustic communication) is facilitated by the UnetStack [85].

5.2 Software Architecture

The software architecture defines how the structure of different software components in the AUV are built and integrated to form a fully functional system. Fig. 5.3 shows the overall system software architecture of the STARFISH AUV. The software components are distributed among the onboard Single Board Computer (SBC) and MicroController Units (MCUs). The C2 agents interact with other software components via message passing mechanism via the Ethernet switch, which acts as the centralized communication hub in the vehicle. The communication between the C2 agents with the system's sensors and actuators are facilitated by the Sentuator components and Sentuator Server described in [82], while the communication with the operator and other vehicles is achieved using the network stack provided by the UnetStack [85].

5.2.1 Command and Control Agents

The C2 agents are the basic functional units of the whole C2 system. They are built using the *fjåge lightweight agent framework* [1] (*fjåge*). Each of the C2 agents is self-contained and defines its own data structure and parameters depending on its tasks and responsibilities. The agent's activity is governed by a finite state machine, which checks the agent's current state at every agent's life cycle, and executes the routines that are defined for that particular agent state to generate the desired behaviors or outputs. The implementation of state machine in C2 agents is to facilitate controllability and observability in the control architecture, both of which are important in a C2 system where supervisory C2 agents at the higher level of control hierarchy can monitor and command the behavior of the lower level C2 agents.

Once defined, C2 agents have to be added to a *fjåge* software container in order to function. The fjåge container manages agents' life cycle, delivers agents' messages and provides services and topics discovery. Inter-agent's interaction is achieved through service registration or topic subscription, using message passing mechanism. A message in *fjåge* is tagged with "performative" that defines the purpose of the message, and can be extended to carry relevant content fields for the receiving agent. Detailed documentations of available "performatives" can be found in *fjåge*. A list is provided in Table. 5.1 for reference.

During the software initialization cycle, a C2 agent can register for the services it provides, and subscribe to the topics of the agent's interest. At anytime during an agent's life cycle, a message can be broadcast as a topic and forwarded by the container to the agents who subscript to the topic. On the other hand, a C2 agent can request for a particular type of service by sending a request message to the C2 agent that provides that service. Results in the form of a message is replied by the servicing agent if the request succeeded; otherwise, a failed message is returned. A list of available C2 services and C2 topics can be found in Appendix B.

[1] https://github.com/org-arl/fjage

TABLE 5.1: LIST OF PERFORMATIVES

Performatives	Description
AGREE	Agree to performing the requested action.
CANCEL	Cancel pending request.
CONFIRM	Confirm that the answer to a query is true.
DISCONFIRM	Confirm that the answer to a query is false.
FAILURE	Notification of failure to perform a requested or agreed action.
INFORM	Notification of an event.
NOT_UNDERSTOOD	Notification that a message was not understood.
CFP	Call for proposal.
PROPOSE	Response for CFP.
QUERY_IF	Query if some statement is true or false.
REFUSE	Refuse to perform the requested action.
REQUEST	Request an action to be performed.

As an example, whenever a mission point is received from the Captain, the Executive Officer sends a request message (with "`performative=REQUEST`") to the Navigator to plan a path to that mission point, either from the vehicle's current location, or from the previous mission point if applicable. A message (with "`performative=AGREE`") containing an array of way points leading to that mission point will be returned by the Navigator if a collision free path is successfully found; otherwise, an empty message with "`performative=FAILURE`" can be returned, signifying failing to plan any feasible path. A set of messages has been defined and implemented for the C2 system and can be found in the Appendix B.

5.2.1.1 Backseat Driver Agent

In order to standardize the interaction between the Captain and the AS, as depicted in Fig. 5.2, the BD agent base-class defines a set of abstract message-handler methods that must be implemented in the inherited-class. These message-handlers are listed as below:

1. **Call-For-Participation**

   `void handleCFPReq(Message::CFPReq)`

Handles the Call-For-Participation request (`CFPReq`) message from the Captain. The implementing agent must check the `MLeg_type` encoded in the message. If the agent is capable of handling the request, a response message (`CFPRes`) with "`performative=PROPOSE`" and the agent's ID must be replied. Non-participating BD agents can remain silent, and will be ignored by the Captain.

2. **Mission Point Request**

   `void handleMissionPtReq(Message::MissionPtReq)`

   If the BD agent is contracted as the mission point generator, mission point request (`MissionPtReq`) message will be sent by the Captain during mission execution. Depending on the implemented algorithms and mission objectives, the BD agent must reply with a mission point response (`MissionPtRes`) message, containing a single mission point or an array of mission points, together with "`performative=INFORM`". The contracted BD agent can surrender its mission point generator's responsibility at any time when the mission objective has been achieved, or failed. In either case, a `MissionPtRes` message can be sent to Captain notifying completion or failure depending on the message's "performative".

3. **Abort**

   `void onStop( )`

   This method is called by the a default message handler that handles abort messages sent by the Captain or the Safety Officer due to any vehicle abnormality. It is provided such that the BD agent has the opportunity to cease the operation of mission point generation, and performs necessary house-keeping routines to prevent system crash.

By default, the BD agent base-class registers to `BACKSEATDRIVER` service and subscribes to `ABORTSIGNAL` topic. This allows the Captain to broadcast a `CFPReq` message to all the BD agents in the AS regardless of the number of BD agents that exist. A particular `MLeg` is considered executable or feasible when one or more `CFPRes` messages are received by the Captain, otherwise the mission is deemed infeasible and

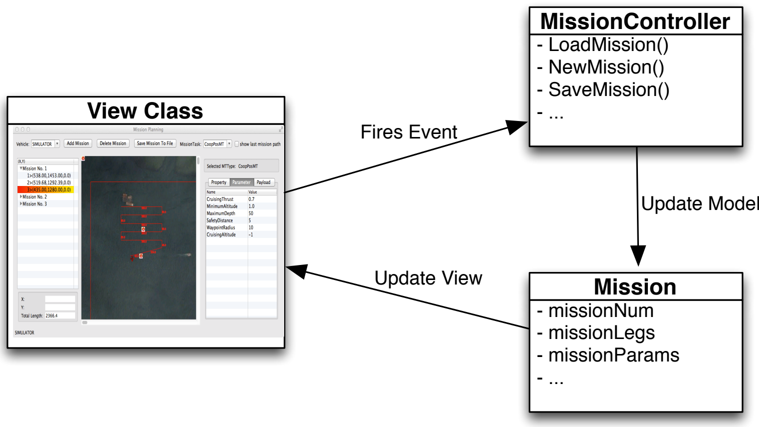


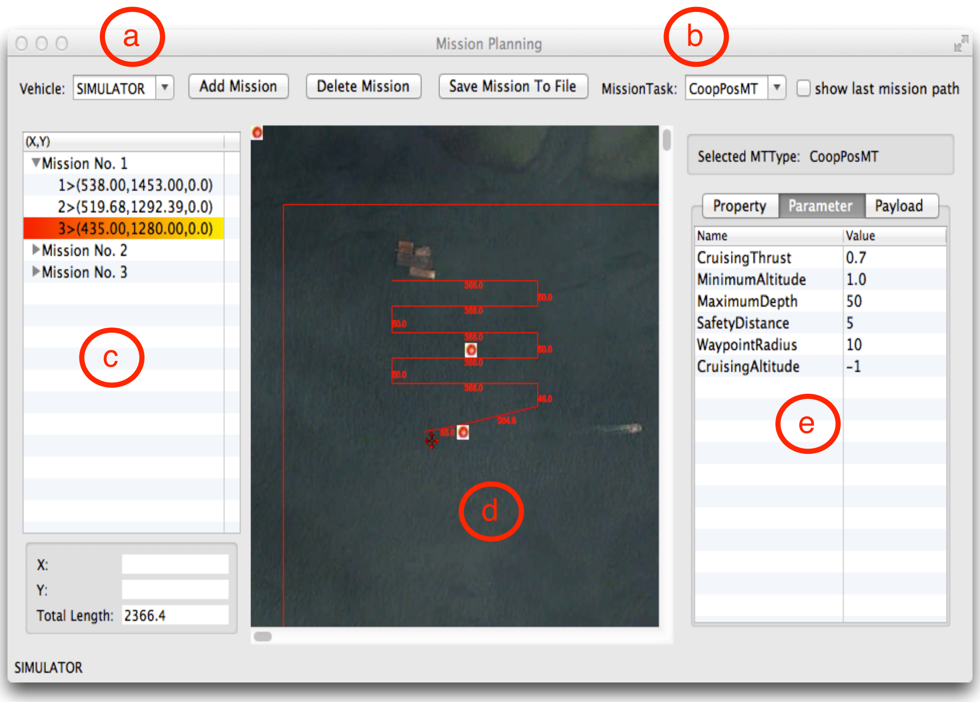
FIGURE 5.4: The Model-View-Controller design pattern of the mission planning component.

the mission is aborted. The same advantage applies to the default abort topic subscription: regardless of the size of AS agent pool, all the BD agents will be put into the stop state whenever an abort message is broadcast on the `ABORTSIGNAL` topic, preventing further interaction from the BD agents, which may cause the C2 system to crash.

5.2.2 Mission Planning

Mission planning is an integral component of the Operator Mission Control (Fig. 5.1). It adopts a simple Model-View-Controller (MVC) design pattern and provides a Graphical User Interface (GUI) for the operator to plan, modify, delete, upload and download missions to and from the AUVs. Fig. 5.4 shows the MVC classes built for the mission planning component. The View class displays the mission information stored in the Mission (model) class, and fires events according to the operator's interaction with the GUI. Depending on the event received, the MissionController class executes different methods to update the Mission class.

Fig. 5.5 shows the GUI for mission planning. It allows the operator to plan missions for all the vehicles within a single application. Whenever a vehicle is selected (via Fig. 5.5-(a)), all the missions belonging to that particular vehicle, together with their corresponding mission legs, will be displayed in the form of a Tree-view panel, as shown in Fig. 5.5-(c). In the mean time, the selected (highlighted) mission's



Vehicle: SIMULATOR

MissionTask: CoopPosMT

(X,Y)
- Mission No. 1
  - 1>(538.00,1453.00,0.0)
  - 2>(519.68,1292.39,0.0)
  - 3>(435.00,1280.00,0.0)
- Mission No. 2
- Mission No. 3

X:
Y:
Total Length: 2366.4

Selected MTType: CoopPosMT

Name	Value
CruisingThrust	0.7
MinimumAltitude	1.0
MaximumDepth	50
SafetyDistance	5
WaypointRadius	10
CruisingAltitude	-1

SIMULATOR

FIGURE 5.5: The mission planning GUI. (a) Drop-down button for selecting vehicle mission file. (b) Drop-down button for selecting mission task of a particular mission leg. (c) Tree-view showing all the missions in the vehicle's mission file, expanded to show all the mission legs of a particular mission. (d). Canvas showing map of the mission area. All the mission task icons support drag-and-drop interaction. (e). Tabs showing mission leg's related information.

planned path is also plotted in the map canvas (Fig. 5.5-(d)), while the individual mission leg's information is displayed in a Tab-panel shown in Fig. 5.5-(e). A new mission can be easily added by clicking the "Add Mission" button, while new mission legs can be added by first choosing the MissionTask from the Drop-down button shown in Fig. 5.5-(b), and then clicking the desired location in the map canvas. Besides, the desired location of a particular mission leg can be modified simply by drag-and-drop operations on the mission leg's icons in the map canvas.

5.2.3 Mission Execution

After the mission is planned and saved, the mission file can be uploaded to the AUVs for execution. Fig. 5.6 shows the mission execution panel which allows the operator to send mission and vehicle commands to the AUVs. Depending on the number of

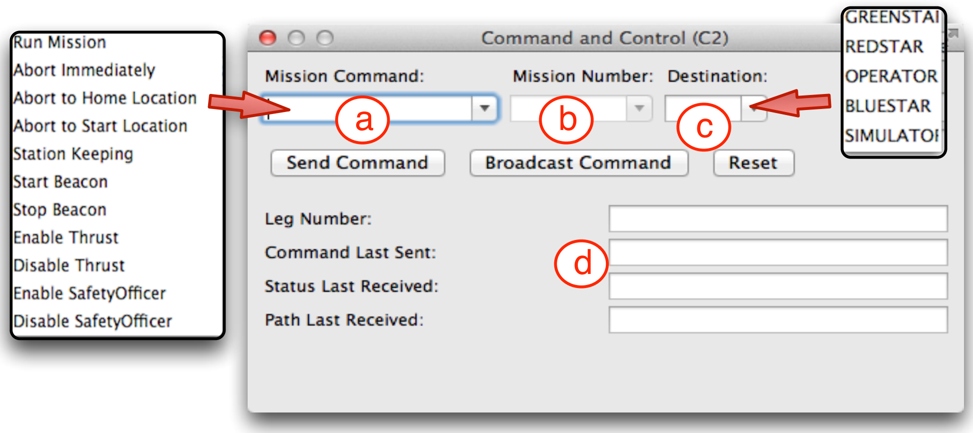


FIGURE 5.6: The mission execution GUI. (a) Drop-down button for selecting mission commands with available mission commands shown in the adjacent text box. (b) Drop-down button for selecting the mission to execute. (c) Drop-down button for selecting the destination of the mission command. It can be one of the assets listed in the adjacent text box (d). Mission status panels, showing mission messages received through acoustic communication.

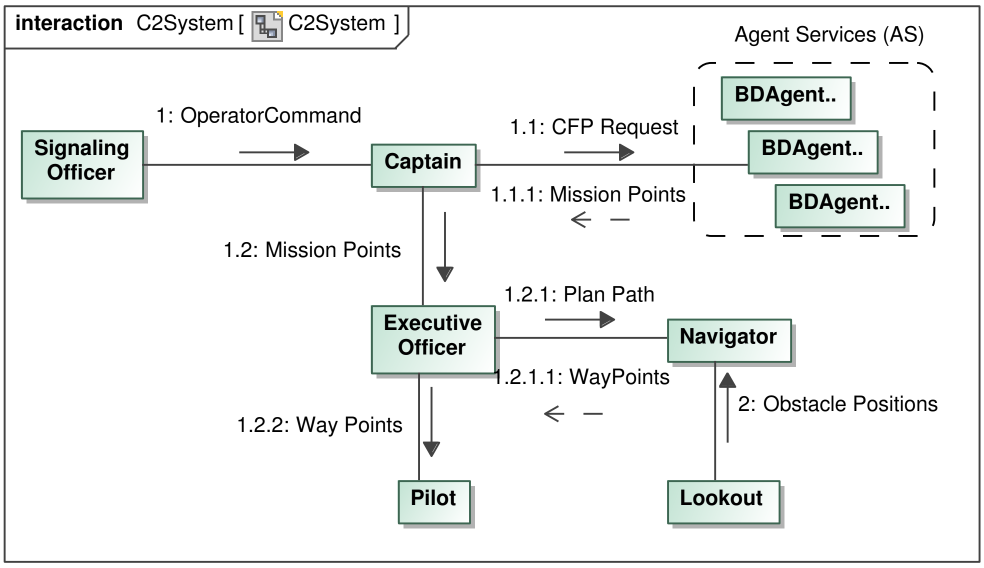


FIGURE 5.7: Communication diagram showing the message passing and interaction among the C2 agents during mission execution.

missions planned (as shown in Fig. 5.5-(c)), once the mission file has been uploaded to the AUV, the corresponding mission numbers will be shown in Fig. 5.6-(b). Whenever acoustic communication is available between the AUV and the operator, mission status of a particular AUV will be displayed in the mission status panel (Fig. 5.6-(d)).

The communication diagram shown in Fig. 5.7 further illustrates the interactions and message passings among the C2 agents during a mission execution. Whenever an operator command is received by the Signaling Officer, it decodes the command and relays it to the Captain. The Captain retrieves the mission legs from the commanded mission and broadcasts a CFP request message to all the BD agents in the AS. The BD agent that is developed to handle the particular mission leg translates the encoded mission objective into mission points and replies them to the Captain. After the mission points are checked to ensure they are within the mission geo-fence, they are relayed to the Executive Officer for execution. One at a time, the Executive Officer sends the mission points to the Navigator for path planning. If a collision-free path can be found, the Navigator feeds back the path to the Executive Officer in an array of way points. These way points are then sent to the Pilot for navigation. During navigation, the Lookout agent process data from obstacle sensor and sends obstacle positions to the Navigator whenever an obstacle is detected. The Navigator performs collision-detection using the obstacle data and re-plan the vehicle's path when necessary. The process is repeated until all the mission legs are executed, or when the mission is aborted due to the failure of the Navigator in finding a collision-free path.

5.3 Simulations

The C2 system described in the previous sections has been implemented using the *fjåge*. The same C2 system can also be easily implemented using other popular middleware such as the DSAAV [82], ROS [86] or MOOS [87]. An AUV simulator has also been implemented in a separate fjåge agent, based on a simplified AUV's dynamical model. The simulator accepts actuator commands and produces simulated sensor data, as if they were generated by a physical AUV. The agent-based design and inter-agent communication via message passing mechanism decouples the C2 system from the physical vehicle's Sentuators, and allows the resultant C2 system to be validated through simulation by simply exchanging the Ethernet switch shown in Fig. 5.3 with the AUV simulator. Once tested, the same C2 system can be loaded into a physical AUV for field experiments without any modification. The simulation can be easily extended to simulate a

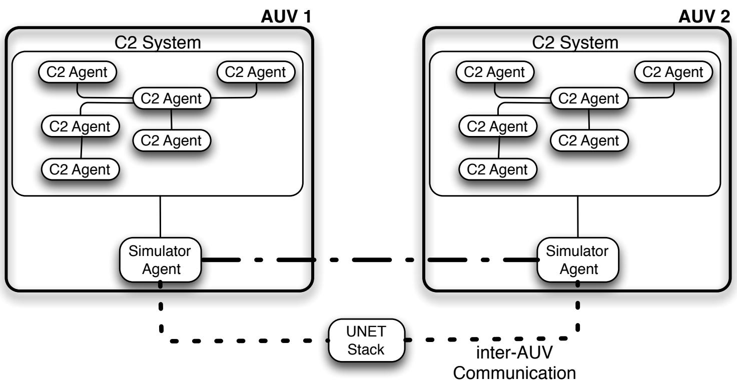


FIGURE 5.8: Multi-AUV simulation can be easily implemented by simply passing the message from one vehicle to another (dash-dotted line), or through the UnetStack [85] software stack (dotted line) if more realistic underwater communication performance is desired.

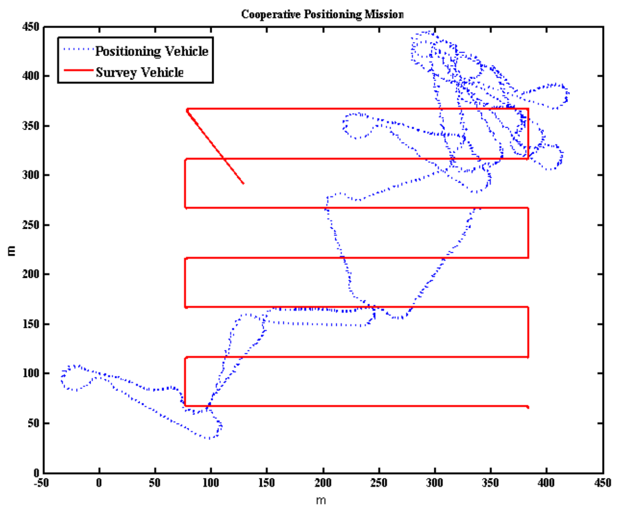


FIGURE 5.9: Simulation results show the resultant path generated by the positioning AUV (blue dotted line) to minimize the position errors of the Survey AUV (red solid line).

multi-vehicle cooperative mission by passing the message from one simulator directly to another simulator, or through the UnetStack if more realistic underwater communication performance is desired (Fig. 5.8). This Simulation-In-The-Loop methodology expedites the design and development of new C2 capabilities and shorten mission turn-around time.

In the simulation, we demonstrate the C2 system's capability in performing a co-operative positioning mission. We refer the reader to Chapter 3 for detailed algorithms.

The cooperative algorithm was implemented within a BD agent (`BD_Coop`) in the positioning vehicle while the lawn mowing survey path was generated by another BD agent (`BD_Lawnmower`) in the survey vehicle. During the mission, the `BD_Coop` generated the mission points adaptively depending on the supported survey vehicle's current position, such that when range and position information were exchanged between the AUVs, the position errors of the survey vehicle were minimized. Fig. 5.9 shows the resultant path of the positioning vehicle (dotted blue line) to support a survey mission (red solid line).

Even though the trajectories were generated based on a simplified dynamical model, the simulation allows the developers to test and debug the agents' behaviors within a single or multiple vehicles mission scenarios. In fact, the cooperative positioning mission simulation has helped us tremendously in preparing and practicing for the field experiment reported in Section 5.18.

5.4 Field Experiments

In this section, we present various field experiment scenarios which make use of the C2 system outlined in the previous sections. The main objective of this section is to illustrate the C2 system's extendability and portability in coping with different mission requirements, in both single and multiple AUVs mission scenarios. For each of the scenarios, we focus only on the C2 system's configurations, field deployments and brief snapshots of the experimental outcomes, and refer the interested readers to the respective publications for the detailed algorithmic developments and experimental results.

The experiments were conducted either in Pandan Reservoir, Singapore, or around the Singapore coastal water. The STARFISH AUV shown in Fig. 1.1(c) was used for the experiments, with different payload sections fitted depending on the mission objectives. Whenever the AUV carries out a mission on the surface, GPS data are logged and used as a ground truth, otherwise, the positions estimated by the onboard positioning system is used during underwater navigation. To further illustrate the C2 systems' portability,

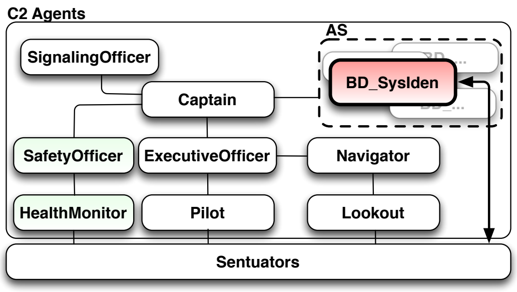


FIGURE 5.10: C2 agents and the `BD_SysIden` agent for system identification missions. The `BD_SysIden` agent can interact with the vehicle's Sentuators directly when performing the idenfication algorithms, while both the SafetyOfficer and HealthMonitor agents can ensure the vehicle's safety throughout the process.

we also showcase an experiment where the C2 system was ported and deployed onto another type AUV for collective mission.

5.4.1 System Identification Mission

The AUV's low-level vehicle control performance determines its capabilities in achieving mission objectives. In a modular AUV like the STARFISH AUV, payload sections can be added or removed depending on the mission's requirements. Different vehicle configurations change the vehicle's dynamics, and thus, affects the vehicle control performance. System identification [88] mission aims to automatically identify the new vehicle dynamic online, and retune the onboard controller.

A BD agent (`BD_SysIden`) has been developed for the purpose of AUV's system identification and deployed in the AS as shown Fig. 5.10. Whenever the identification mission is underway, the C2 system stops its control of the vehicle so that the `BD_SysIden` can interact directly with the vehicle's Sentuators and perform the identification algorithms. System identification algorithms generally require excitation signals to be injected to the vehicle's controller, so that the vehicle's dynamic response can be monitored. Depending on the excitation level, the vehicle may exhibit dramatic maneuvering behaviors.

Fig. 5.11 shows an example of the results obtained from a system identification mission conducted in the reservoir. Performing system identification within the C2

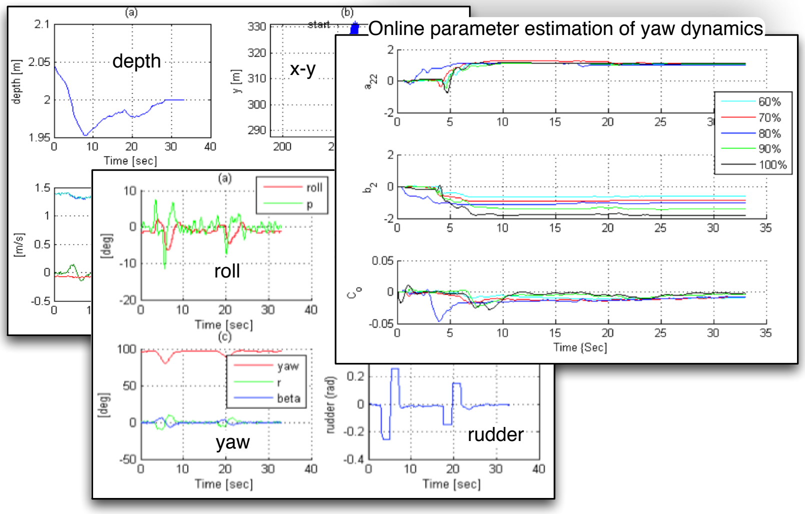


FIGURE 5.11: Example of an experimental mission for the identification of yaw dynamics using STARFISH AUV. Plot of depth, x-y position, roll, yaw, rudder angles and estimated parameters of the vehicle's yaw dynamics. Images quoted from [88].

system's mission setting, as compared to manual launching of vehicle from a platform or vessel, has some operational advantages. First, for the safety of the vehicle, a mission point can be planned to navigate the vehicle away from any man-made structure before the process begins, while an end point for the vehicle to return after the process to ease the vehicle's retrieval. Second, the Safety Officer and the Health Monitor agents continue to ensure the vehicle's safety even during the identification mission. Thus, the mission can be aborted if the vehicle is driven out of geo-fence or in any form of danger, due to `BD_SysIden`'s commands.

Unlike other mission-based C2 systems like MOOS-IvP [52] and Neptus [89], where the system identification has to be performed manually and separately, at the low-level vehicle level, the proposed C2 system incorporates the functionality as one of the high-level mission objectives and allows it to be performed automatically within the setting of an autonomous mission.

5.4.2 Surveying Mission

AUVs are commonly deployed for surveying missions due to their capabilities in carrying various sensors onboard and to navigate to places where it is dangerous for human

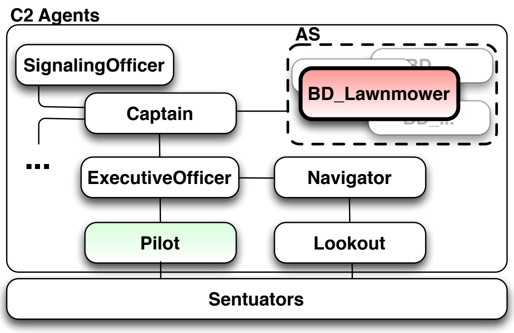
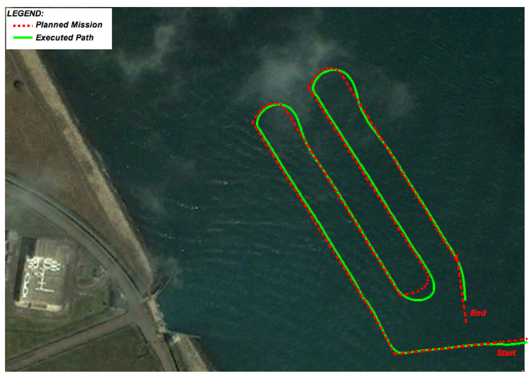
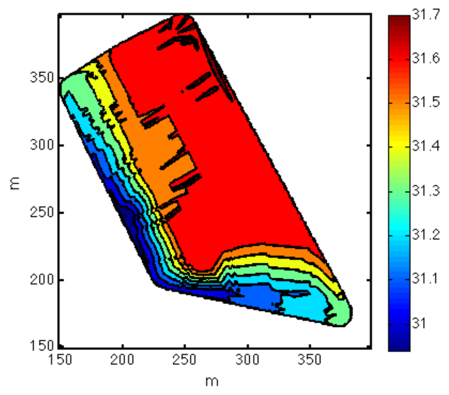


FIGURE 5.12: C2 agents and the `BD_Lawnmower` agent for coastal thermal field survey mission.



(a)



(b)

FIGURE 5.13: (a) Planned (red dotted line) and executed (green solid line) survey paths by the STARFISH AUV during the Tuas, Singapore field experiment in August 2012. (b) Thermal field sensed by the AUV.

surveyors. In this section, we showcase the extensibility of the C2 system in coping with different mission requirements imposed by different payload modules in two separate surveying missions.

5.4.2.1 Coastal Thermal Field Survey

The first mission was carried out at Tuas, Singapore in August 2012, where the C2 system (Fig. 5.12) was asked to carry out a surveying mission at the outlet of the power generator's cooling tower. The mission reused the `BD_Lawnmower` agent mentioned in

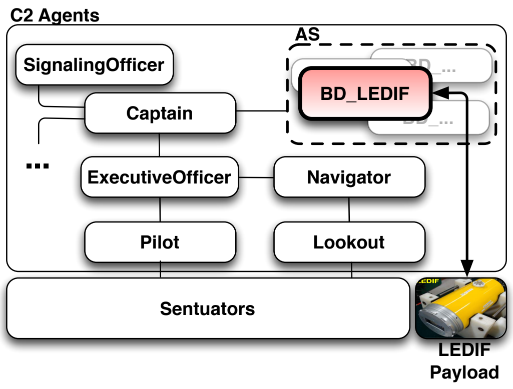


FIGURE 5.14: C2 agents, the BD_LEDIF and the LEDIF payload for chlorophyll survey mission.

section 5.3 to generate the mission points in a lawn-mowing pattern within an area specified by the operator. The mission points were sent to the Mission level for waypoints planning and then to Vehicle level for execution.

The resultant path from the surveying mission is shown in Fig. 5.13. Varying water flow directions were observed in the mission area due to the interaction of the south-east tidal current and the outlet's water flow. Although this presents a challenge for the low level vehicle navigational control, the achieved path has little overshot using the path-following algorithm implemented in the Pilot agent. The decoupling of high-level mission point planning and low-level vehicle control in separate agents made the tasks more manageable. If desired, a new Pilot agent with different path-following algorithms can be constructed to replace the current Pilot agent without interfering the rest of the control system.

5.4.2.2 Chlorophyll Survey

The second mission was carried out at Pandan Reservoir, Singapore, where the C2 system was instructed to perform three-dimensional subsurface mapping of Chlorophyll concentration. In this experiment, the STARFISH AUV was fitted with the LEDIF [90] payload section (Fig.5.14). The LEDIF payload contains a suit of sensors that are capable of measuring the fluorescence, absorbance, and turbidity of natural water. Without any modification to the overall control architecture, the proposed C2 system was

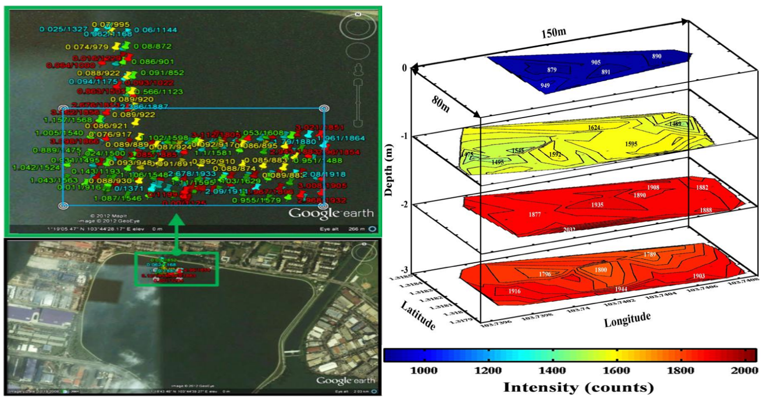


FIGURE 5.15: Results of Chlorophyll survey in the reservoir [90].

able to accommodate these new sensing capabilities with an addition of a Scientist agent, namely the BD_LEDIF agent. At present, the operation of the BD_LEDIF is pre-programmed in the mission file during mission planning stage, where the BD_LEDIF agent can be instructed to turned on or off the LEDIF sensors via the "Payload" Tab shown in the Fig. 5.5-(e). However, one could easily extend the capability of the BD_LEDIF agent for an adaptive sampling mission, in which mission points can be generated in real time according to the data collected by the sensors.

Fig. 5.15 shows examples of the resultant trajectories executed by the C2 system on the left, and the corresponding Chlorophyll concentration mapping on the right. Through this simple experiment, we illustrated the extendability of the C2 system in accommodating the new mission requirements introduced by a sensor payload.

5.4.3 Adaptive Mission

One of the advantages of adopting the back-seat driver paradigm described in Section 5.1.2 is the C2 system's ability in carrying out an adaptive mission. BD agents with different adaptive algorithms can be developed to generate mission points in real time during mission execution, and provide feedback to the Captain for execution. In this section, we showcase how the C2 system can be easily extended to carry out adaptive missions with different BD agents.

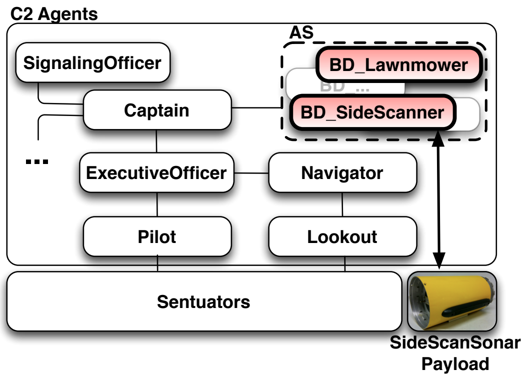


FIGURE 5.16: C2 agents, the `BD_SideScanner` and the `BD_Lawnmower` agents for target revisit mission.

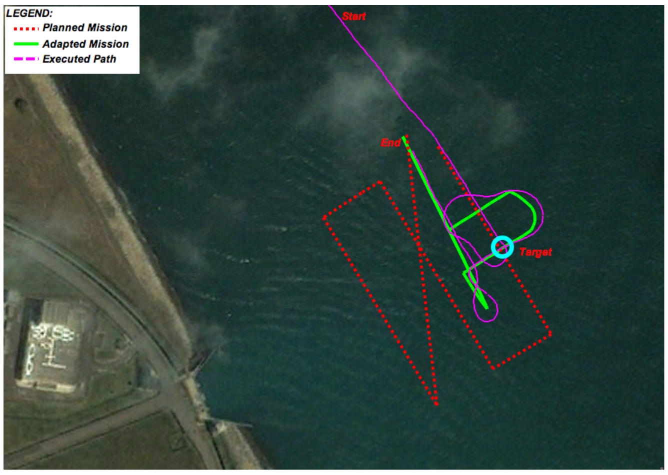


FIGURE 5.17: AUV path during the adaptive mission at Tuas, Singapore field experiment in August 2012. Red dotted line shows the initial planned survey path. When a simulated target is detected, the mission was interrupted and re-planned (green solid line) to re-visit the target before proceeding to the end point.

5.4.3.1 Target Revisit

Fig. 5.17 shows another survey mission carried out by the C2 system at Tuas, Singapore. The mission goal was to simulate a target of interest being detected during the execution of a preplanned lawn-mowing survey mission (red dotted line), and to verify the capability of the C2 system in re-adapting the current mission to re-visit the target with a second fly-by path.

A Scientist agent (`BD_SideScanner`) was developed and introduced into the Agent Service to simulate continuous monitoring of the sensor data returned by the Side Scan Sonar payload module throughout the surveying mission (Fig. 5.16). Whenever a target of interest is detected, it broadcasts a request to the BD agents within the AS for new mission points that will navigate the AUV to re-visit the target position. Once received, the `BD_Lawnmower` replies with a set of mission points which results in a second fly-by over the target position. The execution of the current surveying mission is interrupted and replaced with the newly planned mission points. The AUV then navigates through the target for the second time before completing the mission at the end point.

The mission scenario demonstrated the underlying C2 system's capability in handling new payloads, and its flexibility in incorporating adaptive behaviors during an autonomous mission. More importantly, the concept of Agent Service allows the developer to design different mission adaptivities at the C2 system's Supervisory level, while both the Mission and Vehicle level's agents remain untouched.

5.4.3.2 Cooperative Positioning

In another field experiment conducted at Pandan Reservoir, Singapore, we carried out a multi-vehicle cooperative mission similar the one mentioned in 5.3. We implemented the cooperative positioning algorithm mentioned in Chapter 3, and deployed it into two STARFISH AUVs, as shown in Fig. 5.19(a), with one being the Beacon vehicle and another one being the Survey AUV. Detailed result analysis were presented in Section 3.6.1. The objective of this section is to showcase the robustness of the C2 system in multi-vehicle cooperative mission. The C2 system in both the AUVs were identical, with the exception of a `BD_Lawnmower` agent in the Survey vehicle, and a `BD_Coop` in the Beacon vehicle (Fig. 5.18). Since the water depth in the reservoir is too shallow, a geo-fence was defined for `BD_Coop` so that the cooperative navigation could be executed within a confined area, to reduce the risk of vehicle collision.

During the mission execution, the survey AUV broadcast its position acoustically through the UnetStack. Upon receipt by the UnetStack of the beacon vehicle, the position information is forwarded to the Signaling Officer, who in turn broadcasts it to the `BD_Coop` agent as TeamInfo messages. The beacon vehicle's path is planned adaptively based on the survey AUV's position information. Fig. 5.19(b) shows an example

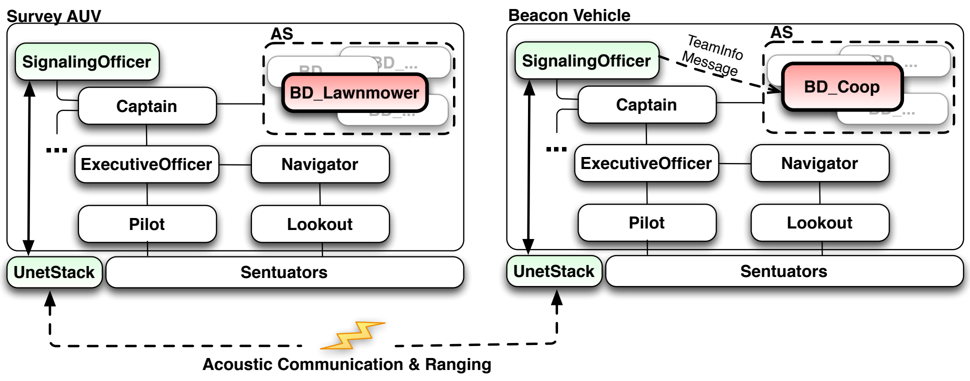
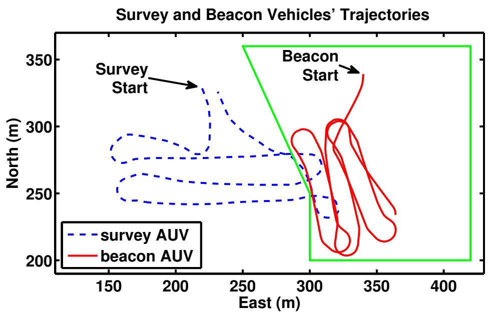


FIGURE 5.18: C2 agents, the `BD_Lawnmower` agent for the survey AUV and the `BD_Coop` agent for the beacon vehicle of the cooperative positioning mission. Inter-vehicle communication and ranging are facilitated by the UnetStack and acoustic modems.

(a)



(b)

FIGURE 5.19: (a) Two STARFISH AUVs used for the cooperative navigation mission at Pandan Reservoir, Singapore. (b) The trajectories executed by the beacon AUV within the geo-fence (green box).

of the trajectories executed by the Survey vehicle (dotted blue line) and the Beacon vehicle (red line) during the multi-vehicle cooperative mission.

Through this mission scenario, we demonstrated the robustness of the C2 system to be deployed for multi-vehicle cooperative missions. The adoption of backseat driver paradigm at the Supervisory level of the control hierarchy allows algorithm developers to focus their efforts on designing the high-level, multi-vehicle mission behaviors, while reusing the same low-level control architecture for the individual vehicles.

5.4.3.3 Cooperative Source Localization

To further illustrate the portability of the C2 system, we present a mission scenario where the C2 system is deployed onto a team of miniature marine vehicles called the SWARMBOT (Fig. 5.21(a)), for a cooperative source localization mission. The experiment aims to demonstrate the emergent behavior which arose from the implicit communication among a team of robotic systems. Fig. 5.21(b) shows a snapshot of the resultant trajectories executed by a team of four SWARMBOTs in localizing a simulated target source within the success zone, using only implicit communication among the team members [91].

Rather than developing a new C2 system for the control of the SWARMBOTs, we simply deploy the existing C2 system with only a minor modification to the overall control architecture. Since the SWARMBOT is equipped with different sensors and actuators, compared to the STARFISH AUV, a new Sentuator module was developed to replace the existing STARFISH Sentuator module. Again, the desired cooperative behavior for the vehicle team can be incorporated within a new `BD_BIOCAST` agent, without any modification to the existing pool of the BD agents or the control architecture. The final C2 system configuration for the SWARMBOT experiment is shown in Fig. 5.20. All the vehicles were running identical copies of the C2 system, and their collective interaction via implicit communication helps the team to achieve the mission objective.

5.5 Discussion

Through various mission scenarios, we have showcased the robustness, extensibility and portability of the proposed C2 system. The design of the C2 system promotes “Separation of Concerns” as it clearly allocates different control responsibilities of an autonomous vehicle into different self-contained software agents. Besides, the agents are distributed over different levels of control hierarchies to further define their tasks and response requirements. The hierarchical multi-agent based design also provides an explicit view of control flow and message passing among the agents. This is important in

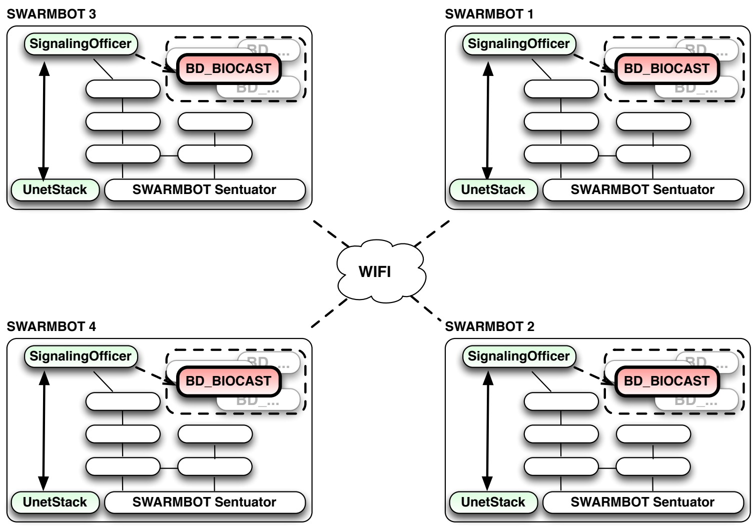
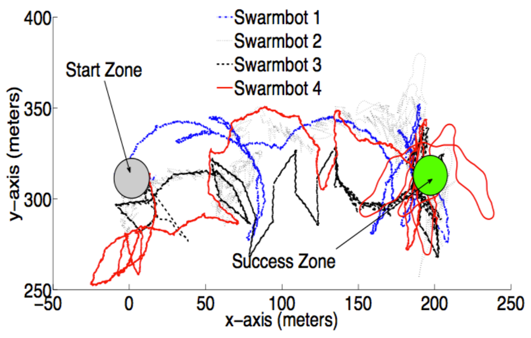


FIGURE 5.20: Identical copies of the C2 system deployed on different SWARMBOT vehicles. Besides introducing the new BD_BIOCAST agent into the AS, the only modification to the existing C2 system is the new SWARMBOT Sentuator component for interfacing with the vehicle's sensors and actuators.

(a)



(b)

FIGURE 5.21: (a) The SWARMBOT vehicles deployed for the collective localization mission. (b) A snapshot of the resultant trajectories planned and executed by the C2 systems deployed onboard four SWARMBOTs [91].

terms of system maintainability as developers could identify specific sub-components of the C2 system to be designed, developed and debugged in isolation without interfering with the overall system's integrity. Furthermore, the modular design that decouples the C2 system from the vehicle's Sentuator components also allows it to be ported into other mission-based robotic systems. This can be achieved simply by replacing the existing Sentuator component with one that interacts the targeted hardware system, as demonstrated in cooperative source localization mission scenario in Section 5.4.3.3.

The adoption of the back-seat driver paradigm also allows the C2 system's mission capabilities to be extended easily to cope with new mission requirements or payload modules. This can be done via the introduction of BD agents that implement the desired mission behaviors, as illustrated by numerous scenarios in the previous section. The interaction between the BD agents with the Captain in the form of mission points provides an intuitive notion for designing mission behaviors, compared to computing the IvP function in the MOOS-IvP [52] or designing the mission plan using declarative language like the New Domain Description Language (NDDL) in the TREX [92]. Apart from deterministic behaviors, adaptive behaviors involving single or multiple vehicles can be easily implemented in a BD agent by providing new mission points to the Captain during the course of a mission, replacing the existing mission points. As illustrated by the adaptive mission scenarios (section 5.4.3), the ability to modify a mission plan adaptively online is crucial since the C2 system has the flexibility to replan mission points that would better achieve the mission objectives, based on the latest sensor measurements.

5.6 Summary

In this chapter, we described a hierarchical multi-agent C2 system and its implementation on the STARFISH AUVs in detail. The system architecture mimics the control structure of a submarine, where mission and vehicle tasks are clearly divided and handled by individual agents organized at different levels in a control hierarchy. The adoption of the backseat driver paradigm at the control system's Supervisory level allows new mission behaviors and capabilities to be introduced into the C2 system with minimum modification to the overall architecture.

Although the C2 system was developed with the marine vehicles in mind and implemented using an in-house agent platform, the architecture may also be used in land or aerial vehicles, and implemented using other popular middleware. The agent-based software design utilizing message passing mechanisms for agent communication decouples the C2 system from the vehicle's Sentuator. This allows the C2 system to be validated in simulation by simply diverting agent messages to a vehicle simulator, and deployed directly into a physical robotic system when it is fully tested in simulation mode.

Through simulation and field experiments around the Singapore's costal water, we demonstrated the capabilities of the C2 system in carrying out single and multiple vehicle missions. Through different mission scenarios, we showed the robustness and extendability of the C2 system in coping with new mission requirements.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

Underwater navigation is a challenging problem for low-cost sensor-limited AUVs. Commercially available underwater positioning systems offer a solution, but require considerable operational effort and are generally costly. The accuracy of AUVs' navigation is crucial in determining the quality of collected data as well as ensuring the vehicle safety.

The availability of low-cost modular AUVs, with different payload configurations have driven researchers' desire for autonomy in a team of vehicles. Multi-vehicle missions offer several advantages over single-vehicle missions in terms of flexibility and tolerance towards a single vehicle failure. Besides, the ability in inter-vehicle communication using acoustic modems heralds a new age of team-based cooperation for underwater vehicles.

In this thesis, we developed a cooperative path planning algorithm for a moving beacon that takes into account inter-vehicle geometries and the position errors accumulated by a team of AUVs. The algorithm allows the AUVs to estimate their positions using an acoustic signal broadcast by the beacon. The path planning problem was formulated within a MDP, with its policy "learned" using two optimization techniques: the Cross-entropy methods and the Variable-length Genetic Algorithm. The algorithm was validated through a series of simulations and field experiments, and the performance was compared against other position estimation techniques. Results showed that the proposed algorithm kept the position errors of the AUVs small, and robust against some degrees of communication and environmental uncertainties.

Given the bathymetry map of a mission area, a team of sensor-limited AUVs equipped only with a single-beam altimeter, are able to perform cooperative localization. We developed a decentralized position estimation for the AUVs based on particle filtering technique. The localization of the individual vehicle depends solely on their water depth measurements and the information obtained from inter-vehicle acoustic communication. Given sufficient terrain information, the proposed localization techniques avoid the need of a beacon signal or expensive sensors for underwater navigation, while still being capable of keeping the position errors of the AUVs relatively small. Simulation studies showed the filter's performance is comparable to that of a centralized filter, with the benefit of much lower communication bandwidth. Offline filtering with field experiment data demonstrated the feasibility of the cooperative localization technique and showed its robustness in handling communication uncertainties and sensor bias.

Finally, a hierarchical agent-based C2 system was developed for a modular AUV. In contrast with other mission controllers, mission, navigation and vehicle tasks were allocated to individual self-contained software agents, each with their own responsibilities and behaviors. The collective interactions among the agents enable the AUV to achieve its mission objective autonomously. The adoption of backseat driver paradigm at the mission level of the control architecture allows for easy expansion of the C2 system's capabilities in handling with new mission requirements. A carefully designed GUI was developed to allow for intuitive multi-vehicle mission planning, while an agent-based AUV simulator was also implemented for offline testing of agent behaviors before field deployment. Numerous field experiments around the Singapore water, with the AUV fitted with different sensor payloads for different mission objectives, showcased the robustness and extendability of the developed C2 system.

6.2 Future Research

The cooperative algorithms and results presented have significantly expanded the applicability of multi-vehicle missions using low-cost sensor-limited AUVs. However, there are several natural extensions that can be pursued to improve their applications in the AUVs:

- The path planning policies for beacon vehicles presented in Section 3.4.2 were "learned" offline, with large number of Monte Carlo simulations. However, it may not capture all the uncertainties that a field experiment may present. On the other hand, learning the policies afresh online may not be the solution either, as frequently visited states may dominate the learning process. One way to overcome this problem is to initially deploy the policy that was learned offline, and allow the candidate policies to continue to evolve using the information captured from field experiments. The initial policy can be replaced whenever its fitness score is lower than any of the candidates.
- Whenever a beacon vehicle is deployed to support multiple AUVs, different strategies can be applied to choose the desired heading command for the beacon vehicle. The decision making presented was ad-hoc at best. An interesting extension would be to identify the strategy deciding factors and apply heuristic techniques to learn the optimum mapping using data gathered from field experiments.
- The relative geometries among the vehicles and the amount of terrain information within the vehicles' mission area determine the performance of the cooperative localization. Since the bathymetry map is given, and the acoustic communication among the vehicles can be scheduled before mission execution, multi-vehicle motion planning algorithms can be designed with desired performance bound on the vehicle localization. The algorithm would plan the vehicles' waypoints per communication slot, taking into account the terrain information attainable by respective vehicles, such that the collective localization of the vehicle keeps their position uncertainties bounded. This approach has the potential to be applied for long-term underwater navigation using a team of sensor-limited AUVs.

- In the C2 system, we adopted a simple priority-based look-up table for contracting a BD agent to generate mission points if there are multiple competing BD agents replying to the Captain's Call-for-participation (see Section 5.1.2). To achieve maximum benefit of the backseat driver paradigm, more sophisticated arbitration rules have to be defined. Besides, automated planning tools can also be integrated in the Captain to increase the level of autonomy in its interaction with the AS.

Appendix A

Error Estimate Covariance Due to Range Updates

Let the state vector $\mathbf{x}$ represents the vehicle's position in 2-D space, that is, $\mathbf{x}_l = [x, y]_l^\top$. At time $l+1$, the beacon vehicle's position is $\mathbf{x}_{l+1}^B = [x^B, y^B]_{l+1}^\top$. Let β_{l+1} be the angle formed by the line joining the beacon vehicle and survey AUV, $\beta_{l+1} = \angle(\mathbf{x}_{l+1}^B - \mathbf{x}_{l+1})$, the observation matrix with respect to the true position is

$$\boldsymbol{H}_{l+1} = \begin{bmatrix} \cos\beta_{l+1} & \sin\beta_{l+1} \end{bmatrix}. \tag{A.1}$$

Assuming that the position error of the survey AUV can be described as an error ellipse, the error estimate covariance $\boldsymbol{P}_l$ can be written as:

$$\begin{aligned} \boldsymbol{\Upsilon}_l \boldsymbol{P}_l' \boldsymbol{\Upsilon}_l^\top &= \boldsymbol{P}_l \\ \boldsymbol{\Upsilon}_l \begin{bmatrix} \varepsilon_l^2 & 0 \\ 0 & \bar{\varepsilon}_l^2 \end{bmatrix} \boldsymbol{\Upsilon}_l^\top &= \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}_l \end{aligned} \tag{A.2}$$

where rotation matrix $\boldsymbol{\Upsilon}_l$ is formed by the angle of the minor axis θ_l, counterclockwise from x axis. $\sigma_{x,l}'^2 = \varepsilon_l^2$ and $\sigma_{y,l}'^2 = \bar{\varepsilon}_l^2$ denote the length of the minor and major axis at time step l after propagation.

Let the measurement error $\sim N(0,\sigma_z^2)$, which includes the ranging error $\sim N(0,\sigma_R^2)$ and position error of the beacon vehicle $\sim N(0,\sigma_B^2)$. We have $\sigma_z^2 = \sigma_R^2 + \sigma_B^2$. The innovation covariance is then derived as:

$$\begin{aligned}\boldsymbol{S}_{l+1} &= \boldsymbol{H}_{l+1}\boldsymbol{P}_l\boldsymbol{H}_{l+1}^\top + \sigma_z^2 \\ &= \boldsymbol{H}_{l+1}\boldsymbol{\Upsilon}_l\boldsymbol{P}_l'\boldsymbol{\Upsilon}^\top\boldsymbol{H}_{l+1}^\top + \sigma_z^2 \\ &= \varepsilon_l^2\cos^2(\beta_{l+1}-\theta_l) + \bar{\varepsilon}_l^2\sin^2(\beta_{l+1}-\theta_l) + \sigma_z^2.\end{aligned} \tag{A.3}$$

The Kalman gain is

$$\boldsymbol{K}_{l+1} = \boldsymbol{P}_l\boldsymbol{H}_{l+1}^\top\boldsymbol{S}_{l+1}^{-1} \tag{A.4}$$

and the error estimate covariance is updated as

$$\boldsymbol{P}_{l+1} = \boldsymbol{P}_l - \boldsymbol{K}_{l+1}\boldsymbol{H}_{l+1}\boldsymbol{P}_l. \tag{A.5}$$

$\boldsymbol{P}_{l+1}$ is a 3×3 symmetric matrix with the components in the upper triangle as:

$$\begin{aligned}\boldsymbol{P}_{11,l+1}\boldsymbol{S}_{l+1} &= \varepsilon_l^2\bar{\varepsilon}_l^2\sin^2\beta_{l+1} + (\varepsilon_l^2\cos^2\theta_l + \bar{\varepsilon}_l^2\sin^2\theta_l)\sigma_z^2 \\ \boldsymbol{P}_{12,l+1}\boldsymbol{S}_{l+1} &= -\frac{1}{2}\varepsilon_l^2\bar{\varepsilon}_l^2\sin(2\beta_{l+1}) + \frac{1}{2}(\varepsilon_l^2-\bar{\varepsilon}_l^2)\sigma_z^2\sin^2(2\theta_l) \\ \boldsymbol{P}_{22,l+1}\boldsymbol{S}_{l+1} &= \varepsilon_l^2\bar{\varepsilon}_l^2\cos^2\beta_{l+1} + (\varepsilon_l^2\sin^2\theta_l + \bar{\varepsilon}_l^2\cos^2\theta_l)\sigma_z^2.\end{aligned} \tag{A.6}$$

The angle of the minor axis by $\mathbf{P}_{l+1}$ is θ_{l+1}. Thus,

$$\tan(2\theta_{l+1}) = \frac{-\varepsilon_l^2\bar{\varepsilon}_l^2\sin(2\beta_{l+1}) + \sin(2\theta_l)(\varepsilon_l^2-\bar{\varepsilon}_l^2)\sigma_z^2}{-\varepsilon_l^2\bar{\varepsilon}_l^2\cos(2\beta_{l+1}) + \cos(2\theta_l)(\varepsilon_l^2-\bar{\varepsilon}_l^2)\sigma_z^2}. \tag{A.7}$$

With the assumption that $\sigma_z^2 \ll \varepsilon_l^2 \leq \bar{\varepsilon}_l^2$, we have $\frac{\sigma_z^2}{\varepsilon_l^2\bar{\varepsilon}_l^2} \to 0$,

$$\begin{aligned}\lim_{\frac{\sigma_z^2}{\varepsilon_l^2\bar{\varepsilon}_l^2}\to 0} \tan(2\theta_{l+1}) &= \tan(2\beta_{l+1}) \\ \lim_{\frac{\sigma_z^2}{\varepsilon_l^2\bar{\varepsilon}_l^2}\to 0} \theta_{l+1} &= \beta_{l+1}.\end{aligned} \tag{A.8}$$

With $\gamma_l = \theta_{l+1} - \theta_l$, the minor and major axis of the error ellipse formed by $\boldsymbol{P}_{l+1}$ are:

$$\begin{aligned} \lim_{\frac{\sigma_z^2}{\varepsilon_l^2 \bar{\varepsilon}_l^2} \to 0} \sigma'^2_{x,l+1} &= \sigma_z^2 \\ \lim_{\frac{\sigma_z^2}{\varepsilon_l^2 \bar{\varepsilon}_l^2} \to 0} \sigma'^2_{y,l+1} &= \frac{\varepsilon_l^2 \bar{\varepsilon}_l^2}{\varepsilon_l^2 \cos^2 \gamma_l + \bar{\varepsilon}_l^2 \sin^2 \gamma_l}. \end{aligned} \tag{A.9}$$

For a Kalman filter with an identity propagation matrix $\boldsymbol{F}$:

$$\begin{aligned} \boldsymbol{P}_{l+1|l} &= \boldsymbol{F} \boldsymbol{P}_{l|l} \boldsymbol{F}^\top + \boldsymbol{Q} \\ &= \boldsymbol{P}_{l|l} + \boldsymbol{Q} \end{aligned} \tag{A.10}$$

where $\boldsymbol{Q} = \mathrm{diag}(q, q)$ which includes the error growth $q = \alpha\tau$. The updated error estimate covariance forms an ellipse with:

- The direction of the minor axis (minimum error) θ_{l+1} is along the line joining the beacon and survey AUV.
- The error in the minor axis has $\varepsilon_{l+1}^2 = \sigma_R^2 + \sigma_B^2 + \alpha\tau$.
- The error in the major axis has $\bar{\varepsilon}_{l+1}^2 = \frac{\varepsilon_l^2 \bar{\varepsilon}_l^2}{\varepsilon_l^2 \cos^2 \gamma_l + \bar{\varepsilon}_l^2 \sin^2 \gamma_l} + \alpha\tau$.

Appendix B

Command and Control System Software Specifications

B.1 C2 Services

B.1.1 Captain Service

The Captain starts, coordinates and controls the execution of a mission. Whenever an Operator's command is received, it broadcasts mission planning requests to the Agent Services and assign the mission point generator. It also handles the different types of mission point generation by the Agent Service.

Requests and Responses

Request	Response	Description
OperatorCmdReq	AGREE	Request to execute operator's command
MissionPtOWReqExe	AGREE, REFUSE	Request to overwrite the current mission point
MissionPtHPReqExe	AGREE, REFUSE	Request to insert the mission point right after the current mission point
MissionPtReqExe	AGREE, REFUSE	Request to append the mission point at the end of all available mission points
PosReq	PosRes	Request position information where mission started

Topics and Notifications

Notification	Topic	Description
C2EventNtf	C2EVENT	Notification for C2 events
C2CommandNft	ABORTSIGNAL	Notification for abort signal commanded by the Captain
MissionStatusNtf	MISSIONSTATUS	Notification for the latest mission status

B.1.2 SignalingOfficer Service

The SignalingOfficer acts as the AUV's external communication node. It decodes the mission commands received from the Operator and passes it to the Captain. It also handles the requests from the GUI and notifies the GUI/Operator of critical C2 events.

Requests and Responses

Request	Response	Description
OperatorCmdReq	AGREE	Request to execute operator's command
LogfilenameReq	AGREE	Request for the name of current logfile
C2EventReq	DatagramReq	Request for C2Event by Operator's GUI

B.1.3 ExecutiveOfficer Service

The ExecutiveOfficer receives mission tasks in the form of mission points and passes them to the Navigator for waypoints planning. Once waypoints are received, sends them to the Pilot for execution. The mission and waypoints are received and passed asynchronously to the Navigator and the Pilot to ensure the smoothness of mission execution. For more details, please refer to Appendix C.

Requests and Responses

Request	Response	Description
C2CommandReq	AGREE	Request to execute command from Captain
MissionPtReqExe	AGREE, REFUSE	Request to execute mission point

B.1.4 Navigator Service

The Navigator implements path planning algorithms to translate mission points to waypoints. Future extension will include collision detection and re-planning whenever an obstacle is detected by the Lookout.

Requests and Responses

Request	**Response**	**Description**
C2CommandReq	AGREE	Request to execute command from Captain
WayPtReq	WayPtRes	Request to translate mission point to way points

B.1.5 Pilot Service

The Pilot translates the waypoints into the vehicle control commands. During a mission execution, it also broadcast the waypoints execution status periodically.

Requests and Responses

Request	**Response**	**Description**
C2CommandReq	AGREE	Request to execute command from Captain
WayPtReqExe	AGREE, REFUSE	Request to execute way points
StateReportReq	StateReport	Request Pilot's current state

Topics and Notifications

Notification	**Topic**	**Description**
MissionPointStatusNtf	MISSIONPOSITIONSTATUS	Notification for status of the current mission point execution

B.1.6 EngineRoom Service

The EngineRoom handles all the vehicle commands and passes them to the vehicle's actuators. Besides, it also pools the vehicle's sensor status periodically and broadcast them to all the agents in the C2 system.

Requests and Responses

Request	Response	Description
DivingOfficerReq	AGREE, REFUSE	Request to dive to certain depth
HelmsmanReq	AGREE, REFUSE	Request to steer to certain bearing
EngineRoomReq	AGREE, REFUSE	Request to thrust at certain thruster level
VehicleStatusReq	AGREE	Request to report current engine room's status
OperatorCmdReq	AGREE, REFUSE	Request to enable/disable thruster

Topics and Notifications

Notification	Topic	Description
C2EventNtf	C2EVENT	Notification for C2 events reported from vehicle level
VehicleStatusNtf	VEHICLESTATUS	Notification of latest status of the vehicle

B.1.7 Backseat Driver Service

Depending on the algorithms implemented, the BackseatDriver generates mission points and passes them to the Captain for execution. If necessary, the BackseatDriver can interrupt the Captain for mission point modification, addition or cancellation.

Requests and Responses

Request	Response	Description
CFPReq	CFPRes	Request to Call-For-Participation for mission leg
C2CommandReq	AGREE	Request to execute command from Captain
MissionPtReq	MissionPtRes	Request to generate mission point, if this is the contracted agent

Topics and Notifications

Notification	Topic	Description
C2EventNtf	C2EVENT	Notification for C2 events
C2CommandNft	ABORTSIGNAL	Notification for abort signal commanded by the Captain
MissionStatusNtf	MISSIONSTATUS	Notification for the latest mission status

Appendix C

Java Command and Control System Developer Guide

Contents

1 Overall Concept

1.1 Agent-based Command and Control (C2) System

The C2 system is developed based on the paper [1]. The overall framework is shown in Fig 1.

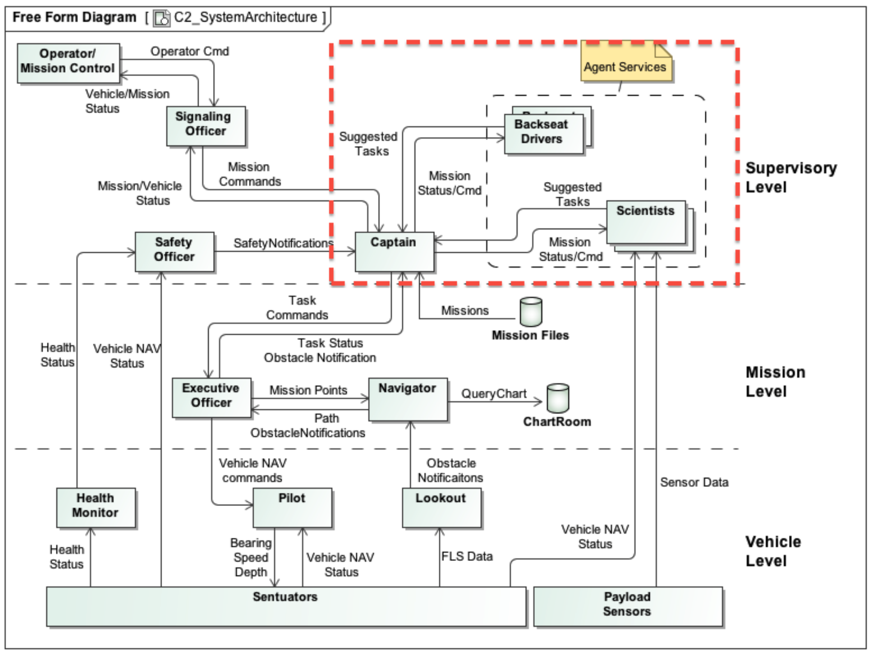


Figure 1: Heirarchical Command and Control System.

This document aim to serve a few purposes:

1. Describe the interaction between the Captain and the BackSeat Driver (BD) agents.
2. Explain the development of third party BD agents (programmatically) and its compilation steps against the provided Java C2 (JC2) framework.
3. Performing simulations using the StarControl and some simple data analysis of the result (like the vehicle's planned and executed paths).

1.2 StarFish Missions and Mission Tasks

A StarFish mission consists of a sequence of mission tasks (`MissionTask`) arranged in sequential order. During mission execution, each mission leg is executed in the same order. Each mission leg consists of 3 main components : (1) the Mission position (`MissionPosition`) that associate with this mission task (More about this association later), (2) the payload (`Payload`) commands (Optional) and (3) the user-defined properties. Every mission task comes with an unique ID (`taskID`) which is automatically generated when the mission task class is initialized. This taskID will be broadcasted to all the agents in the framework so that the issuing agent will be notified when the task is being executed. It is up to the issuing agent to decide on the activities/algorithms to perform during the task execution.

1.3 Mission Execution

The mission execution (between the mission point generation and execution) is asynchronous. This means the mission point generated by the BDAgent may not be executed immediately after being received by the Captain. There is a mission point queue maintained by the Captain. Whenever the number of mission points are less than 2, the Captain will start requesting for more mission points from the contracted BDAgent. Fig. 2 further illustrated this.

This is to ensure the smoothness of mission execution so that the vehicle does not have to stop while waiting for new mission points to be generated.

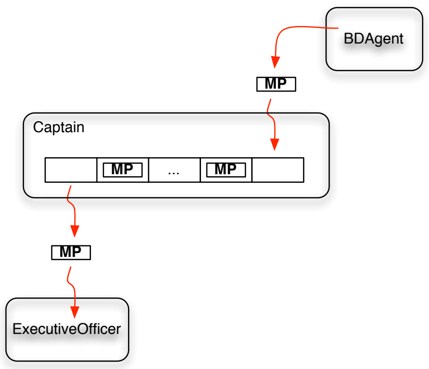


Figure 2: Asynchronous Mission Point Generation and Execution.

2 BackSeat Driver Agent

Depending on the mission requirements, the Agent Service can consist as many agent as desired. Each BD agent can response and handle to one or many Mission Tasks depending on the implemented algorithms. Fig. 3 shows the agent services pool and the Captain agent. The Scientist BDAgents is just our conceptual categorization of the agents in the service pool. A BDAgent that interacts and controls the physical payload module while generating mission points is called the Scientist, while a BDAgent who merely generates mission points depending on the vehicle status, mission status and its implemented algorithm is simply called the BDAgent.

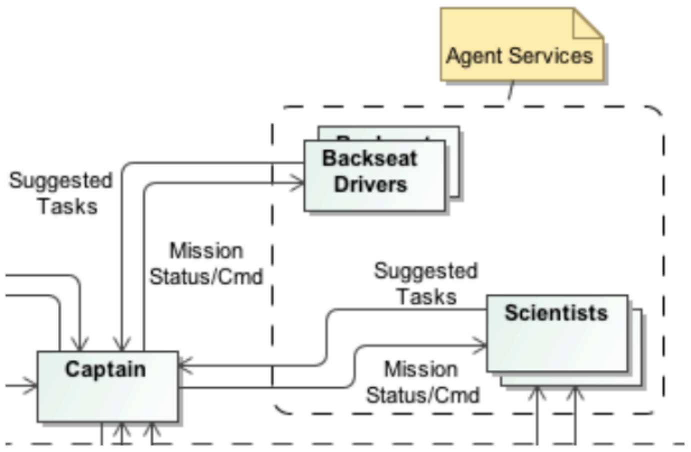


Figure 3: Agent Services and Captain

2.1 Captain and BD Agent Interaction

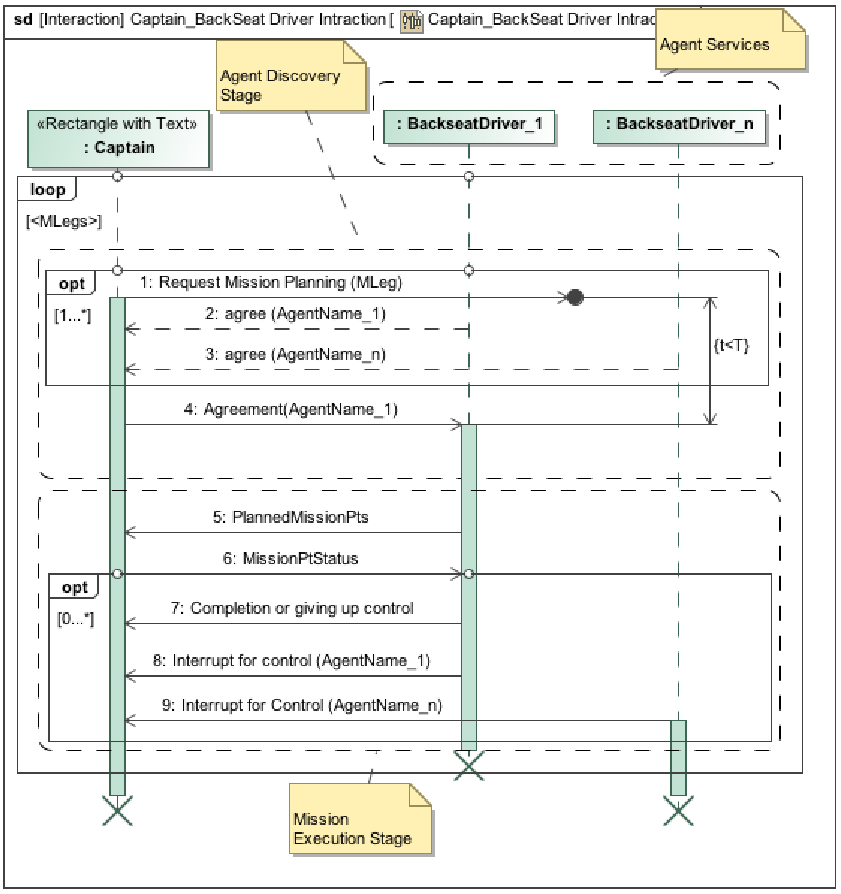

sd [Interaction] Captain_BackSeat Driver Intraction [Captain_BackSeat Driver Intrac
Agent Services
Agent Discovery Stage
«Rectangle with Text» : Captain
: BackseatDriver_1
: BackseatDriver_n
loop
[<MLegs>]
opt
[1...*]
1: Request Mission Planning (MLeg)
2: agree (AgentName_1)
3: agree (AgentName_n)
{t<T}
4: Agreement(AgentName_1)
5: PlannedMissionPts
6: MissionPtStatus
opt
[0...*]
7: Completion or giving up control
8: Interrupt for control (AgentName_1)
9: Interrupt for Control (AgentName_n)
Mission Execution Stage

Figure 4: Interaction diagram between Agent Services and Captain

The interaction between the Captain and the BDAgents are divided into two stages:

2.1.1 Agent Discovery Stage

At this stage, the depending on the missions, the Captain broadcast a Call For Participation Request (`CFPReq`) message to all the agents in the agent service pool. The message contains the specific mission task for the particular mission leg. BDAgents who are programmed to response to this particular mission task should response with Call For Participation Response (`CFPRes`) message with their AgentID attached, and the option for the Captain to request for mission point immediately or until certain condition is fulfilled.

Upon receiving all the responses (a simple time out on waiting implemented in the Captain), The Captain will decide on which DBAgent to be contracted as the agent that will be responsible for generating mission points for this particular mission leg.

2.1.2 Mission Execution Stage

Once the Captain has selected the DBAgent as the contracted Agent, the mission execution stage can happen in two modes:

2.1.3 Passive Interaction

In this normal mode, the Captain will started sending `MissionPtReq` message to the contracted BDAgent. Upon receiving the message, the contracted BDAgent should response with `MissionPtRes` message with the generated Mission Point. An empty `MissionPtRes` with its `State` set to `COMPLETED` can be returned once the BDAgent has completed its tasks or given up the reponsibility.

2.1.4 Active Interaction

There may be scenarios when the BDAgent needs to generate the mission points adaptively and requires immediate execution of the mission points, the asynchronous mission point execution mentioned in the section 1.3 would not satisfy this requirement. In such cases, the BDAgent should not wait for the `MissionPtReq` message, instead, they should send the Captain either one of these two messages:

1. `MissionPtReqExe`
   Mission Point Request for Execution, this message request the Captain to insert extra mission point/points at the end of currently executing mission point queue. The execution of the newly submitted points will only be executed after all the mission points the queue have been executed.

2. `MissionPtHPReqExe`
   Mission Point Request for Execution (High Priority), this message request the Captain to stop the execution of the current mission point and execute the requested mission point. Once the requested mission point is completed, the current mission point execution will be resumed. Equivalent to push the HP mission Point to the front of the queue.

3. `MissionPtOWReqExe`
   Mission Point Request for Execution (OverWritten), this message request the Captain stop and delete the the current mission point and replaced it with the requested mission point. The mission point queue is cleared, this OW mission point is put into the empty queue.

After the execution of either the HP or OW, the execution always falls back to the passive mode, until the contracted BDAgent return `COMPLETED` message as mentioned before.

2.1.5 Examples

In the following examples, a simple mission is planned navigating the vehicle from the starting point through 3 mission points: MP1, MP2, MP3. As an example, another BDAgent is programmed to monitor the progress of the execution and interrupt when the vehicle reached the MP1. The interrupts were in the form of 1). Mission Point Request for Execution Fig. 5, 2). Mission Point Request for Execution - High Priority Fig. 6, and 3). Mission Point Request for Execution - OverWrite Fig. 7. This example showcased the 3 different behaviors of active interaction between the Captain and the BDAgents.

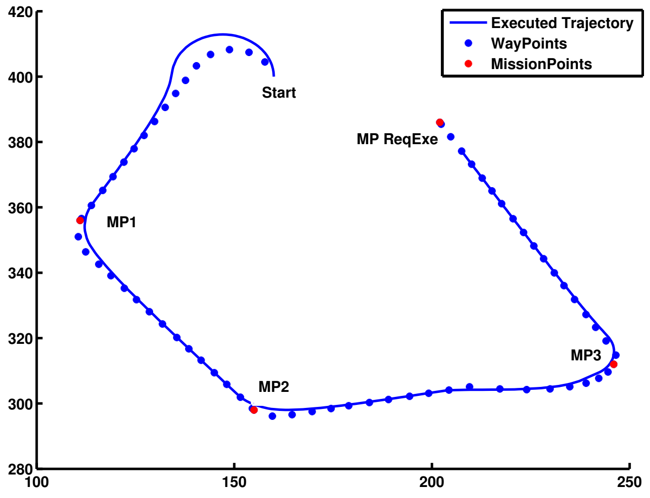


Figure 5: Mission Point Request for Execution

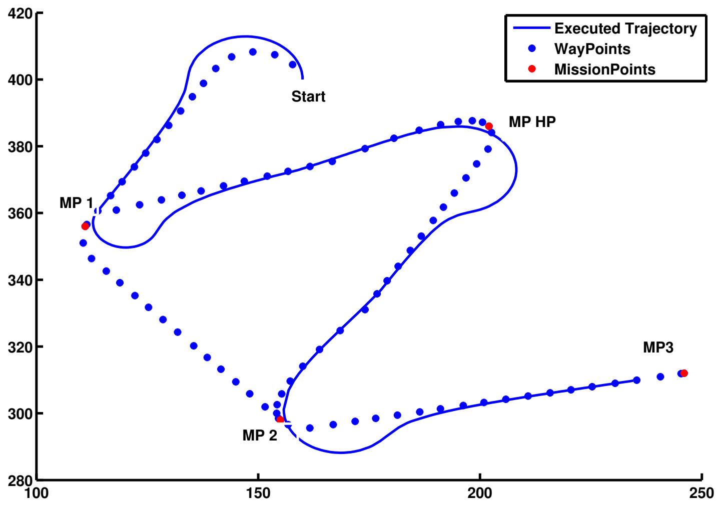


Figure 6: Mission Point Request for Execution - High Priority

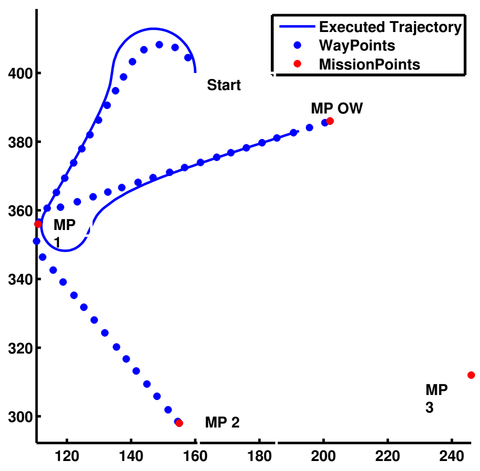


Figure 7: Mission Point Request for Execution - Overwrite

3 Programming Guide

A BD agent extends the base class Agent which belongs to the Framework for Java and Groovy Agents (fjåge). The developers are allowed to use all the facilities provided by the fjåge framework. Please refer to the online documentation fjåge for more details. However, in order to interact with the JC2 framework, the developers must extend the BDAgent class.

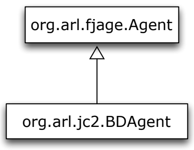


Figure 8: DBAgent extends the base class Agent from the `org.arl.fjage package`.

Depending on the desired mission behaviors, new BD Agents can be added to the C2 system to extend and handle different mission requirements. In this part of the documentation, a simple template project which build a Loiter_BDAgent on top of the JC2 framework will be used as the example to demonstrate programing a BD agent that generates mission position so that the vehicle will loiter around some given mission positions. The sample codes can be found in the attached folder called "JC2_aTemplate".

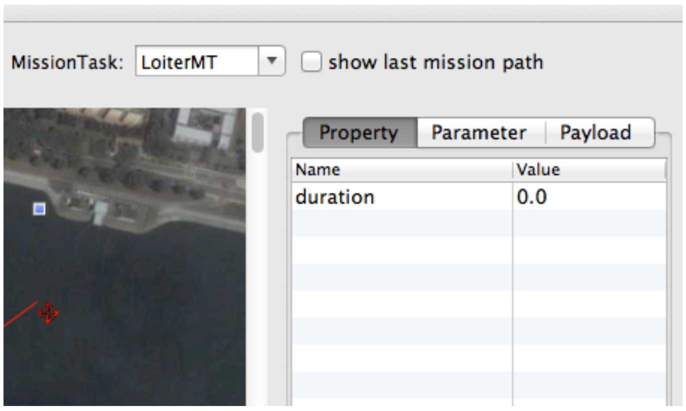
3.1 Mission Task

The base class - MissionTask provides all the housekeeping methods required to interact with the StaControl mission planning GUI as well as the JC2 for mission execution. What left is for the developers to define the properties for the mission task. These properties can be assigned with the desired values during mission planning, and be applied during mission execution. The code snippet shown in Listing 1 is a simple example showing the desired duration for the LoiterMT task, specifying the duration the vehicle should be loitering.

```groovy
package org.smart.censam.mtt

import org.arl.jc2.mtt.MissionTask;

class LoiterMT extends MissionTask {

  //declare the desired properties for this task.
  //Only primitive types are supported at the moment.
  float duration = 0

}
```

Listing 1: LoiterMT.groovy

During the mission planning, this property will be shown in the GUI's Property tab when a MissionTask has been selected and a mission position has been dropped on the map, as shown in Fig. 9. Currently, only the primitive types are supported !!



Name	Value
duration	0.0

Figure 9: LoiterMT in StarControl GUI showing its position in the mission map as well as the duration property defined in the source.

3.2 BDAgent

BDAgent.groovy is the base class of JC2 BD agent. All the third-party BD agent must extend this class in order to interact with the Captain agent in the JC2 framework. An example of BDAgent development is shown in Listing 2. It is the `BD_Loiter` agent that listens to the CFPReq for the mission task `LoiterMT` mentioned in section 3.1 and response with mission positions that exhibit loitering behavior. There are 4 IMPORTANT methods to be implemented and overrided from the DBAgent base class:

line 15: The base class `super.init()` method must be called in the inherited class's `init()` method.

line 51: The `handleCFPReq` method must be override to handle the incoming `CFPReq` messages. If the message containing the mission task that this BD agent can/should handle, it must reply with its own AgentID as shown in the listing.

line 73: The `handleMissionPtReq` method must be override to handle the incoming `MissionPtReq` messages. The BDAgent will only receive this when it has been contracted by the Captain agent as the mission position generator for the particular mission leg. Depending on the implemented algorithms, the BDAgent must reply with its generated mission positions. Since the mission position generation and execution are asynchronous, please read through the listing to find out how one can be notified when the mission positions generated are being executed.

line 97: The `onStop()` method must be implemented by the inherited class to stop all the operations when the method is called by the Captain agent. This method will only be called if the mission is aborted for whatever reason.

```
 1  class BD_Loiter extends BDAgent{

 3    //mp1 as the current point, mp2 is 50 meters apart diagonally.
      MissionPosition mp1 = null , mp2 = null
 5
      //boolean to keep track which of the mission points has been sent
 7    boolean isMp1Sent = true

 9    //Position to record the current position.
      Position curPos
11

13    //This method is called when the agent is initialized and added to
          the platform. The super class's init() method must be called.
      @Override
15    void init() {
        super.init()
17
        //subscribe to vehicle status for current position
19      subscribe(topic(C2Topics.VEHICLESTATUS))

21      //msgHandler is of type Map that maps the message class to its
            handler. Depending on the received message, its corresponding
            handler will be called.
        this.msgHandler << [
23          (VehicleStatus.class) : {handleVehicleStatus(it)},
          ]
25
        //New handlers can be added as needed. It is optional for the
            developers to make use of the super class's msgHandler. If
            preferred, new handler can be declared in this extended class,
            and new MessageBehavior can be added to listen to the new and
            desired message types.
27
    //      add(new MessageBehavior(){
29  //        @Override
    //        void onReceive(Message msg){
31  //          if(msg instanceof VehicleStatus){
    //            //handle VehicleStatus message here.
33  //          }
    //        }
35  //      })
      }
37
      /**
39     * Handler for the VehicleStatus message. For the purpose of this
           demo, the vehicle's current position is recorded whenever a
           VehicleStatus message is received.
       *
41     * @param msg Message of the type VehicleStatus
       */
43    void handleVehicleStatus(VehicleStatus msg) {
        curPos = msg.getPos()
```

```
}

/**
 * Implementation of the abstract method. Since this BDAgent is
     programmed  to handle LoiterMT mission task, whenever a CFPReq
     message is received with the content LoiterMT, this agent should
      reply with CFPRes message containing its own AgentID.
 */
@Override
public void handleCFPReq(CFPReq msg) {
  if(msg.getMt() instanceof LoiterMT){
    LoiterMT lmt = msg.getMt()
    CFPRes reply = new CFPRes(msg)
    reply.setConID(getAgentID())
    send(reply)

    //Assign the mission point for the vehicle to loiter.
    mp1 = new MissionPosition(curPos)
    mp2 = new MissionPosition(curPos.x-50,curPos.y-50)
  }
}


/**
 * Implementation of the abstract method. If this agent is contracted
      to provide the mission point for the particular mission leg,
     this method will be called whenever a MissionPtReq message is
     received from the Captain. It is  the BDAgent's responsibility
     to keep track of the mission points sent.
 *
 * Every MissionPosition generated comes with a unique mpID, this
     mpID will be broadcasted as the MissionPointStatusNtf message
     during the mission point's execution by the Pilot. BDAgents whom
      the implemented algorithms are missionposition depended should
     subscribe to this  message and react accordingly when the
     submitted missionposition is being executed.
 *
 *  Alternatively, the BDAgent can reply with MissionTasksRes message
      type to delegate the mission position  generation to another
     BDAgent (Lawnmover pattern etc.), if they exist in the agent
     services pool. Every Mission Task  generation comes with a
     unique taskID too, and this taskID will be broadcasted together
     with the missionpositions's mpID generated under the delegated
     agent's execution (all in the MissionPointStatusNtf message).
     Thus, the BDAgent can listen to this message and act accordingly
      when the submitted task is being executed.
 */
@Override
public void handleMissionPtReq(MissionPtReq msg) {

  //BDAgent should reply with MissionPtRes with the mission position,
       or set the missionposition's state to COMPLETED.

  MissionPtRes rm = new MissionPtRes(msg,Performative.INFORM)
  if(mp1 != null && mp2 != null){
    if(isMp1Sent){
      rm.setMp(mp2)
```

```
            isMp1Sent = false
        }else{
            rm.setMp(mp1)
            isMp1Sent = true
        }
    }else {
        //inform the Captain this mission task has completed.
        rm.setState(State.COMPLETED)
    }
    send(rm)
  }

  /**
   * This method will be called whenever the mission is aborted by
       whatever reason. Upon receiving the call, the BDAgent must stop
       all their operations and re-initialize the states if necessary.
   */
  @Override
  public void onStop() {
    mp1 = null
    mp2 = null
  }

}
```
Listing 2: BD_Loiter.groovy

4 Software Compilation

4.1 JC2_Template Folder

The given template folder should at least consists of two sub-folders and an ant build file as shown in Fig 10. The lib folder contains a jar file has all the base classes required (both fjåge and JC2). The src folder, of course, contains the source codes of the BDAgent and the mission task that the agent is responsible for. The ant build file consists of some generic compilation rules for the final jar compilation. As long as you have all the source codes in the src directory and jars in the lib directory, you are fine !

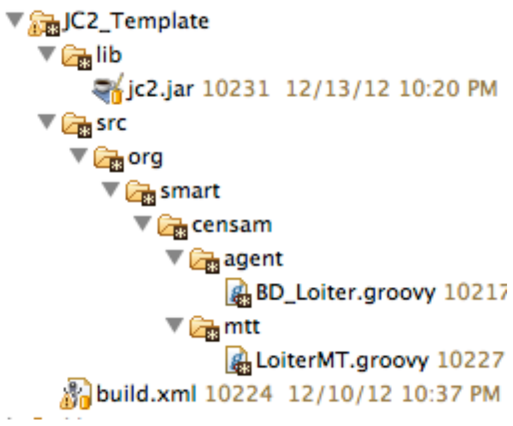


Figure 10: Folder tree of the JC2_Template folder.

4.2 The little "Ant" - build.xml

In order for the StarControl GUI to be aware of all the available Mission Task Types (mtt) during the mission planning, the developers must specify the package where their Mission Task (in this case, it is the package that the `LoiterMT` belong to) resides. From the listing shown in section 3.1, it is "`org.smart.censam.mtt`". A new attribute tag must be added under the manifest tag with the `name=`` Mtt-Package"` and `value=``org.smart.censam.mtt"` as shown in line 4 of Listing 3.

```
1   <manifest file="MANIFEST.MF">
      <attribute name="Built-By" value="${user.name}"/>
3     <attribute name="Built-Timestamp" value="${timestamp}"/>
      <attribute name="Mtt-Package" value="org.smart.censam.mtt"/>
5   </manifest>
```

Listing 3: build.xml

Once this has been taken care of, you can just "Ant" it !!!

5 Simulation and Data Analysis

5.1 Mission Files Location

Once the StarControl GUI is opened, you will be presented with a FileDialog asking for "Mission File Directory" as shown in Fig. 11. Choose the directory where all the mission files are stored. For the purpose of this simulation demonstration, the mission files are located at the same directory as the StarControl.app.

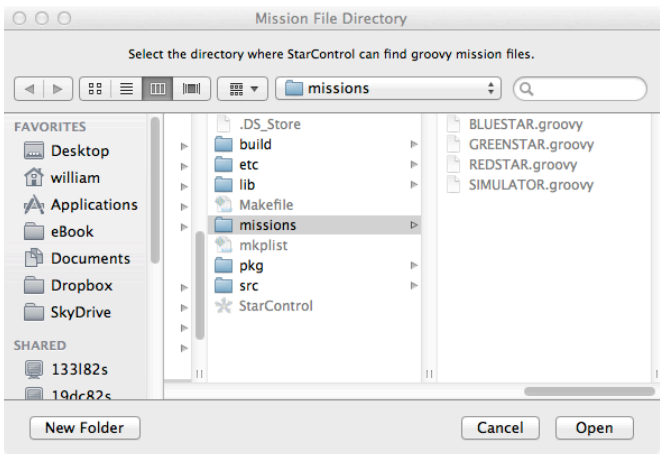


Figure 11: FileDialog for specifying the location of the mission files.

5.2 Add the Compiled Jar to StarControl

In order for the StarControl GUI to be aware of the newly developed LoiterMT mission task (Refer to section 3.1), its compiled Jar file must be added to the GUI's lib directory. To do this, go to "StarCtrl2" menu bar, **Administration**>**Add Jar** menu to open the "Select JarFile" FileDialog. Selected the compiled jar and hit the "open" button.

5.3 StarControl

The StarControl GUI is the main application used for mission planning and control. The icons labelled in Fig. 12 show the core functionalities required to operate the simulation, and will be explained in the following sections.

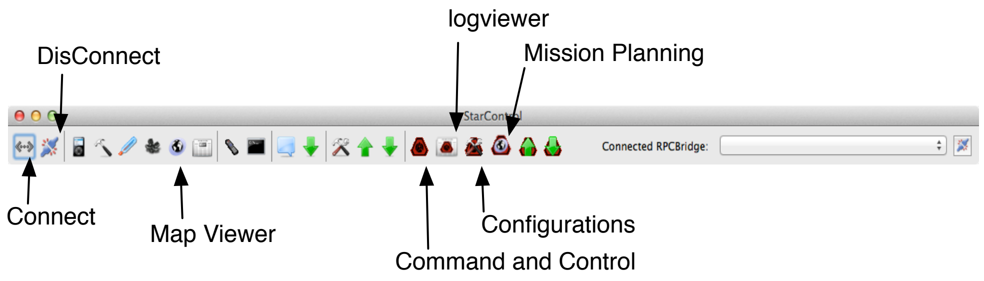
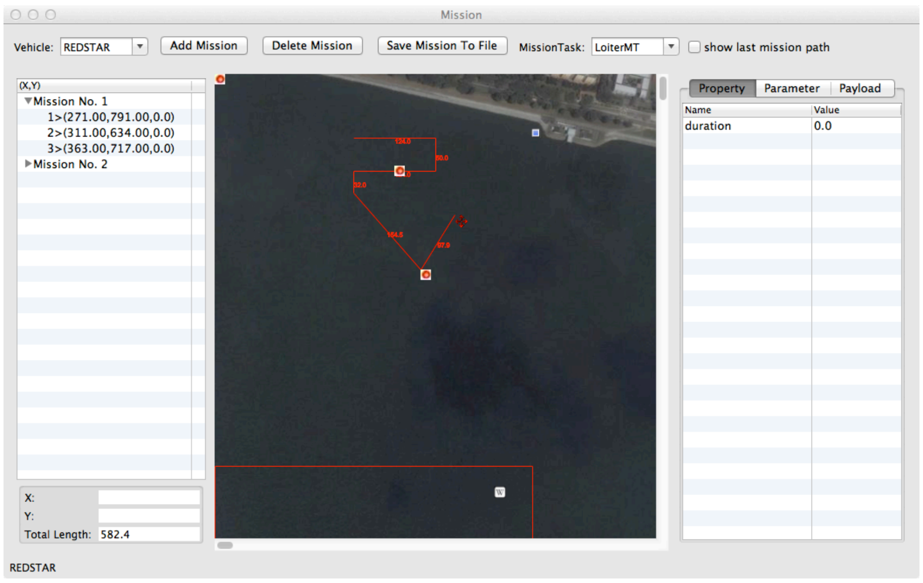


Figure 12: The StarControl GUI.

5.4 Mission Planning

The mission planning GUI is shown in Fig. 13. It allows the operator to select the **Vehicle** to plan the mission for; adding, deleting and saving the planned missions and specifying the values of the **Mission Task** properties. The Operator first select the **Vehicle**, then select the mission number. The planned mission with the associated mission number will be displayed in the map area. The operator can then interact directly with the mission tasks.

For the purpose of this demonstration, if the `LoiterMT` and `BD_Loiter` agent have been compiled and the step mentioned in section 5.2 has been performed correctly, the operator should be able to see the newly added mission task as one of the options in the **MissionTask** dropdown combo box. If that particular mission task has been selected and a mission point has been planned (by click on a location in the map area), the operator should see the properties defined in section 3.1 being displayed under the **Property** tab with their default values as shown in Fig. 13.



Vehicle: REDSTAR
MissionTask: LoiterMT
show last mission path

(X,Y)
Mission No. 1
1>(271.00,791.00,0.0)
2>(311.00,634.00,0.0)
3>(363.00,717.00,0.0)
Mission No. 2

X:
Y:
Total Length: 582.4

Name	Value
duration	0.0

REDSTAR

Figure 13: The mission planning GUI.

5.5 Configurations

When the **Configurations** icon is clicked, the operator will be presented with the init script (Assuming that TextWrangler is installed.). Listing 4 shows the content of the init script for starting the simulated vehicle using the StarControl GUI. There are a few important things to take note:

line 6: This is where you define the name for this simulated vehicle. Each vehicle must have a qualified name (one of the entries in the enum file extending the `org.arl.jc2.enums.Vehicle` interface). During the simulation, only the command sent to this vehicle (**Destination** dropdown shown in section 5.8) will be accepted.

line 9: There are two types of platforms can be started for the simulation. Please refer to the JAF documentation for more details. Basically the `DiscreteEventSimulator()` allows one to run the simulation as fast as the machine can support, thus, save the developer's time in obtaining the resultant/executed paths/patterns. However, the live position feedback that will be shown in section 5.9 may not make sense.

line 22: This is where the simulated AUV is added to the simulation environment. The user can specify the start locations and other properties that are defined in `org.arl.jc2.agent.AUVSim` class. If observing the simulation live on the map viewer is desired, one could start the simulation in `RealTimePlatform()`, and speedy up the AUV by changing the `thrustScale` and the `turnRate`.

Note ! the simulated AUV's dynamic will change too, "hack" at your own risk !!!

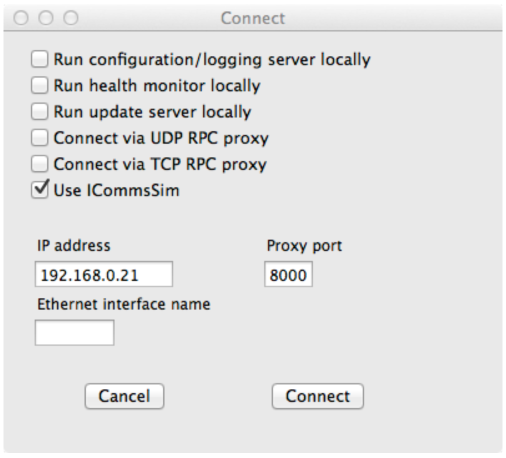
line 45: This is where the newly developed third party BD agent (`BD_LOITER`) is added.

```
 1
   //initrc script to start all the necessary container and setup udplink
 3
   //################################################## Config
 5 //##########################################################Parameters
   def AUVName = org.arl.jc2.enums.StarfishVehicles.SIMULATOR
 7
   //################################################## Plaform
 9 //Starting the RealTime platform and Container
   def p = new org.arl.jaf.RealTimePlatform()
11 //def p = new org.arl.jaf.DiscreteEventSimulator()
   def c = new org.arl.jaf.Container(p)
13
   //################################################## UdpLinkAgent
15 //setting up udplink agent
   def udp = new org.arl.unet.link.UdpLink()
17 udp.setMulticastIface('en0')
   udp.setAddress(AUVName.getAddress())
19 c.add "udp",udp

21 //####################################################auvSim
   def auv = new org.arl.jc2.agent.AUVSim()
23 auv.xPos = 140
   auv.yPos = 770
25 c.add "AUVSim", auv

27
   //################################################## Captain
29 def captain = new org.arl.jc2.agent.Captain()
   captain.VEHICLEID = AUVName
31 c.add "CAPTAIN", captain

33 c.add "COOPMDP"           , new org.arl.jc2.agent.BD_COOPMDP()
   c.add "COOPAUVSIM"          , new org.arl.jc2.agent.BD_COOPAUVSIM()
35 c.add "MISSIONGENERATOR"    , new org.arl.jc2.agent.MissionAgentLoader(
        missionDir:'./../../../missions/')
   c.add "SIMPLEMP"            , new org.arl.jc2.agent.BD_SimpleMP()
37 c.add "ABORTER"           , new org.arl.jc2.agent.BD_Aborter()
   c.add "STATIONKEEPING"      , new org.arl.jc2.agent.BD_StationKeeping()
39 c.add "LAWNMOWER"         , new org.arl.jc2.agent.BD_LawnMower()
   c.add "EXECUTIVEOFFICER"    , new org.arl.jc2.agent.ExecutiveOfficer()
41 c.add "NAVIGATOR"         , new org.arl.jc2.agent.Navigator()
   c.add "PILOT"             , new org.arl.jc2.agent.Pilot()
43 c.add "SIGOFFICER"          , new org.arl.jc2.agent.SignalingOfficer()

45 //####################################################ThirdPartyBDAgent
   c.add "LOITER"            , new org.smart.censam.agent.BD_Loiter()
47

49 //##########################################Start the platform
   p.start()
```

Listing 4: initSim.groovy

5.6 Connect (Start Simulation)

When the **Connect** icon is clicked, the **Connect** window is shown as in Fig. 14. Check the **Use ICommsSim** checkbox and hit the **Connect** button, the simulation should start !


Connect

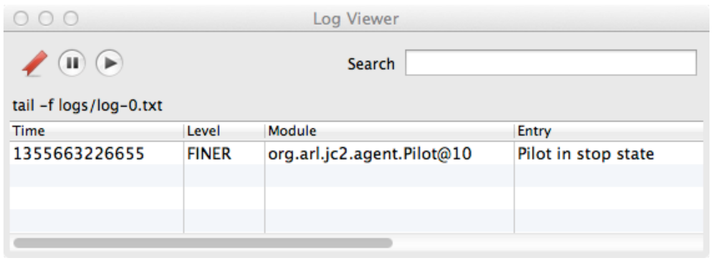
- [ ] Run configuration/logging server locally
- [ ] Run health monitor locally
- [ ] Run update server locally
- [ ] Connect via UDP RPC proxy
- [ ] Connect via TCP RPC proxy
- [x] Use ICommsSim

IP address: 192.168.0.21
Proxy port: 8000
Ethernet interface name:

Figure 14: Start and connect to the simulation.

5.7 LogViewer

One way to check if the simulation is started correctly is to open the **LogViewer** window. If the **JC2** agents are up and running, the **Pilot** agent should report that it is in "Stop" state as seen in Fig. 15. When a C2 command is issued and the simulation is running, this log window should be very active dumping out all the logs.



tail -f logs/log-0.txt

Time	Level	Module	Entry
1355663226655	FINER	org.arl.jc2.agent.Pilot@10	Pilot in stop state

Figure 15: Simulated vehicle's log.

5.8 Command and Control (C2)

The C2 panel allows the operator to send different C2 command to the vehicle. The **Destination** field must match to the name of the vehicle specified in Listing 4 line number 6.

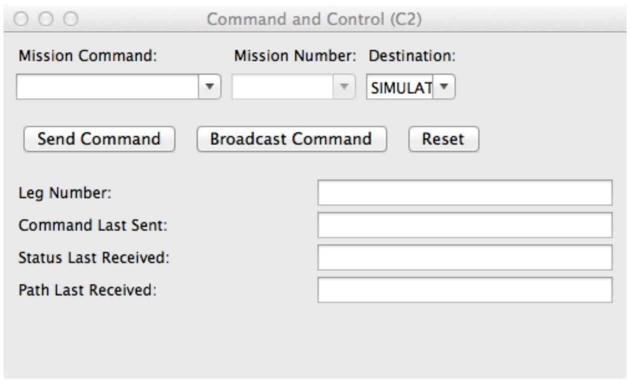


Figure 16: The command and control panel.

5.9 Map Viewer

The map viewer shows the current location of the vehicle during the simulation. The trajectory of the vehicle is shown in yellow dot/lines in the map, while the vehicle's current location (x and y) is shown at the right top corner of the window.

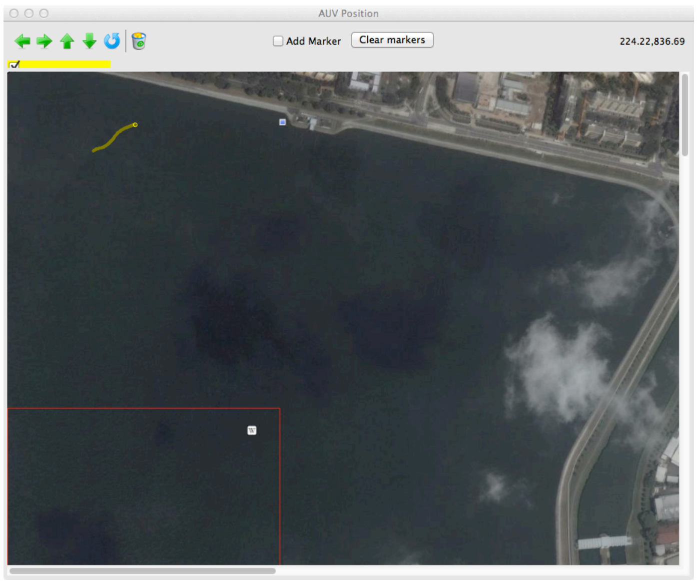


Figure 17: The map viewer showing the location of the vehicle (yellow dot/lines) and the location of the vehicle in the map (two numbers on the right top corner of the window).

5.10 LogFile Extraction

Once the simulation has completed, the user can copy the log files to their desired directory. This can be done by clicking on the **Administration**>**Extract Logs** menu. Once the user has chosen the destination directory, all the files in the underlying "logs" directory inside the StarControl.app will be copied to the specified location. The content of the simLog folder are :

1. `c2log-*.log` : The main log file of the JC2 Agents.
2. `m*` : Folders containing all the logs extracted from the log-0.txt.
3. `time.txt` : file contains all the start and end time for each of the missions.
4. `guilog-0.txt` : The log from the GUI agent. Only used for debugging purpose.

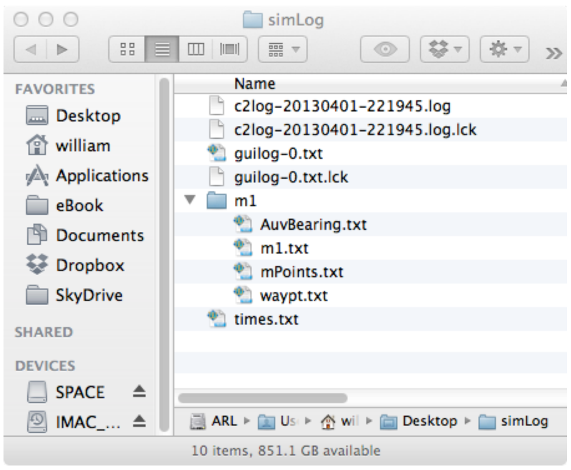
The format of the files in the m$*$ folder are as follows:

1. `mPoints.txt` : The planned mission points.
   <timeStamp xPos yPos zPos >
2. `waypt.txt` : The planned way points.
   <timeStamp xPos yPos zPos >

3. `AuvBearing.txt` : The vehicle's positions, bearings and distances to the next way points.
   <timeStamp xPos yPos zPos bearing distanceToNextWayPt >

4. `m*.txt` : The original c2 logs of the particular mission.

with the extracted data, one can easily plot the resultant trajectories using any plotting programs.



```
c2log-20130401-221945.log
c2log-20130401-221945.log.lck
guilog-0.txt
guilog-0.txt.lck
▼ m1
    AuvBearing.txt
    m1.txt
    mPoints.txt
    waypt.txt
times.txt
```

10 items, 851.1 GB available

Figure 18: Folder "simLog" contains the original log files and the extracted log files.

Bibliography

[1] J. Elvander and G. Hawkes. ROVs and AUVs in support of marine renewable technologies. In *Oceans, 2012*, pages 1–6, Oct 2012.

[2] C. von Alt, B. Allen, T. Austin, and R. Stokey. Remote environmental measuring units. In *Autonomous Underwater Vehicle Technology, 1994. AUV '94., Proceedings of the 1994 Symposium on*, pages 13–19, Jul 1994.

[3] T.B. Koay, Y.T. Tan, Y.H. Eng, R. Gao, Mandar Chitre, J.L. Chew, N. Chandhavarkar, R.R. Khan, T. Taher, and J. Koh. STARFISH – a small team of autonomous robotic fish. *Indian Journal of Geo-Marine Sciences*, 20(2):157–167, April 2011.

[4] C.C. Eriksen, T.J. Osse, R.D. Light, T. Wen, T.W. Lehman, P.L. Sabin, J.W. Ballard, and AM. Chiodi. Seaglider: a long-range autonomous underwater vehicle for oceanographic research. *Oceanic Engineering, IEEE Journal of*, 26(4):424–436, Oct 2001. ISSN 0364-9059.

[5] J. Sherman, R. Davis, W. B. Owens, and J. Valdes. The autonomous underwater glider "spray". *Oceanic Engineering, IEEE Journal of*, 26(4):437–446, Oct 2001. ISSN 0364-9059.

[6] Amy Nevala. A glide across the gulf stream. *OCEANUS*, 44(1), June 2005.

[7] W. J. Kirkwood. Development of the DORADO mapping vehicle for multibeam, subbottom, and sidescan science missions. *Journal of Field Robotics*, 24(6):487–495, 2007. ISSN 1556-4967. doi: 10.1002/rob.20191.

[8] P. Wadhams and M. J. Doble. Digital terrain mapping of the underside of sea ice from a small AUV. *Geophysical Research Letters*, 35(1), 2008. ISSN 1944-8007. doi: 10.1029/2007GL031921.

[9] StefanB. Williams, Oscar Pizarro, Michael Jakuba, and Neville Barrett. AUV benthic habitat mapping in south eastern Tasmania. In Andrew Howard, Karl Iagnemma, and Alonzo Kelly, editors, *Field and Service Robotics*, volume 62 of *Springer Tracts in Advanced Robotics*, pages 275–284. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13407-4. doi: 10.1007/978-3-642-13408-1_25.

[10] Yanwu Zhang, J.G. Bellingham, M.A. Godin, and J.P. Ryan. Using an autonomous underwater vehicle to track the thermocline based on peak-gradient detection. *Oceanic Engineering, IEEE Journal of*, 37(3):544 –553, july 2012. ISSN 0364-9059. doi: 10.1109/JOE.2012.2192340.

[11] B. Anderson and J. Crowell. Workhorse AUV - a cost-sensible new autonomous underwater vehicle for surveys/soundings, search, rescue, and research. In *OCEANS, 2005. Proceedings of MTS/IEEE*, pages 1–6, Sept 2005. doi: 10.1109/OCEANS.2005.1639923.

[12] A. Matos, N. Cruz, A. Martins, and F. Lobo Pereira. Development and implementation of a low-cost LBL navigation system for an AUV. In *OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Century*, volume 2, pages 774 –779 vol.2, 1999. doi: 10.1109/OCEANS.1999.804906.

[13] P. Rigby, O. Pizarro, and S.B. Williams. Towards geo-referenced AUV navigation through fusion of USBL and DVL measurements. In *OCEANS 2006*, pages 1 –6, 2006. doi: 10.1109/OCEANS.2006.306898.

[14] A. Alcocer, P. Oliveira, and A. Pascoal. Study and implementation of an EKF GIB-based underwater positioning system. *Control Engineering Practice*, 15(6): 689 – 701, 2007. ISSN 0967-0661. doi: 10.1016/j.conengprac.2006.04.001.

[15] Sarah E Webster, Ryan M Eustice, Hanumant Singh, and Louis L Whitcomb. Advances in single-beacon one-way-travel-time acoustic navigation for underwater vehicles. *The International Journal of Robotics Research*, 31(8):935–950, 2012.

[16] J. C. Alleyne. Position estimation from range only measurements. Master's thesis, Naval Postgraduate School, Monterey CA, September 2000.

[17] Alexander Bahr, John J. Leonard, and Maurice F. Fallon. Cooperative localization for autonomous underwater vehicles. *The International Journal of Robotics Research*, 28(6):714–728, 2009.

[18] Maurice F Fallon, Georgios Papadopoulos, John J Leonard, and Nicholas M Patrikalakis. Cooperative AUV Navigation using a Single Maneuvering Surface Craft. *The International Journal of Robotics Research*.

[19] A.P. Scherbatyuk. The AUV positioning using ranges from one transponder LBL. In *OCEANS '95. MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings.*, volume 3, pages 1620–1623 vol.3, Oct 1995.

[20] J. Hartsfiel. Single transponder range only navigation geometry (STRONG) applied to REMUS autonomous under water vehicles. Master's thesis, MIT, 2005.

[21] A. Bahr, J.J. Leonard, and A. Martinoli. Dynamic positioning of beacon vehicles for cooperative underwater navigation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3760–3767, 2012. doi: 10.1109/IROS.2012.6386168.

[22] M. Chitre. Path planning for cooperative underwater range-only navigation using a single beacon. In *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, pages 1 –6, 2010. doi: 10.1109/AIS.2010.5547044.

[23] D.K. Meduna, S.M. Rock, and R.S. McEwen. Closed-loop terrain relative navigation for AUVs with non-inertial grade navigation sensors. In *Autonomous Underwater Vehicles (AUV), 2010 IEEE/OES*, pages 1–8, 2010.

[24] S. Carreno, P. Wilson, P. Ridao, and Y. Petillot. A survey on terrain based navigation for AUVs. In *OCEANS 2010*, pages 1–7, 2010. doi: 10.1109/OCEANS.2010.5664372.

[25] Bharath Kalyan and Mandar Chitre. A feasibility analysis on using bathymetry for navigation of autonomous underwater vehicles. In *Proceedings of the 28th*

Annual ACM Symposium on Applied Computing, SAC '13, pages 229–231, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1656-9. doi: 10.1145/2480362.2480411.

[26] I. Nygren and M. Jansson. Terrain navigation for underwater vehicles using the correlator method. *Oceanic Engineering, IEEE Journal of*, 29(3):906–915, 2004. ISSN 0364-9059. doi: 10.1109/JOE.2004.833222.

[27] K.B. Anonsen and O. Hallingstad. Terrain aided underwater navigation using point mass and particle filters. In *Position, Location, And Navigation Symposium, 2006 IEEE/ION*, pages 1027–1035, April 2006. doi: 10.1109/PLANS.2006.1650705.

[28] R. Karlsson and F. Gustafsson. Particle filter for underwater terrain navigation. In *Statistical Signal Processing, 2003 IEEE Workshop on*, pages 526 – 529, sept.-1 oct. 2003. doi: 10.1109/SSP.2003.1289507.

[29] P.-J. Nordlund and F. Gustafsson. Sequential monte carlo filtering techniques applied to integrated navigation systems. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4375–4380 vol.6, 2001. doi: 10.1109/ACC.2001.945666.

[30] T. Schon, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *Signal Processing, IEEE Transactions on*, 53 (7):2279–2289, 2005. ISSN 1053-587X. doi: 10.1109/TSP.2005.849151.

[31] F Teixeira, António Pascoal, and Pramod Maurya. A novel particle filter formulation with application to terrain-aided navigation. In *Proc. IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV'2012), Porto, Portugal*, pages 10–12, 2012.

[32] Arnaud Doucet, Nando de Freitas, Kevin P. Murphy, and Stuart J. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-709-9.

[33] Stephen Barkby, Stefan B. Williams, Oscar Pizarro, and Michael V. Jakuba. A featureless approach to efficient bathymetric SLAM using distributed particle mapping. *Journal of Field Robotics*, 28(1):19–39, 2011. ISSN 1556-4967.

[34] Nathaniel Fairfield, George A Kantor, and David Wettergreen. Towards particle filter SLAM with three dimensional evidence grids in a flooded subterranean environment. In *Proceedings of ICRA 2006*, pages 3575 – 3580, May 2006.

[35] N. Fairfield and D. Wettergreen. Active localization on the ocean floor with multibeam sonar. In *OCEANS 2008*, pages 1–10, 2008. doi: 10.1109/OCEANS.2008.5151853.

[36] G.T. Donovan. Position error correction for an autonomous underwater vehicle inertial navigation system (INS) using a particle filter. *Oceanic Engineering, IEEE Journal of*, 37(3):431 –445, July 2012. ISSN 0364-9059. doi: 10.1109/JOE.2012.2190810.

[37] C. Roman and H. Singh. Improved vehicle based multibeam bathymetry using sub-maps and SLAM. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3662–3669, Aug 2005.

[38] M. F. Fallon, M. Kaess, H. Johannsson, and J. J. Leonard. Efficient AUV navigation fusing acoustic ranging and side-scan sonar. In *IEEE International Conference on Robotics and Automation (ICRA),Shanghai, China*, May 2011.

[39] Francisco Curado Teixeira, João Quintas, and António Pascoal. AUV terrain-aided doppler navigation using complementary filtering. In *Proc IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC'2012), Arenzano, Italy*, 2012.

[40] Pramod Maurya, Francisco Curado Teixeira, and António Pascoal. Complementary terrain/single beacon-based AUV navigation. In *Proc IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles (NGCUV'2012), Porto, Portugal*, pages 10–12, 2012.

[41] Matt Rosencrantz, Geoffrey Gordon, and Sebastian Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Nineteenth conference*

on Uncertainty in Artificial Intelligence, UAI'03, pages 493–500, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc. ISBN 0-127-05664-5.

[42] Bo Jiang and B. Ravindran. Completely distributed particle filters for target tracking in sensor networks. In *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 334–344, 2011. doi: 10.1109/IPDPS.2011.40.

[43] Xiaohong Sheng, Yu-Hen Hu, and P. Ramanathan. Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 181–188, 2005. doi: 10.1109/IPSN.2005.1440923.

[44] H. Yavuz and A. Bradshaw. A new conceptual approach to the design of hybrid control architecture for autonomous mobile robots. *Journal of Intelligent and Robotic Systems*, 34(1):1–26, 2002.

[45] Alexander M. Meystel and James Sacra Albus. *Intelligent Systems: Architecture, Design, and Control*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2000. ISBN 0471193747.

[46] Ronald C. Arkin. *An Behavior-based Robotics*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262011654.

[47] P. Ridao, J. Yuh, J. Batlle, and K. Sugihara. On AUV control architecture. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 2, pages 855–860 vol.2, 2000. doi: 10.1109/IROS.2000.893126.

[48] A. Yavnai. Architecture for an autonomous reconfigurable intelligent control system (ARICS). In *Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on*, pages 238–245, Jun 1996. doi: 10.1109/AUV.1996.532421.

[49] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback. Towards component-based robotics. In *Intelligent Robots and Systems, 2005. (IROS 2005).*

2005 IEEE/RSJ International Conference on, pages 163–168, Aug 2005. doi: 10.1109/IROS.2005.1545523.

[50] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. *ICRA workshop on open source software*, 3(3.2), 2009.

[51] Paul Michael Newman. MOOS-mission orientated operating suite. *Massachusetts Institute of Technology, Tech. Rep*, 2299(08), 2008.

[52] Michael R. Benjamin, Henrik Schmidt, Paul M. Newman, and John J. Leonard. Nested autonomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics*, 27(6):834–875, 2010. ISSN 1556-4967. doi: 10.1002/rob.20370.

[53] Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 117(2):277 – 296, 2000. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(99)00107-1.

[54] Y. T. Tan and Mandar Chitre. Single beacon cooperative path planning using cross-entropy method. In *IEEE/MTS OCEANS, KONA, Hawaii*, September 2011.

[55] Y. T. Tan and Mandar Chitre. Direct policy search with variable-length genetic algorithm for single beacon cooperative path planning. In *International Symposium on Distributed Autonomous Robotic Systems (DARS) 2012*, Baltimore, Maryland, USA, November 2012.

[56] Y. T. Tan, Rui Gao, and M. Chitre. Cooperative path planning for range-only localization using a single moving beacon. *Oceanic Engineering, IEEE Journal of*, 39(2):371–385, April 2014. ISSN 0364-9059. doi: 10.1109/JOE.2013.2296361.

[57] Marco Wiering and Martijn van Otterlo. *Reinforcement Learning: State of the Art*, volume 12 of *Adaptation, Learning, and Optimization*. Springer Berlin Heidelberg, 2012.

[58] RonaldJ. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[59] Jonathan Baxter, Peter L. Bartlett, and Lex Weaver. Experiments with infinite-horizon, policy-gradient estimation. *J. Artif. Int. Res.*, 15(1):351–381, November 2001. ISSN 1076-9757.

[60] Thomas Philip Runarsson. Learning heuristic policies – A reinforcement learning problem. In *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 423–432. Springer Berlin Heidelberg, 2011.

[61] Pieter tjerk De Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134:19–67. doi: 10.1007/s10479-005-5724-z.

[62] Shie Mannor, Reuven Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *In International Conference on Machine Learning*, pages 512–519. Morgan Kaufmann, 2003.

[63] Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley Series in Probability and Statistics. Wiley, 2nd edition, 2011.

[64] D E Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[65] Jianping Tu and S.X. Yang. Genetic algorithm based path planning for a mobile robot. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 1221 – 1226, sept. 2003. doi: 10.1109/ROBOT.2003.1241759.

[66] Chang Wook Ahn and R.S. Ramakrishna. A genetic algorithm for shortest path routing problem and the sizing of populations. *Evolutionary Computation, IEEE Transactions on*, 6(6):566 – 579, dec 2002. ISSN 1089-778X. doi: 10.1109/TEVC.2002.804323.

[67] Jun Wei, Haining Zheng, Haoliang Chen, Boon Hooi Ooi, M. H. Dao, Wonjoon Cho, Paola M. Rizzoli, P. Tkalich, and Nicholas M. Patrikalakis. Multi-layer model simulation and data assimilation in the Serangoon Harbor of Singapore. In *International Offshore (Ocean) and Polar Engineering Conference*, June 2010.

[68] P.O. Arambel, Constantino Rago, and R.K. Mehra. Covariance intersection algorithm for distributed spacecraft state estimation. In *American Control Conference, 2001. Proceedings of the 2001*, volume 6, pages 4398–4403 vol.6, 2001. doi: 10.1109/ACC.2001.945670.

[69] A. Bahr, M.R. Walter, and J.J. Leonard. Consistent cooperative localization. In *IEEE International Conference on Robotics and Automation (ICRA), Kobe, Japan*, May 2009.

[70] Y. T. Tan, Mandar Chitre, and F. Hover. Collaborative bathymetry-based localization of a team of autonomous underwater vehicles. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014.

[71] F Teixeira. *Terrain-Aided Navigation and Geophysical Navigation of Autonomous Underwater Vehicles*. PhD thesis, Dynamical Systems and Ocean Robotics Lab, Lisbon, 2007.

[72] Per-Johan Nordlund. *Sequential Monte Carlo Filters and Integrated Navigation*. PhD thesis, Linkopings universitet, 2002.

[73] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1): 34–46, Feb 2007. ISSN 1552-3098.

[74] Jun S. Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

[75] Paul Fearnhead. *Sequential Monte Carlo methods in filter theory*. PhD thesis, University of Oxford, 1998.

[76] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0471241954.

[77] Oswald Lanz. An information theoretic rule for sample size adaptation in particle filtering. *Image Analysis and Processing, International Conference on*, 0:317–322, 2007.

[78] S.E. Webster, J.M. Walls, L.L. Whitcomb, and R.M. Eustice. Decentralized extended information filter for single-beacon cooperative acoustic navigation: Theory and experiments. *Robotics, IEEE Transactions on*, 29(4):957–974, 2013. ISSN 1552-3098. doi: 10.1109/TRO.2013.2252857.

[79] D.P. Eickstedt and S.R. Sideleau. The backseat control architecture for autonomous robotic vehicles: A case study with the Iver2 AUV. In *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, pages 1 –8, 26-29 2009.

[80] Y. T. Tan, Mandar Chitre, and Prahlad Vadakkepat. Hierarchical agent-based command and control system for autonomous underwater vehicles. In *International Conference on Autonomous and Intelligent Systems (AIS) 2010*, Povoa de Varzim, Portugal, June 2010.

[81] Y. T. Tan and Mandar Chitre. Hierarchical multi-agent command and control system for autonomous underwater vehicles. In *IEEE AUV 2012*, Southampton, UK, September 2012.

[82] M. Chitre. DSAAV - A distributed software architecture for autonomous vehicles. In *OCEANS 2008*, pages 1 –10, sept. 2008. doi: 10.1109/OCEANS.2008.5151848.

[83] G. Hollinger and S. Singh. Multi-robot coordination with periodic connectivity. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4457 –4462, May 2010.

[84] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen. A deliberative architecture for AUV control. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1049 –1054, May 2008.

[85] Mandar Chitre, R. Bhatnagar, and W.-S. Soh. UnetStack: an agent-based software stack and simulator for underwater networks. In *IEEE OCEANS, St. John's, Canada*, September 2014.

[86] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[87] P. M Newman. MOOS—A mission oriented operating suite (Tech. Rep. OE2003-07). Technical report, Department of Ocean Engineering, MIT, Cambridge, MA:, 2003.

[88] Eng You Hong, Teo Kwong Meng, and Mandar Chitre. Online system identification of the dynamics of an autonomous underwater vehicle. In *Underwater Technology Symposium (UT), 2013 IEEE International*, pages 1–10, March 2013. doi: 10.1109/UT.2013.6519846.

[89] P.S. Dias, G.M. Goncalves, R.M.F. Gomes, J.B. Sousa, J. Pinto, and F.L. Pereira. Mission planning and specification in the Neptus framework. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3220–3225, May 2006.

[90] Chee-Loon Ng, Schuyler Senft-Grupp, and Harold F Hemond. A multi-platform optical sensor for in situ sensing of water chemistry. *Limnol. Oceanogr. Methods*, 10:978–990, 2012.

[91] M. Shaukat, Mandar Chitre, Y. T. Tan, and Ashish Raste. Bio-CAST: A bio-inspired control algorithm for small team of robots using implicit communication for cooperative source localization. *Bioinspired & Biomimetics (Submitted)*, 2014.

[92] Frederic Py, Kanna Rajan, and Conor McGann. A systematic agent framework for situated autonomous systems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 2*, AAMAS '10, pages 583–590, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0-98265-712-9.

Publications

[1] Mansoor Shaukat, Mandar Chitre, Y. T. Tan, and Ashish Raste. Bio-CAST: A bio-inspired control algorithm for small team of robots using implicit communication for cooperative source localization. *Bioinspired & Biomimetics (Submitted)*, 2014.

[2] Y. T. Tan, Mandar Chitre, and Franz Hover. Collaborative bathymetry-based localization of a team of autonomous underwater vehicles. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014.

[3] Y. T. Tan, R. Gao, and Mandar Chitre. Cooperative path planning for range-only localization using a single moving beacon. *Oceanic Engineering, IEEE Journal of*, 39(2):371–385, April 2014. ISSN 0364-9059. doi: 10.1109/JOE.2013.2296361.

[4] Y. T. Tan and Mandar Chitre. Direct policy search with variable-length genetic algorithm for single beacon cooperative path planning. In *International Symposium on Distributed Autonomous Robotic Systems (DARS) 2012*, Baltimore, Maryland, USA, November 2012.

[5] Y. T. Tan and Mandar Chitre. Hierarchical multi-agent command and control system for autonomous underwater vehicles. In *IEEE AUV 2012*, Southampton, UK, September 2012.

[6] T. B. Koay, Y. T. Tan, Y. H. Eng, R. Gao, Mandar Chitre, J. L. Chew, N. Chandhavarkar, R. Khan, T. Taher, and J. Koh. STARFISH - A small team of autonomous robotics fish. In *3rd International Conference on Underwater System Technology: Theory and Applications 2010, (Cyberjaya, Malaysia)*, November 2011.

[7] Y. T. Tan and Mandar Chitre. Single beacon cooperative path planning using cross-entropy method. In *IEEE/MTS OCEANS, KONA, Hawaii*, September 2011.

[8] T.B. Koay, Y.T. Tan, Y.H. Eng, R. Gao, Mandar Chitre, J.L. Chew, N. Chandhavarkar, R.R. Khan, T. Taher, and J. Koh. STARFISH – A small team of autonomous robotic fish. *Indian Journal of Geo-Marine Sciences*, 20(2):157–167, April 2011.

[9] J. L. Chew, T. B. Koay, Y. T. Tan, Y. H. Eng, R. Gao, Mandar Chitre, and N. Chandhavarkar. STARFISH: An Open-Architecture AUV and its Applications. Defence Technology Asia, 2011, February 2011.

[10] Y. T. Tan, Mandar Chitre, and Prahlad Vadakkepat. Hierarchical agent-based command and control system for autonomous underwater vehicles. In *International Conference on Autonomous and Intelligent Systems (AIS) 2010*, Povoa de Varzim, Portugal, June 2010.