ONLINE BANDWIDTH ADAPTATION AND DATA TRACKING OF UNDERWATER ACOUSTIC TRANSMISSIONS TO OPTIMIZE COLLABORATIVE AUV MISSIONS

by

Dainis Nams

Submitted in partial fulfilment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
December 2014

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A method for on-line characterization of in-water acoustic transmission conditions and intelligent adaptation of transmission rates to these conditions is implemented on autonomous underwater vehicles (AUV). The objective is to optimize use of the communications channel during collaborative shallow water missions with other AUVs. A software database cheaply (in terms of bandwidth) tracks the success of transmitted packets, providing operator data tracking in addition to communications layer visibility into current channel conditions. A rate selector chooses optimal transmission rates using an adaptive distance bin approach. Outcomes are improvements in bandwidth, reduction in modem power draw, and increased visibility into data success compared to traditional, constant-rate acoustic communication patterns.

LIST OF ABBREVIATIONS USED

ADCP	Acoustic Doppler Current Profiler
ASC	Autonomous Surface Craft
AUV	Autonomous Underwater Vehicle
CDMA	Code Division Multiple Access
CE-DFE	Channel Estimate Based Decision Feedback Equalizer
CNA	Communications and Navigation Aid
CML	Concurrent Localization and Mapping
DRDC	Defence Research and Development Canada
DFE	Decision Feedback Equalizer
DOF	Degrees of Freedom
DSP	Digital Signal Processing
DVL	Doppler Velocity Log
EIF	Extended Information Filter
EKF	Extended Kalman Filter
EM	Electromagnetic
FDMA	Frequency Division Multiple Access
GA	Genetic Algorithm
GPS	Global Positioning System
FKS	Frequency Shift Keyed
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ISI	Inter-Symbol Interference
LBL	Long Baseline
LMS	Least Mean Squares
MCM	Mine Countermeasures Mission
MI	Magneto-Inductive
MOOS	Mission Oriented Operating Suite
MLO	Mine Like Object
NMCM	Naval Mine Countermeasures Mission
NNAB	Neural Network for Avoidance Behaviour
OWTT	One Way Travel Time
RBPF	Rao-Blackwellized Particle Filter
RLS	Recursive Least Squares
PSK	Phase-Shift-Keyed
RF	Radio Frequency

RL	Reinforcement Learning
ROS	Robot Operating System
ROV	Remote Operated Vehicle
SDMA	Space Division Multiple Access
SEIF	Sparse Extended Information Filter
SLAM	Simultaneous Localization And Mapping
SODIM	Self-Organization Direction Mapping Network
SNR	Signal to Noise Ratio
SVM	Support Vector Machine
SZ	Surf Zone
TDMA	Time Division Multiple Access
TL	Transmission Loss
TWTT	Two Way Travel Time
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
VSW	Very Shallow Water

CHAPTER 1 INTRODUCTION

Autonomous underwater vehicles (AUV) are widely used in applications such as seabed survey, hull inspection, environmental monitoring, and naval threat detection. Initially used singly, AUVs are increasingly collaborating to complete large-scale missions. While many collaborative AUV applications exist, Naval Mine Counter-Measures (NMCM) missions are of interest to Defence Research and Development Canada (DRDC).

NMCM operations consist of several phases, the following of which can involve AUVs. Route surveys are conducted using a sidescan sonar before the introduction of mines to provide a baseline for the mission areas, e.g. entrances to ports. The first step to verifying mine presence or absence is the acquisition phase, where a survey similar to the route survey is conducted. Sidescan sonar data is compared against the prior route survey to detect potential mine like objects (MLOs). Next, the identification phase involves targeting suspected MLOs by surveying with a higher-resolution sonar to produce multi-aspect views of a target. Finally, once identity of a mine is confirmed, the mine is neutralized using one of many possibilities (for example, an expendable ROV) [1].

1.1. MOTIVATION

If implemented as a collaborative mission, NMCM requires a robust communications network between collaborating AUVs. Electromagnetic waves are quickly attenuated underwater, leaving underwater acoustic modems as the best method for inter-AUV communication as far as range is concerned. However, acoustic communications are limited to low frequency (<50kHz) carrier bands due to frequency-dependent absorption [2]. Network bandwidths are further reduced by scattering, natural and anthropogenic ambient noise sources, and time-varying sound speed profiles (SSPs). These factors limit typical mean bandwidths to bytes per second (B/s) rather than the kB/s and MB/s available to in-air radio networks. While sufficient to transmit/receive status updates it does not facilitate high-bandwidth collaborative algorithms such as distributed Simultaneous Localization and Mapping (SLAM), a research topic for DRDC Atlantic.

Increased communications bandwidth is an active research area with simple alternating high and low data rate [3] and acoustically optimized vehicle path solutions presented by Schneider [4]. The latter work used an online ray trace to simulate the acoustic channel and determine locations favorable for transmission which were used to inform the vehicle path planner. Optimal message routing is proposed by Chen and Pompilli who use the Urick channel model to make best use of the limited channel in a large fleet scenario [5]. These efforts proved effective in the scenarios to which they were applied, but would not be for NMCM due to the following unique constraints NMCM placesd on AUV behaviour.

When NMCM operate in shallow (< 100m) waters it introduces multi-path effects not captured in Pompilli's simple Urick model [5]. AUV paths are planned to optimize sidescan sonar coverage and not usually for acoustic performance as implemented in [4]. NMCM AUVs cannot be assumed to have environmental sensors such as the conductivity, temperature, and depth (CTD) sensors used in [4]. To improve communications for NMCM operations intelligent communications control is necessary.

1.2. Thesis Contributions

This thesis presents a framework that uses an on-line adaptive transmission rate selector along with improved data transmission performance tracking. These combine to allow the NMCM AUV to intelligently adapt its transmission rate by learning from the immediate past rather than relying on unavailable sensor information. The contributions of this thesis are as follows:

1. An adaptive data rate behaviour that yields improved bandwidth and reduced duty cycles compared to traditional, non-adaptive methods at only moderate cost to frequency of success. This behaviour does not require detailed measurements of the acoustic channel as it learns from its recent history of successes and failures.

This is significant because it provides a computationally cheap platform for optimized data rate choice-making in shallow-water NMCM scenarios; a feature not yet provided in the literature.

2. A robust software data tracking transmission performance system that is more reliable than the WHOI Micro-modem's built-in acknowledgement feature while costing significantly less bandwidth. This is significant because it gives data transfer protocols reliable knowledge of whether or not data reached the intended recipient, reducing the need for unnecessary re-transmissions while ensuring lost data is re-transmitted.

3. Quantitative analysis of relative performance of four transmission rates provided by the WHOI micromodem using data collected from the 17th October, 2013 field trials. This is significant because a rate-by-rate comparison of WHIO Micro-Modem performance has not been published in the literature to date.

1.3. Thesis Organization

This thesis is organized according to the following sections:

- Chapter 2: Literature review covering AUV positioning methods with focus on SLAM applied to NMCM operations and acoustic communications with focus on the environmental adaptivity necessary for multi-agent SLAM.

- Chapter 3: Field trials to characterize the relative performance of WHOI Micro-Modem data rates.

- Chapter 4: Development of a simulation environment testbed for testing adaptive communications algorithms.

- Chapter 5: Design and performance comparison of environmentally adaptive communications control algorithms.

- Chapter 6: Conclusions and recommended future work.

CHAPTER 2 LITERATURE REVIEW

This section begins with a brief overview of typical AUV hardware configurations. The state of the art in AUV positioning techniques is reviewed with focus on SLAM systems and in particular multi-agent SLAM implementations. Additionally, the acoustic communications environment and current modem technology are reviewed with focus towards adaptive bandwidth systems.

2.1. AUV HARDWARE

AUV hardware generally consists of a pressure vessel containing control actuators, batteries, power systems, a variety of internal and external sensors, and at least one control computer. Many unique AUV designs exist and continue to be developed both by industry and academia, but typical AUV designs are based on hydrodynamic torpedo shaped pressure vessels such as the Ocean Server IVER2 [6] shown in Figure 1. A main drive motor / propeller provides thrust while servo-mounted fins provide control of yaw, pitch, and roll. Ballast systems adjust buoyancy for depth change. Batteries supply electrical power to all subsystems.

At least one OEM "front seat" computer provides basic closed-loop control; often a multiple computer front-seat / back-seat division of labour is implemented [7]. Software frameworks such as the mission-oriented operating suite (MOOS) [8], robot operating system (ROS) [9], and Orca [10] among others are used to act as middleware between various autonomy, communications, and hardware interface programs required.

RF antennae are top-mounted for use, while surfaced, of communication components such as 802.11 wireless, GPS, or Iridium phones. Acoustic modems are used both for localization via ranging and also for communications. A wide range of acoustic and non-acoustic sensors can be implemented for pose and environment measurement as seen in Table 1 on page 6. Section 2.2 provides details on typical AUV sensors and how they are integrated to form position estimates.

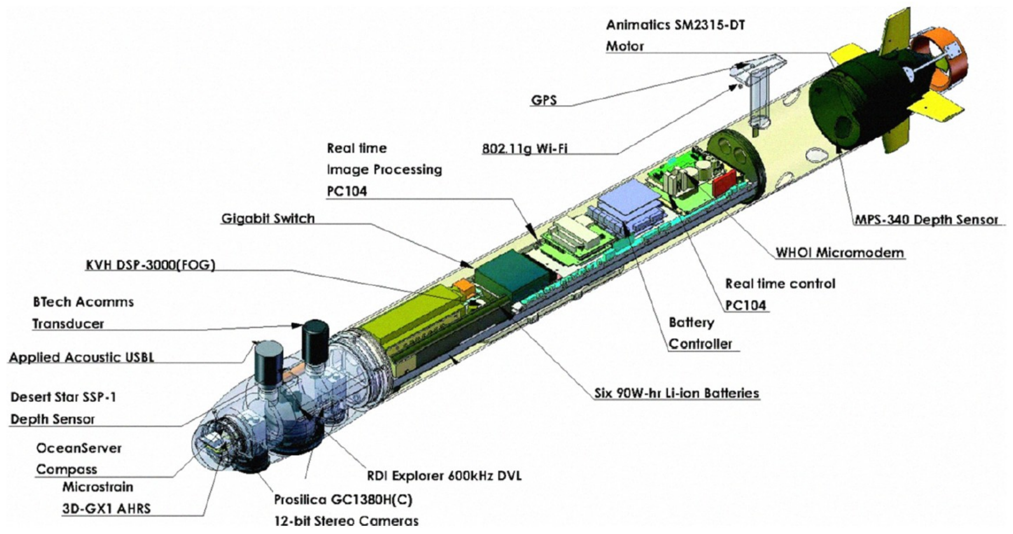


Figure 1 Example AUV physical hardware and layout (modified Ocean Server IVER2) [11]

2.2. AUV POSITIONING

A robot's pose is defined as the combination of its position (x,y,z) and orientation (yaw, pitch, roll) in an inertial reference frame. AUVs, like most mobile robots, must be able to estimate their pose in order to effectively interact with their environment. While the satellite-based Global Positioning System (GPS) is available to AUVs when surfaced, these high-frequency radio signals suffer rapid attenuation in water and provide poor or no position fix while submerged [12]. Thus, AUVs must resort to alternative methods to maintain localization while submerged. Sensors commonly used to measure AUV pose are listed in Table 1, adapted from [12][2]. Localization approaches may be broadly divided into three categories. Dead reckoning tracks motion commands and can be used stand-alone or be aided by a Doppler Velocity Log (DVL) or Inertial Measurement Unit (IMU). Acoustic ranging exploits the time of flight for acoustic waves from a source of known location. Geophysical or terrain-based techniques, in particular SLAM, use environment-detection sensors such as sonar to localize the AUV with respect to landmarks in its surroundings.

Table 1: Commonly used AUV sensors for pose measurement [12][2]

	Device	Measurement Taken	Application	Strengths	Weaknesses
Non-acoustic	Conductivity, Temperature, and Depth (CTD) Sensor	• Water salinity • Water temperature • Depth	• Creating sound velocity vs depth profiles • Tracking ocean salinity and temperature	• High precision • High update rate	• Large size
	Inertial Measurement Unit (IMU)	• Acceleration • Attitude • Heading	• AUV orientation estimation	• High update rate • High precision	
	Gyrocompass	• Heading	• AUV orientation estimation	• Unaffected by metallic objects • Points north even in arctic operations	• Expensive
	Pressure Sensor	• Depth	• AUV Depth estimation	• Inexpensive • Compact • Precise	
	Inertial Navigation	• Position and orientation using integration of	• AUV pose estimation	• High update rate • High precision	• Unbounded position error

	Device	Measurement Taken	Application	Strengths	Weaknesses
	System (INS)	IMU data			
	Global Positioning System (GPS)	• Position	• AUV position estimation	• Bounded error • Precise	• Unavailable while submerged
	Doppler Velocity Log (DVL)	• Speed over seabed • Altitude from seabed	• AUV Navigation	• Accurate • One of few AUV speed measurement tools	
acoustic	Pencil Beam Scanning Sonar / Echo Sounder	• Range to object using a single pencil sonar beam	• Profiling • Point measurements of bottom depth	• Small	• Accuracy depends on AUV attitude • Resolution inversely proportional to frequency
	Multi-Beam / Bathymetric Sonar	• Range to object using multiple beams simultaneously	• Creation of 3D seafloor images • Underwater SLAM	• More time efficient than single pencil beam sonar	• Accuracy depends on AUV attitude • Resolution inversely proportional to frequency

	Device	Measurement Taken	Application	Strengths	Weaknesses
	Acoustic Doppler Current Profiler (ADCP)	• Profile of water current	• Measuring current profile • Aiding INS AUV navigation		• Resolution inversely proportional to frequency
	Forward Looking Sonar	• Range to object using multiple forward looking beams	• Underwater navigation • Target detection • Obstacle avoidance		
	Side Scan Sonar	• Multiple beams that measure intensity of returns to create a 2D image	• Large area 2D scans to search and detect targets within an area	• Works at relatively high vehicle speeds (10kt)	• Resolution inversely proportional to range
	Synthetic Aperture Sonar (SAS)	• Coherent processing of consecutive displaced returns to create a virtual array / antenna	• Very large area 2D scans to search and detect targets within an area.S	• Very high resolution (cms) • Large range (100s of meters) • Good target classification • Wide dynamic range and high SNR	• Works only under tightly prescribed speed and path • Works only at low speeds (~3 knots)

2.2.1. *Dead Reckoning*

Dead reckoning estimates the AUV's current position by performing a vector sum of distance and heading travelled since the previous position estimate [2]. Distance travelled is obtained from combining actuator control motions with the measurements of speed-over-ground sensors such as the DVL, inertial orientation sensors such as an IMU or gyroscope, and heading sensors such as a magnetic compass or north seeking gyroscope. High end systems feature an Inertial Navigation System (INS), which integrates the IMU sensor outputs to extrapolate inertial frame position in six axes [2]. A navigation-grade INS could be a significant portion of the AUV price before payload (>$100k) [13], limiting their practicality in multiple-vehicle operations [14]. All sensors used in dead reckoning have high update rates and precision but, with the exception of compasses, suffer from unbounded error growth. This causes dead-reckon guided AUVs to suffer positional drift and requires repeated surfacing to reacquire GPS position and thus zero their position error [2].

2.2.2. *Acoustic Ranging*

First developed in the 1960s, acoustic ranging is based on the principle that an acoustic transceiver with knowledge of the local sound speed can use the Two Way Travel Time (TWTT) of an acoustic pulse to calculate the distance between source and receiver using time-of-flight. If the transceiver detects pings from three or more known-position sources, it can triangulate its position (known as range-range navigation) [15]. Alternatively, a multiple hydrophone phased receiver array can be used to detect both range and bearing to a single source (range-bearing navigation) [15].

Long Baseline (LBL) acoustic ranging is a widely used AUV positioning technique. LBL requires an array of submerged seabed transponders be deployed in known locations. The vehicle queries each transponder and uses the received reply to calculate the distance using TWTT. Similar in principle, Short or Ultra-Short Baseline (SBL/USBL) features multiple transceivers mounted in close proximity; often on a ship [16].

An emerging ranging strategy is One Way Travel Time (OWTT). Similar in principle to LBL, this approach requires all transceivers have synchronized and very accurate internal clocks and transmit pulses on a predetermined schedule. This allows for faster, more accurate ranging [2][17][18]. Current research focus is on the implementation of OWTT as an inter-AUV fleet positioning tool compared to the historic static baseline transponder methods [19].

Acoustic ranging techniques can provide bounded error, sub-meter accuracy position estimations. However, they suffer from slow update rates, outliers, and random drop-outs. Additionally, the logistics of deploying a wide-area LBL transponder net are nontrivial and positional updates are limited within the deployed area [2].

2.2.3. GEOPHYSICAL LOCALIZATION

Geophysical Localization is a method by which a mobile robot can localize itself in an environment using landmarks as reference points. This can be done using an *a priori* map given to the robot prior to mission commencement, or by building a map in situ as the robot explores. The latter process is known as Concurrent Mapping and Localization (CML) or more commonly SLAM. Various sensors have been used to accomplish AUV geophysical feature detection such as imaging sonar, sidescan sonar, bathymetric multi-beam sonar, and optical cameras [20][21][22][23]. AUV geophysical navigation employs state estimation and does not yield more accurate position estimates than acoustic triangulation methods such as LBL. It is used because it can provide bounded-error position estimates without requiring external references like a transponder network [2]. SLAM is particularly versatile because it does not require an *a priori* information map of an area; an AUV can be submerged in an unmapped area and generate bounded-error position estimates [24]. This makes it appealing for terrain-exploration missions such as bathymetric surveys, mine hunting, and seafloor photography [25][26][23]. SLAM algorithms, capabilities, and current implementations are discussed in detail in Section 2.2.4.

2.2.4. SLAM

Introduced in its current form in the mid 1990s, the SLAM problem requires that a mobile robot placed at an arbitrary location in an unknown environment be able to simultaneously build a map of its environment while localizing itself within that map [27][28]. Such ability is of inestimable value to robotics development in applications where prior maps are unavailable such as disaster relief, construction, and undersea and planetary exploration [29]. SLAM research interest continues to grow with the literature reporting on a wide variety of successful implementations [27]. SLAM can be made more efficient for large-scale outdoor applications by having multiple robots integrate their measurements in situ to build a common map [30].

The SLAM problem was first applied to, and solved for, single robot applications. This review focuses on underwater SLAM solutions that have been tested on real-world datasets. Extended Kalman Filter SLAM is the original and most basic solution, but its real-world implementation has had limited success as complexity increases with the square of landmarks and it is difficult for current sonar based feature-extraction and data association methods to robustly function in unstructured marine environments [27] [31]. Particle filters solutions can reduce computational complexity and directly represent nonlinear systems [32]. The pose graph method goes farther in mitigating feature extraction issues by tracking robot poses using a comparison of perceptual data rather than extracting any discrete features. It has been successfully applied to a variety of underwater map-generation scenarios [33][34].

SLAM assumes a robot must generate a map (m) and localize itself within it (x_t) using only the set of measurements and controls from the first to the most recent at time t: ($z_{1:t}$) and ($u_{1:t}$) respectively. Figure 2, taken from Durrant-Whyte and Bailey's excellent introduction to SLAM [27], visualizes the classic SLAM scenario: a simultaneous estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations.

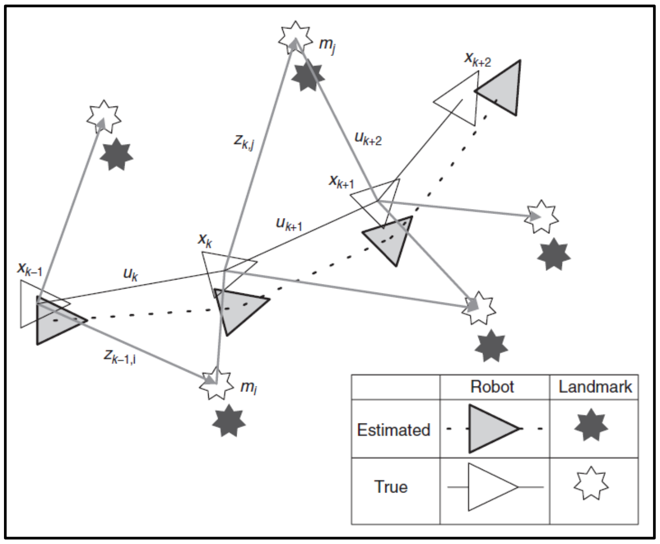


Figure 2. The essential SLAM problem [27]

SLAM is formulated as a probabilistic problem that calculates the posterior probability density function either over the entire robot trajectory $p(x_{1:t}, m \,|z_{1:t}\,, u_{1:t})$ or only over the current pose $p(x_t, m \,|z_{1:t}\,, u_{1:t})$ [35]. This review considers SLAM as a tool for real-time vehicle localization, so the focus will be on the less computationally intensive current pose calculation. This problem can be translated into a standard recursive predict-correct form:

Predict:

$$p(x_t, m \,|z_{1:t-1}\,, u_{1:t}) = \int p(z_{1:t}\,, u_t)\; p(x_{t-1}, m \,|z_{1:t-1}\,, u_{1:t-1}) \qquad \textbf{(1)}$$

Correct:

$$p(x_t, m \,|z_{1:t}\,, u_{1:t}) = \frac{p(z_t|\, x_t, m)p(x_t, m|z_{1:t-1}\,, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} \qquad \textbf{(2)}$$

Solving this problem requires an observation $p(x_t, m \,|z_{1:t}\,, u_{1:t})$ and motion model $p(x_t, m \,|z_{1:t-1}\,, u_{1:t})$ that allow for efficient computation of the predict/correct steps above. The most common model used is a state space with additive Gaussian noise, with the EFK being the most popular solution method [35]. This approach is inherently limited to phenomena that are Gaussian. Durrant-Whyte presents the complete algorithms in [27]. Like all successful SLAM solutions, the EKF method is monotonically convergent with the landmark relationships converging to zero and the individual landmark positions converging to a lower bound based on observation error.

One major drawback to the EKF solution is that it is feature based, storing all point landmarks and associations. Native Kalman Filter implementation yields $O(n^3)$ update complexity, which can be reduced to $O(n^2)$ in SLAM from the sparseness of typical robot environments [36]. Still, as the number of features grows the computation costs quickly become untenable. One possible mitigation for this is the use of submaps [37][38] where multiple, bounded-feature maps are created, transforms between submaps are tracked either relatively or globally, and then submaps are fused. This presents the requirement for map fusion which is another active area of research. However, even given this improvement, basic EKF SLAM becomes impractical for use in large, feature-dense situations. Conversely, SLAM convergence becomes problematic in feature-poor environments which can occur underwater.

An EKF-based SLAM approach was first successfully implemented underwater in 1999. Newman [39] operated the custom-built Oberon AUV within a pool environment with sonar beacons for landmark features in order to simplify data association and feature detection. In that same year, Leonard *et al.* [40] performed similar in-pool EKF based experiments using a forward looking sonar, achieving bounded-error results. He also used ship-mounted AUV sensors (sonar, INS, DVL) to attempt natural feature extraction as a first step in underwater SLAM. By 2000, Williams et. al. [24] graduated to small-scale underwater SLAM tests using the Oberon off the coast of Sydney, Australia, albeit again with sonar beacons for unambiguous landmark features. Newman and Leonard [41] applied EKF SLAM with mixed success to range-only LBL transponder data gathered in 2002 on their Odyssey III AUV. In 2004 Mahon and Williams [42] used EKF SLAM to integrate pencil sonar and visual camera data collected on the feature rich Great Barrier

Reef environment. While their process was able to create environmental models, it had difficulties with the longer-term feature association required for successful loop closure.

Loop closure is a key aspect of SLAM whereby a robot re-visits an previously observed feature. By correctly associating the feature as being previously observed, rather than a newly discovered feature, the position of the robot over its whole trajectory is corrected (improved)[43]. Ruiz et. al. [21] also applied an EKF based SLAM algorithm to sonar data in 2004, but were forced to manually locate features from the data prior to SLAM implementation. Beginning in 2006, Ribas *et al.* have run successful tests [20][44] using a mechanical imaging sonar and modified EKF SLAM to detect wall-shaped landmarks in structured environments such as marinas, as compared to the previous researchers' focus on point features. Koh *et al.* [45] present a 2009 test with EKF SLAM running online during sea trials with a Meredith AUV. However, artificial sonar reflectors were used as features.

Overall, EKF SLAM is the first variety of SLAM to be implemented online or on real-world underwater environments. Implementations have been characterized by inadequate natural feature detection from sonar data, often requiring researchers to use artificial sonar reflectors for features. EKF performs acceptably well for these controlled scenarios, but underwater research trends are moving toward SLAM variants more capable of handling feature-rich environments. A particular area of current interest is in implementations of SLAM that scale well to multiple vehicle scenarios; this application will be the focus of Section 2.2.5.

2.2.5. MULTI-VEHICLE SLAM

As electronics miniaturization continues to decrease the cost of robot hardware [6] and embedded systems become increasingly powerful, research is increasingly focusing on using multiple robots simultaneously to collaboratively or cooperatively accomplish large-scale missions. While multi-vehicle operations inherently increase mission complexity, operational overhead, and capital costs, they also provide the possibility for significant benefits. Area coverage per time is increased [46], a critical requirement for

NMCM operations: the United States Navy Unmanned Underwater Vehicle (UUV) master plan requires clearance of a 100-900 nautical square mile area in 7-10 days [26]. In addition, collaboration improves operational redundancy by providing automated vehicle replacement [47] and allows the possibility of heterogeneous groups where specialized robots accomplish different tasks within an overall mission framework. This could be used to increase the fleet capabilities by using AUVs with narrow, high resolution and broad, low resolution sensors to perform both acquisition and identification[46] [48] or to decrease cost by having one AUV with an expensive localization sensor suite provide localization for several low-budget vehicles [11][49].

Current SLAM efforts are beginning to move from single-vehicle applications to the development of capable multi-robot algorithms [11][50]. This section summarizes the basis of collaborative SLAM and highlights key implementations from the ground and aerial robotics communities. Unfortunately, communication bandwidth restrictions and high attenuation make underwater collaborative behaviours challenging [11]; most online multi-robot underwater SLAM implementations are simulated due to the difficulty of real life implementation. Proposed and simulated underwater collaborative algorithms will be reviewed with a focus on the barriers preventing actual implementation.

Single vehicle SLAM requires that a robot revisit previously acquired features or robot poses to provide "loop closure" by returning to known spaces [51]. Loop closure decreases the robot's location covariance which otherwise grows proportional to map size. Multi-vehicle SLAM offers the possibility of sharing observations between robots to generate loop closure even if a particular vehicle has never 'backtracked', substantially improving the positional accuracy of all vehicles involved. The lower positional error bound possible with multi-vehicle SLAM is lower than the single vehicle initial covariance which bounds the single-vehicle case [30]. Additionally, in a heterogeneous scenario, vehicles with few or low quality sensors may obtain good position estimates using the measurements taken by collaborating vehicles with high-quality sensors, reducing overall system cost [49].

In 2000 Nettelton *et al.* [51] introduced one of the first investigations of cooperative SLAM, presenting a closed form solution in both the standard state space and information forms for a 1D simulation. They conclude that the information form is best suited for collaborative SLAM: being agnostic to chronological order of updates allows for a simpler decentralized implementation. This view is shared by the robotics community, with the majority of research efforts focusing on the information form.

Notable exceptions include Leonard *et al.* who performed indoor ground multi-vehicle experiments in 2002 [30] and 2004 [52] using the state space form of EKF SLAM. The former experiment involved the distributed integration of each vehicle's complete state estimate vector to form a distributed state vector, and the latter was a post-processed centralized implementation where a master robot integrated all measurement data from all robots into a modified single vehicle algorithm. Neither of these implementations considered communications constraints or latencies; they would in large area scenarios require prohibitive communications bandwidth and be vulnerable to latency.

In contrast, Thrun *et al.* have focused on applying their Sparse Extended Information Filter (SEIF) to multi-vehicle mapping. They applied the multi-vehicle SEIF in 2003 [53] to the Victoria Park SLAM dataset (partitioned to simulate multiple vehicles), and in 2004 [54] to a simulated scenario involving navigation of three UAV aircraft. In 2006 Howard [55] performed an encounter-based multi-robot particle filter SLAM experiment where vehicles performed individual SLAM until physically sighting another robot, at which point each would integrate the other's map into their own, providing loop closure. In 2006 Ong *et al.* [56] applied a particle filter approach to a two UAV dataset.

While the experiments presented have been conducted via post-processing of datasets rather than online, the datasets were taken from real world robot experiments. The information and particle filter forms presented by Thrun *et al.* , Howard, and Ong *et al.* require bandwidth not exceeding the limits of modern broadband RF. Thus, while still an emerging field, successful solutions to multi-robot SLAM exist for ground and aerial applications.

Although AUVs have difficulty with feature recognition and data association, the lack of online collaborative SLAM is primarily due to limited communication bandwidths. A survey of simulated collaborative AUV SLAM algorithms is presented; these typically assume unrealistically high acoustic modem throughput and 100% connectivity between vehicles. For reference, researchers who implement real-world collaborative AUV behaviours report acoustic data rates ranging from 80 bits per second (bps) (low) to 5kbps (high) [11]. As a result, typical real-world communication protocols conservatively assume approximately 32 byte packages sent every 6-10 seconds [48][57].

Diosdado and Ruiz [58] present a SEIF collaborative SLAM solution with simulated AUV testing. While it can deal with arbitrary communications latencies, it assumes sufficient bandwidth for all AUVs to transmit all detected feature locations, which would require significantly more than the abovementioned data rates.

Nettleton *et al.* [59] present an improved multi-vehicle SLAM algorithm using a decentralized EIF with constant duration communication cycles achieved by having each robot only transmit the most important not yet transmitted feature locations every communications cycle. It does not require all feature locations be transmitted. His simulated implementation assumes each robot transmits five feature locations and their corresponding covariance matrix every 10 seconds, achieving errors of only twice those incurred with complete feature set updates. While this is an improvement over [58], real-world AUV implementations are likely to increase the cycle time by an order of magnitude which would reduce effectiveness unacceptably.

Pfingsthorn *et al.* [60] present a multi-AUV pose graph based photo mosaiking algorithm which they apply to a real-world photographic dataset. They are careful to address typical acoustic modem capabilities, assuming a data rate of 4,800 bit/s; the minimum required rate of their algorithm for complete execution (loop closure) is 212 bytes per second per robot. These rates are at the upper limit of typical modem performance today under ideal acoustic propagation conditions, and not representative of actual rates (80 bps) used for collaborative behaviour [48][57].

In summary, the lack of successful solutions for multi-AUV SLAM under realistic operating conditions is primarily due to the harsh bandwidth and attenuation constraints imposed by the underwater acoustic channel. Eustice *et al.* explicitly point out in [11] that collaborative AUV research at the University of Michigan is avoiding implementation of complete multi-vehicle SLAM due to the extreme acoustic channel bandwidth restrictions. To gain a better understanding of the reasons for the bandwidth restrictions, underwater acoustic communications are reviewed in Section 2.3.

2.3. UNDERWATER COMMUNICATIONS

AUVs are differentiated from remote operated vehicles (ROVs) in that they are by definition free-swimming. While this allows far greater range of motion, it necessitates wireless communications for transmission of control instructions and data feedback. A variety of wireless communication channels are available underwater, with the acoustic being most effective for AUV applications.

Electro-magnetic (EM) waves propagate poorly because the conductivity of salt water causes absorption of radio frequency (RF) waves that is two orders of magnitude higher than in fresh water. Additionally, the absorption is proportional to the root of frequency, meaning high-frequency RF waves (10s-100s MHz) suffer rapid attenuation [2]. [61] has proposed the use of low-frequency (1-3 KHz) magneto-inductive (MI) magnetic communications to provide reliable communications in complicated mediums such as the very shallow water (VSW) and surf zones (SZ) where NMCM operations are sometimes conducted. While these can provide reliable 100-300 bps data rates at ranges up to 400m using very compact receivers, range is proportional to the cube of field strength generated by the transmitter. High power and large heavy antennas are required to achieve appreciable range, rendering MI viable only as a one-way ship to AUV communication tool.

Optical communications also suffer poor range as water attenuates light three orders of magnitude more quickly than does air. Additionally, the presence of suspensions and ambient light in shallow water can further decrease effectiveness of optical communications due to scattering. In good conditions optical systems can achieve ranges of 10-100m [2].

Acoustic communications can provide far superior ranges (10 km in ideal to < 1 km in poor conditions [62]) compared to EM and optical systems. The major drawbacks are a propagation speed five orders of magnitude and frequencies three orders of magnitude less than EM / optical, yielding associated increased cycle time and decreased bandwidth [2]. Thus, while EM / optical methods may be optimal for a closely spaced static underwater network, the potential range available to acoustic signals makes it the method of choice for mobile AUV communications.

The following sections will provide background on the underwater acoustic channel, including environmental effects on channel performance and the significance of the channel impulse response. The strengths and weaknesses of basic modem modulation schemes will be detailed along with a summary of channel access schemes employed. Finally, the environmental adaptivity of current communications systems is surveyed: high-rate coherent modems contain adaptive feedback equalizers that use the instantaneous impulse response to demodulate colliding symbol inputs, but collaborating AUV groups do not currently adapt their communications cycles based on spatial or environmental conditions.

2.3.1. *Acoustic Channel*

The acoustic channel is characterized by the following properties. Transmissions suffer geometric spreading losses proportional to the wave front area. In deep water this is approximated by an expanding sphere (proportional to square of distance), but in shallow water the wavefront quickly reaches the sea bottom and free surface, reflecting horizontally to approximate an expanding cylinder (proportional to distance) [2]. Transmissions also suffer absorption loss by which the wave energy is converted to heat.

These losses are represented by $e^{-\alpha(f)R}$ where R is the range or transmitted distance and $\alpha(f)$ is the absorption coefficient which is proportional to frequency squared. As shown in Figure 3 from [63], absorption losses set a practical upper limit on usable frequencies where > 50 KHz acoustic waves propagate for only short ranges [2]; for moderate range communications 20-50 KHz is typically used [2].

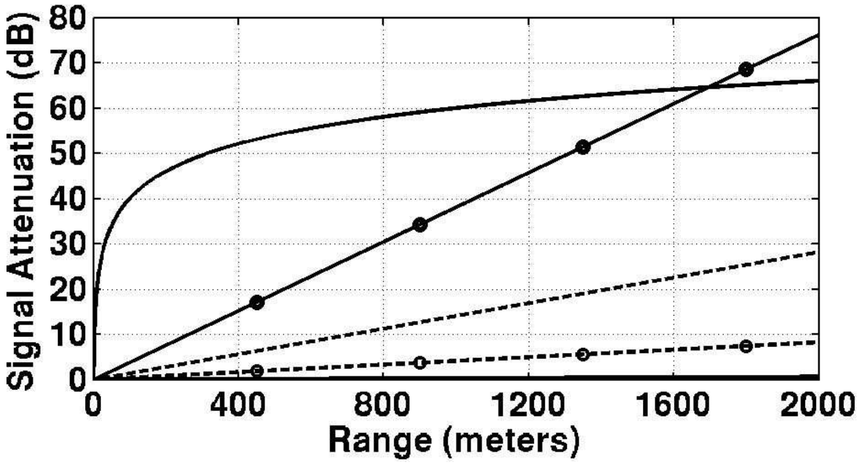


Figure 3: Acoustic signal attenuation as a function of range in seawater relative to the attenuation at a distance of 1 meter from the source [63]. Spherical spreading loss (upper solid line) and absorption losses 5 kHz (lower solid line), 25 kHz (dashed line with circles), 50 kHz (dashed line), and 100 kHz (solid line with circles).

Transmission speeds are on the order of 1500 m/s in water . This relatively slow speed means typical AUV speeds of 2m/s in conjunction with a standard modem frequency of 25kHz would produce a 33Hz Doppler shift, increasing the apparent channel fluctuation [2]. Additionally, the slow speed means that multi-path bounces can be delayed by 80-100ms [63] which far exceeds the single symbol transmission period, leading to significant intersymbol interference (ISI) for high datarate transmissions [62]. Finally, the saltwater sound velocity is not constant, but rather varies with temperature, density (depth), and salinity. The differences in temperature and depth create a sound velocity versus depth profile that varies widely based on location, temperature, depth, and salinity. Particularly important is that the profile often does not monotonically increase or decrease, but rather can peak within the water column as shown in Figure 4 (c), taken from [64]. Acoustic transmissions travelling at sufficiently shallow angles can refract along these peaks, causing either data loss or localization transponder signal distortion [62].

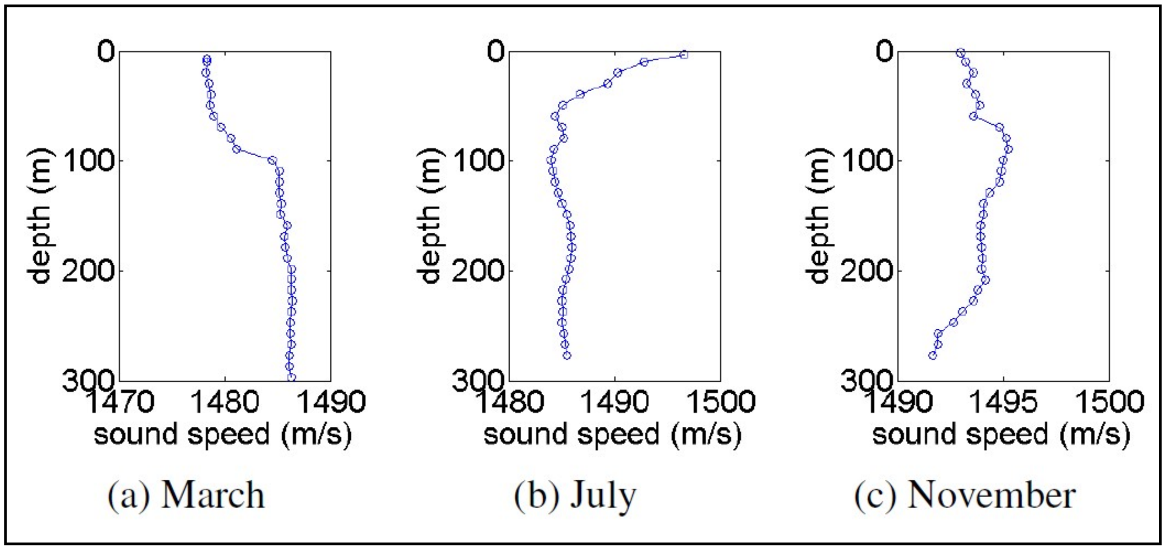


Figure 4: Sound velocity profiles [64]

The underwater acoustic channel suffers from potentially rapid fluctuations; change can happen on the order of 40ms [63]. This means that channel state estimation employed by the transmitter could be obsolete before it is even received. In addition to rapid small-scale fluctuations, the channel also suffers from longer-scale fluctuations whereby location, temperature, or salinity changes may drastically affect communications performance on an hourly or daily basis.

The physical environment has a marked effect on acoustic channel performance. Mid-column deep water environments can provide a stable waveguide with high signal to noise ratio (SNR) and few scattering or multi-path effects. However, the littoral, VSW, and SZs where NMCM operations are conducted can be characterized by low SNR, extensive scattering and multi-path effects, and quickly shifting shadow zones [63].

SNR is characterized by the level of background noise: sources include human (shipboard machinery, seismic surveys, blasting), biological (whale communication), surface (waves, wind, rain), shoreline (crashing waves). These effects are typically more pronounced near the surface and in shallow water zones.

Multi-path effects occur in water shallow enough for a signal to reach the receiver both directly and via at least one reflection from the bottom or surface or refraction due to the SSP. Multi-path arrivals are typically sparse as each arrival is delayed proportional to the distance of its particular path, with paths differentiated by discrete numbers of reflections [65][63].

Scattering causes waves to deviate from their undisturbed trajectory, effectively weakening the signal coherence. Scattering can happen at the surface reflection due to wave motion or bubbles, at the bottom reflection due to topography, and throughout the water column due to suspensions [2].

Finally, shadow zones are caused by the discrete ray path pattern generated by surface / bottom reflections. Figure 5 from [63] demonstrates how reflection from the bottom and surface combine with refraction to form high and low intensity standing wave patterns. Any receiver in a location where ray paths are sparse or non-existent would suffer poor reception. This is particularly applicable to close-range shallow water operations, with minor changes in locations or environment drastically changing the shadow zone shape and thus the communications link reliability [63].

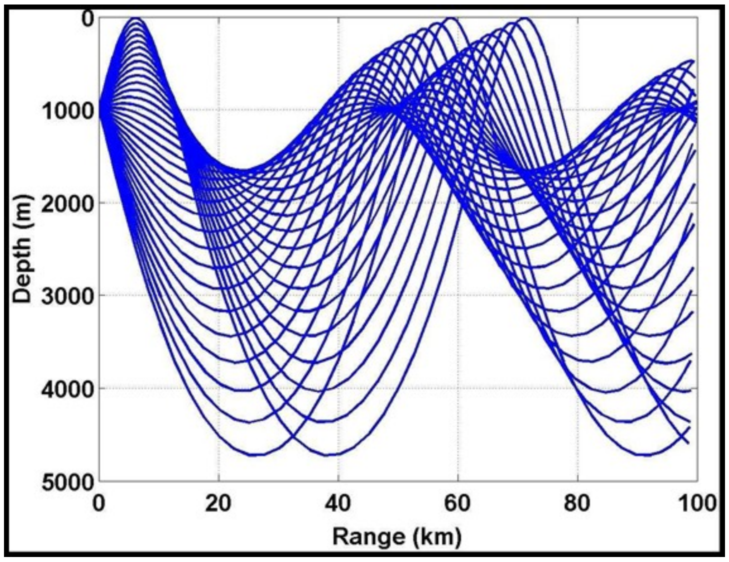


Figure 5: Ray paths demonstrating shadow zone formation [63]

Overall, the VSW zone in which NMCM operations are sometimes conducted is susceptible to a number of time-varying effects that can alternately distort, scatter, or enhance communications throughput.

2.3.2. Acoustic Modems

This review provides an overview of how current underwater modem technology works rather than a summary of the capabilities of existing modems. Thus, hardware examples will be limited to WHOI modems popular within the research community and the Teldyne-Benthos modems used both in research and industrial applications.

Acoustic communications may be broadly differentiated into phase incoherent and phase coherent modulations. Incoherent modulation is the historic industry standard, capable of yielding reliable longer range communications albeit at limited bit rates [65]. Coherent modulation is a rapidly evolving technology, currently providing much higher data rates at the cost of algorithmic and hardware complexity [65] and less reliability in difficult acoustic conditions. State of the art general-purpose modems such as the WHOI Micro-Modems [66] and Tedyne-Benthos ATM 900 [67] have change-on-the-fly settings that allow online selection between a variety of incoherent and coherent data rates.

The first digital acoustic modems, introduced in the late 1970s, employed frequency-shift-keyed (FSK) modulation [65]. FSK modulation alternately shifts a carrier frequency either higher or lower in a pattern to form distinct symbols. Being an incoherent (energy detecting) method, this is robust to the time and frequency channel distortions induced by multi-path and other effects described in section 2.3.1 [62]. Despite relatively low data rates, modern modems such as the WHOI Micro-Modem 1 and 2 [66][17] and the Teledyne-Benthos ATM 900 series [67][68] continue to employ frequency hopping FSK at 80 bps as the most basic option. The ATM 900 also provides a multi-channel M-ary FSK option with rates up to 2400 bps, but this does not work for multiple users. Even though coherent methods are required for higher throughput, the vast majority of literature describing real-life collaborative AUV operations using WHOI Micro-Modems to date exclusively use the basic FSK mode in preference to the modem's coherent capabilities. [3] notes that this is because researchers require communications be reliable first and only then be fast.

As most telemetry applications are bandwidth constrained, a major focus of modern modem design has been bandwidth efficient technology [62]. Phase-shift-keyed (PSK) modulation is a phase coherent (phase detecting) method first implemented in the 1980s for deep water vertical links [65]. Initially thought to be too vulnerable to multi-path and time delay effects for use in horizontal and littoral or VSW environments, FSK was preferred for these acoustically noisy environments. However, advancements in processor and digital signal processing (DSP) technology have allowed the creation of adaptive digital feedback equalizers (DFEs) capable of reconstructing time-scattered signals online [69]. Environmental adaptivity in modern DFEs is more closely investigated in Section 2.3.4. The WHOI Micro-Modem is capable of several PSK rates up to 5kpbs [66], while the Teledyne-Benthos ATM 900 also boasts a variety of rates with the fastest being close to 15kbps [68]. However, the upper limit PSK modes can only be realized under ideal acoustic conditions [68]. In summary, modern acoustic modems incorporate a range of coherent settings that trade off increased speeds for decreased reliability in complex environments. However, these remain largely unexploited in current AUV research.

Stojanovic *et al.* [70] proposed but never implemented a PSK-based AUV network protocol; only Schneider and Schmidt [3] have actually implemented a high-rate PSK mode in real world AUV collaboration experiments.

2.3.3. ACOUSTIC CHANNEL ACCESS PROTOCOLS

A variety of methods exist for sharing a communications channel between multiple users. By far the most prevalent in current AUV collaboration experiments is time division multiple access (TDMA) scheme. TDMA requires a communications cycle be predefined for a group, with every member being allocated a transmit time slot within the cycle [2]. Benefits include simplicity and the ability for all group members to listen to the currently broadcast transmission. The primary drawback is cycle time increases proportionally with group size. Other multiple access protocols include frequency division multiple access (FDMA) in which different users transmit on different frequency bands and code division multiple access (CDMA) in which transmissions are identified by pseudonoise codes [71].

2.3.4. ENVIRONMENTAL ADAPTIVITY

As presented in the preceding sections, not only is the acoustic channel bandwidth limited but its quality also varies environmentally and temporally. To improve performance, underwater communications systems must adapt to exploit the maximum transmission rates supported by the current state of the acoustic channel. Adaptivity can be present both at the modem level and at the AUV system level.

Modern PSK modems incorporate significant environmental adaptivity into their demodulators. While incoherent FSK systems intentionally avoid ISI, PSK modems suffer from potentially hundreds of overlapped symbols and must use adaptive DFEs to reconstruct the original message [65]. Figure 6 from [72] elucidates the structure of two types of DFE: the channel estimate based decision feedback equalizer (CE-DFE) and the linear equalizer. In both cases, the channel impulse response is used to calculate filter weights for symbol extraction. A training period is required prior to an operation, in which the filter weights are updated by algorithms ranging from simple but slow least mean squares (LMS) to the complex but fast recursive least squares (RLS).

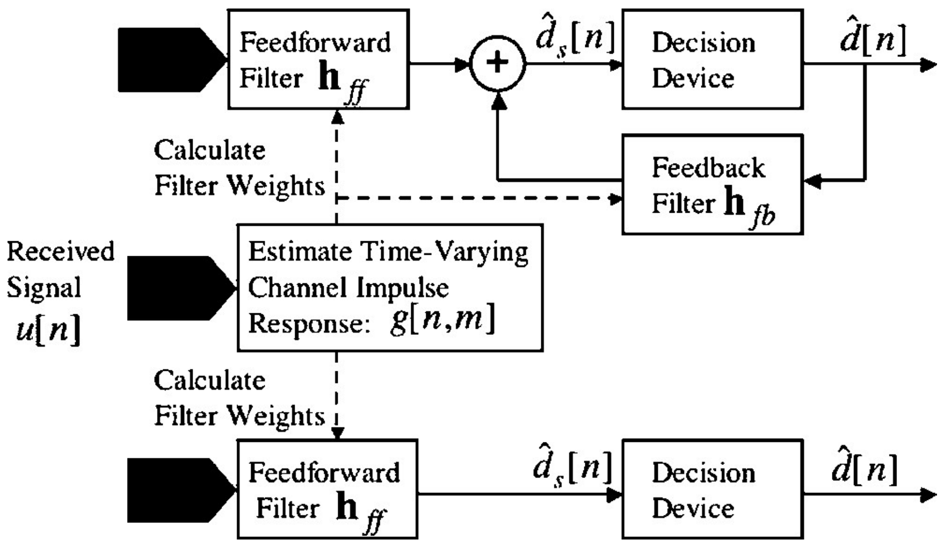

Feedforward Filter $\mathbf{h}_{ff}$
$\hat{d}_s[n]$
Decision Device
$\hat{d}[n]$
Calculate Filter Weights
Feedback Filter $\mathbf{h}_{fb}$
Received Signal $u[n]$
Estimate Time-Varying Channel Impulse Response: $g[n,m]$
Calculate Filter Weights
Feedforward Filter $\mathbf{h}_{ff}$
$\hat{d}_s[n]$
Decision Device
$\hat{d}[n]$

Figure 6: The structure of channel estimate based coherent equalizers [72]

The received signal, u[n], is processed to generate estimates of the time-varying impulse response of the channel between the transmitter and each receive hydrophone. The impulse response estimates are used to compute the equalizer filter weights. These filter weights are used to implement the equalizer and estimate the desired data symbol. Two different types of equalizers are shown. The upper equalizer is a channel estimate based decision feedback equalizer CE-DFE and the lower equalizer is a linear equalizer.

Various other DFE-based adaptivity methods currently implemented or under investigation include blind algorithms that do not require prior training sequences [62], sparse equalizers that exploit discrete multi-path arrivals to reduce the number of channel taps [65] and a plurality of methods seeking computational efficiency via replacement of RLS with a simpler algorithm [62].

While PSK modems depend on real-time channel adaptivity within the demodulator, this is used to deinterlace incident symbols and improve the signal's bit error count. It is not used to adapt the modem between the modem's range of PSK data rate settings; these are user-specified and must be controlled at an AUV system level. In contrast to modem-level environmental adaptivity, the author cannot find any examples of experimentally implemented adaptive AUV communications where modem speed or transmit/no transmit conditions are modified to adapt to the current environmental conditions. In fact, with few exceptions, practical implementations of collaborative AUV manoeuvres still use nonadaptive incoherent FSK modem settings for communications. Schneider and Schmidt [3] cycle high and low speed settings, emphasizing the importance of reliable message reception. Shahabudeen *et al.* [73] propose a multi-channel AUV network approach by which three modems of different frequencies are installed on each AUV to optimize communications based on inter-vehicle ranges. This has not been implemented and the physical installation of three modem systems in each AUV could prove power and space intensive. In Chitre *et al.* [69]'s 2008 comprehensive survey of underwater networking, AUV networks are described as being in their infancy. While several dynamic network protocols and topologies are described that can potentially adapt data paths based on link quality, these are focused on static networks with no mention of any implementation in mobile AUV scenarios.

The slow adoption of high data rate coherent modems in collaborative AUV operations is driven by reliability concerns. Significant bandwidth improvements without decreased reliability could be possible if AUVs adapt their communication settings to the limits of their current environment, rather than simply assuming the worst channel possible and using slow but reliable incoherent settings.

2.3.5. *ADAPTIVE AUV CONTROL*

Section 2.2.5 concluded that collaborative underwater SLAM has not been successfully implemented to date due primarily to acoustic bandwidth and attenuation restrictions. Section 2.3.4 found that while acoustic channel capacity varies widely with environment, current AUV protocols sacrifice significant bandwidth by assuming worst-case capacity and transmitting slowly at all times to ensure reliability. Bandwidth during collaborative AUV operations could potentially be increased without loss of reliability by adapting modem speeds online to match the available channel capacity. This would require a shift from the currently employed deliberative architecture protocols to a hybrid or behaviour-based architecture. This section investigates adaptive control architectures currently implemented on AUVs as possibilities for adaptive communications protocols. Many of these make use of machine learning techniques when deterministic state models are complex or unavailable.

Historically, AUV control was first provided by deliberative architectures that caused vehicles to follow a predefined series of actions (such as waypoint following) according to an *a priori* mission plan. Sensor input was used for data collection but typically did not factor into behavioural decisions [74][2]. This approach is simple but does not allow the AUV to react to unplanned situations or adapt to optimize behaviour for the current environment. There is a growing focus on moving from pre-planned to adaptive architectures [74][75][76][77]. Behaviour-based or reactive architectures eliminate structured mission plans in favour of action directly based on timely sensor input. Hybrid architectures attempt to integrate the best aspects of the two by, for example, combining a deliberative planning layer with a functional reactive layer [78]. An AUV has two ways it can act: motion and communication. The action outputs of current adaptive AUV algorithms almost universally act on AUV motion (i.e. path planning); a notable exception being the fault-diagnosis algorithm developed by [76]. However, similar principles can be applied to act on communication settings instead of motion. The following are various methods by which behavioural architectures can react to their environments.

A cost function that compares a set of discretized actions available to the vehicle against their expected cost is used by Seto and Hudson [79] to implement an adaptive path planner for a communications and navigation aid (CNA). While similar to the cost / reward aspect of reinforcement learning, this application was simple enough that the set of actions and cost function could be predefined. This is not ideal for adaptive communications because no similar closed-form cost function exists for modem data rate settings.

Genetic algorithms (GAs) are used by Seto for online path planning based on remaining vehicle energy reserves [80][77] and for recalculation of control surface movement in case one control surface unexpectedly becomes underactuated [81]. Based on an iterative mutate-verify cycle, GAs are appropriate where potentially complex functions (i.e. motion paths or control gains) must be tailored to optimize certain output parameters (i.e. surveyed area given available energy). They are not ideal for adaptive communications that require selection of states (discrete modem speed settings) which have no method of closed-form performance parameter estimation.

An information gain with branch entropy is used by Paull *et al.* [82] to path plan online for complete area coverage using sidescan sonars. This approach decomposes the area to be covered into hexagonal grid cells to make it able to apply a form of online A* path planning to ensure complete coverage. Again, while this approach works well for a path planning behaviour, it would be ill suited to communication rate-changing behaviours as they cannot be broken into a grid or tree to which search techniques may be applied.

Neural networks have been applied to a variety of AUV behavioural control problems. A form of supervised machine learning, they are typically trained offline using data consisting of desired input/output combinations. Originally developed for classification, they can now be used for regression and function estimation and can provide good control of complex nonlinear systems where an explicit motion model is difficult to derive [76]. In the early 1990s Sutton *et. al* [83] explore use of neural networks to maintain AUV depth control. Antonelli *et al.* [76] apply a support vector machine (SVM) neural network to diagnose potential AUV actuator faults online by operating a learned diagnostic observer in parallel with the observed AUV motion to detect differences. Guerrero-Gonzalez *et. al* [75] present a particularly interesting implementation of dual neural networks for AUV navigation. The first is a self-organization direction mapping network (SODMN) which is trained online to build the vehicle motion model through detecting the effects of random control inputs. The second is neural network for avoidance behaviour (NNAB) which, trained completely online to avoid obstacles, does not even require prior knowledge of robot geometry or sensor quality. Implementations of these algorithms were successfully tested both in a pool and lagoon setting. Neural networks present much more promise for adaptive communications behaviour than the previously presented methods: once trained they are well suited to take a large number of inputs and present an optimal output state.

Finally, reinforcement learning (RL) is also commonly applied to AUV behaviour control. Reinforcement learning teaches an optimal policy for state-space navigation through a cost/reward feedback. Carreras *et. al* [84] successfully apply the Q_learning form of RL to behaviour selection for an AUV that uses different behaviours to visually track a target while avoiding obstacles. Kawano and Ura [85] propose a Q_learning method to maintain AUV vertical ascent trajectory during unexpectedly severe current conditions. RL is another method of adapting AUV behaviour that shows good promise for communications state selection as it allows for online adaptivity to situations that it has never previously trained for.

In summary, behavioural architecture is receiving significant research attention with a multitude of behaviour selection methods proposed to address AUV outputs such as path planning, control surface and thrusters actuation, and self-diagnosis. No proposed communications bandwidth selection methods have been found, but the methods used for motion selection could be applied.

The next step to developing an adaptive communication control behaviour is to characterize the relative reliability of the available WHOI Micro-Modem data rates. This characterization, detailed in Section 3, will provide necessary information for wise decision making.

CHAPTER 3 MODEM CHARACTERIZATION

To develop communication algorithms that optimally adapt the data rate at which a modem communicates, the relative reliability of the data rates must first be characterized. The WHOI Micro-Modem, like other acoustic modems, offers a variety of possible data transmission rates as listed in Table 2. Note that the packet transmit duration of the data rates are comparable, meaning the baud rate scales with the packet size.

Table 2: WHOI Micro-Modem data rate comparison [86]

WHOI Micro-Modem Data Rate	Frame Size (Bytes)	Number of Frames per Packet	Packet Size (Bytes)	Packet Length (s)
FSK Rate 0	32	1	32	3.90
PSK Rate 1	64	3	192	3.08
PSK Rate 4	256	2	512	3.15
PSK Rate 5	256	8	2048	3.04

While it is understood that lower bandwidth transmissions provide a more robust link than high bandwidth rates [3], there has been little work done to characterize the relative probability of packet success for the WHOI data rates in real conditions. In the most comprehensive work available, Pompili's laboratory at Rutgers used two Micro-Modem hardware units that communicated through a software simulated environmental interface to compare frame error rates to SNR [5]. While this is the type of relative rate characterization required for intelligent adaptive rate control, the Rutgers results are not applicable because only deep-water cases were studied and no field experiment validation was presented. Shallow water field trials are required to characterize the relative performance between data rates in representative NMCM conditions.

3.1. EXPERIMENTAL DESIGN

Communications are implemented between two WHOI Micro-Modems with co-processors installed to allow use of PSK high-speed data rates [86]. One Micro-Modem is located on the deck of the DRDC Acoustic Calibration Barge (ACB) and controlled by a barge-mounted computer. It drives a WHOI 25kHz towfish transducer that is suspended to a depth of 10m. The second Micro-Modem is installed in an IVER2 man portable AUV and controlled by the vehicle's onboard backseat PC. The trial configuration is outlined in Figure 7. The trial location is the Bedford Basin near Halifax, Nova Scotia, Canada. The Basin is a shallow water silt bottom environment with depths ranging from 30-40m in the trials area to 80m maximum.

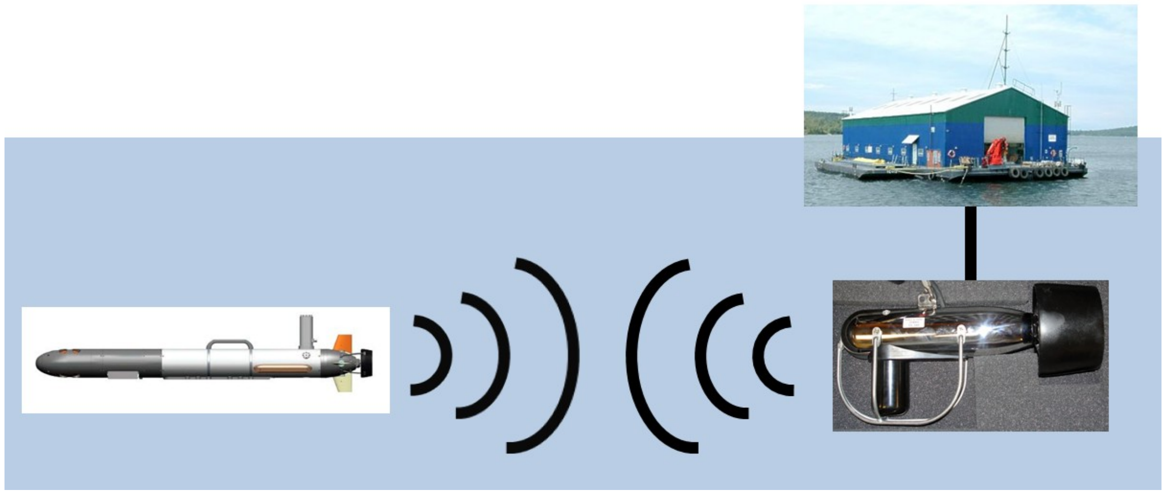


Figure 7: Experimental hardware configuration

The software configurations on the deck and IVER2 computers are identical save for additional live display capabilities integrated into the deck workstation. The computers' autonomy capabilities are built using the MOOS-IVP [8][87] autonomy suite installed in the Ubuntu 10.04 Linux operating system. MOOS provides a modular publish-subscribe framework allowing quick integration of new programs with existing autonomy and sensor packages. The Goby2 acoustic communications package was used to control the modem transmissions. Goby implements the Dynamic Compact Control Language (DCCL) [88] to compress data for transmission over low bandwidth acoustic links.

Two new applications are used for this trial. First, a logger was developed that captures time stamped AUV poses at the moment of modem activity in an easy-to-analyse format. The logger is responsible for assigning outgoing messages unique serial numbers and tracking which are received and which fail. Secondly, a modem rate controller was developed. While Goby2's *pAcommsHandler* does allow the user to choose the rate of transmission, the communications cycle is input during program start and assumed to remain constant throughout the mission duration. A custom interface was designed that allows on-the-fly rate change decisions to be implemented. The software architecture is detailed in Figure 8 where red colour represents applications developed by the author and grey represents external hardware interfaces. Note that this architecture does not include vehicle movement controllers. This trial does not require *in situ* path-planning changes and can therefore be run using a precomputed mission plan. This is accomplished using the manufacturer's path planning software located on the OEM computer.

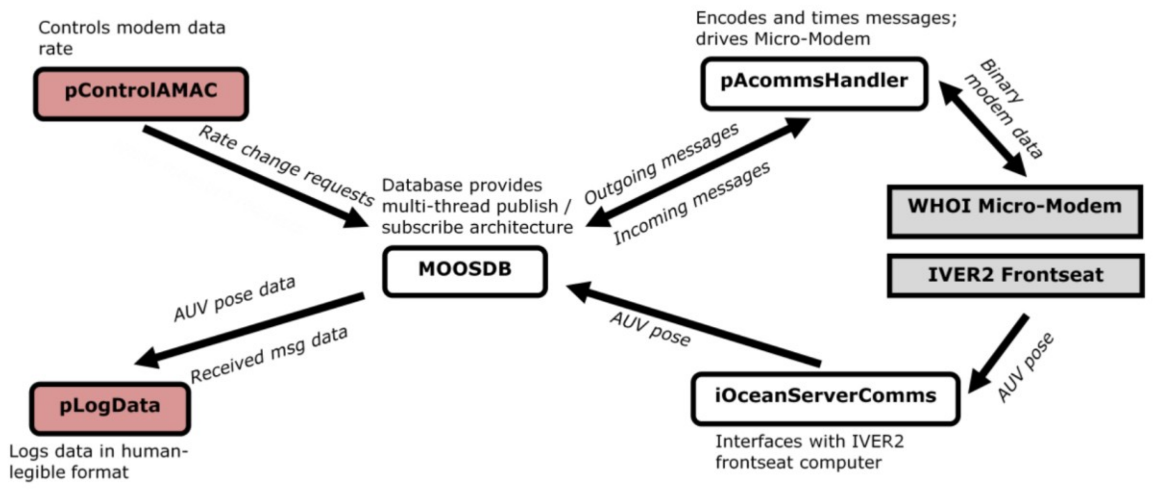


Figure 8: Experimental software architecture.

This architecture was used to implement a simple two-way communications protocol where the modems alternated transmissions and continually cycled between WHOI data rates 0, 1, 4, and 5 (refer to Table 2). Rates 2 and 3 have equivalent packet sizes as rates 1 and 4, but are antiquated and no longer recommended for use [86]. The success or failure of each frame was recorded along with timestamp and AUV pose.

In order to isolate the effect of modem data rate on transmission performance the trial had to hold as many other environmental influences constant as possible. All testing was conducted in a single day: the SSP can be driven to daily change by precipitation or temperature fluctuation, and sea state changes impact ambient noise levels and surface refraction. The IVER2 was programmed to follow a back-and-forth mission path along a line roughly radial in direction from the barge. Constraining the path to a line allows the collection of a large number of sample points over a limited geographical area, further controlling the impact of environmental factors. Having the line radiate from the Barge constrains the relative heading of the AUV to one of two directions. The only differences introduced into the paths follow is to have some of them run on the surface and some at 5m altitude. This was to gain insight into the effect that these two common depth settings have on data rate success. The mission path is shown in from a bird's eye view in Figure 9 and from a 3D depth perspective in Figure 10. Note that the submerged IVER2 does not follow a constant-depth path; this is due to the profiled terrain of the Bedford Basin at the test location.

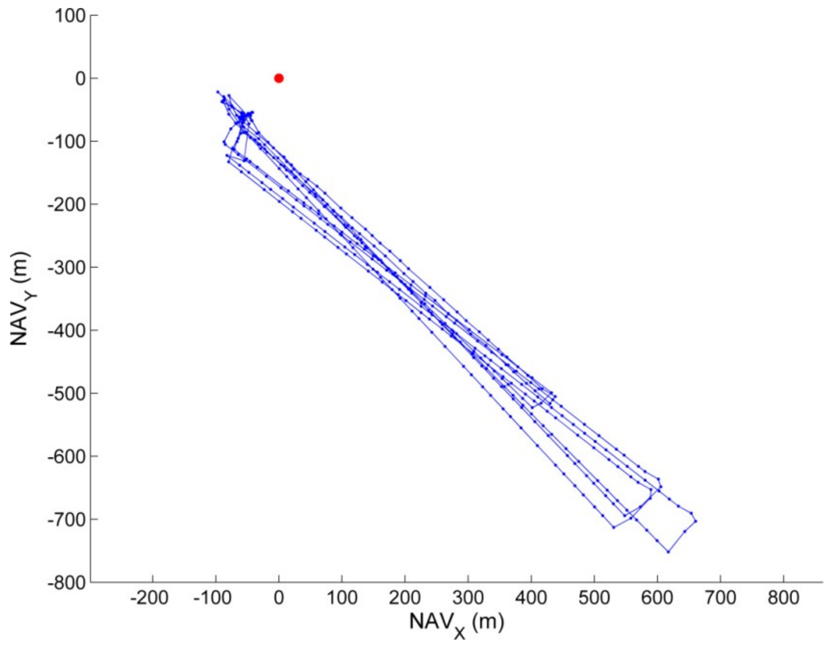


Figure 9: Barge trial mission path top view. Red dot represents the barge location; blue lines the IVER2 AUV mission path; and blue dots locations of acoustic activity.

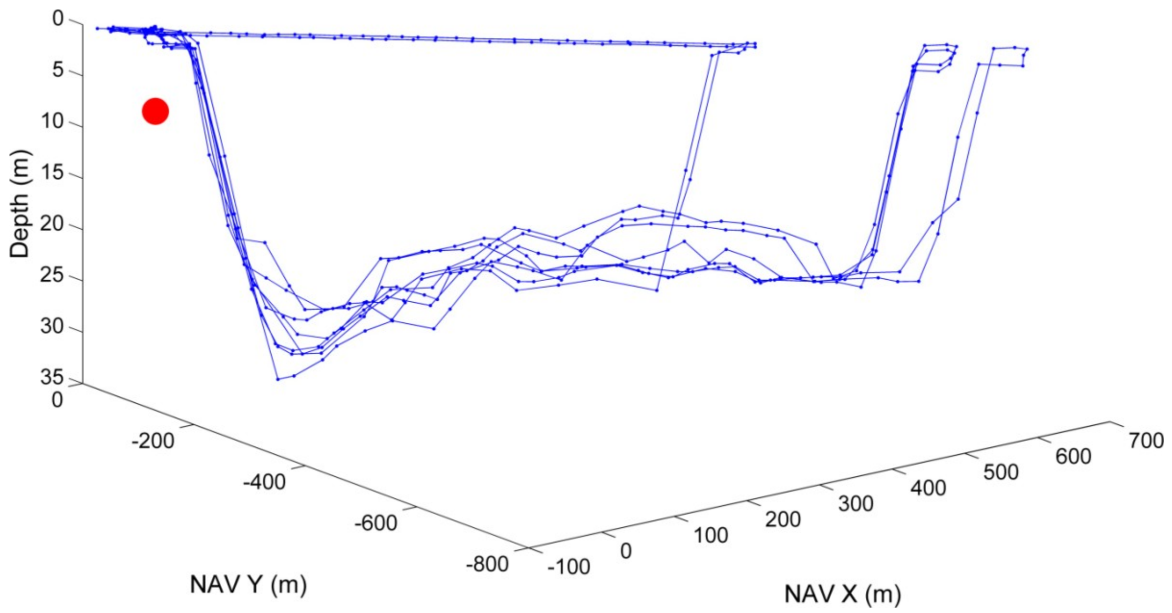


Figure 10: Barge trial mission path isometric view. Red dot represents the location of barge transponder; blue lines the IVER2 AUV mission path; and blue dots locations of acoustic activity.

3.2. Results

The trials were successful and collected 380 paired transmission / reception data points throughout the course of 17th October, 2013. The most basic stage of analysis is to correlate the geographical location of the IVER2 AUV with success or failure of transmissions regardless of WHOI data rate. The following two images display the success or failure of acoustic transmissions according to the position of the IVER2 at the time of transmission. Figure 11 shows a top view and Figure 12 a section view. In these two images, green dots represent AUV position at time of successful transmissions and red dots at time of failed transmissions. The white (Figure 11) and black (Figure 12) rectangles represent the ACB transducer location. Note that in some instances a packet was only partially received: for the purposes of these initial analyses a partial reception is counted as successful. The smallest unit of data input to, and output from, the Micro-Modem is the frame rather than the bit or packet [86]. Therefore, transmissions using the multi-frame PSK data rates 1, 4, and 5 can partially succeed by receiving some but not all frames. The single-frame FSK rate 0 transmissions can only succeed or fail.

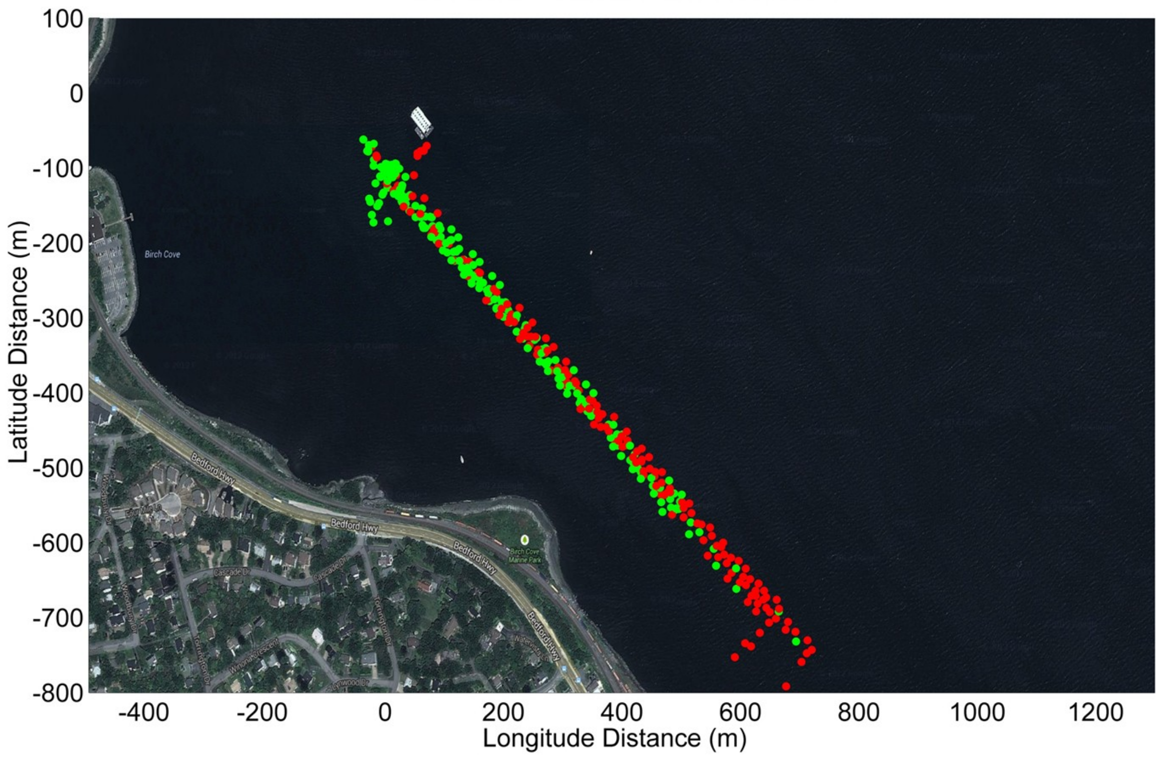


Figure 11: Top view of field trial transmission successes

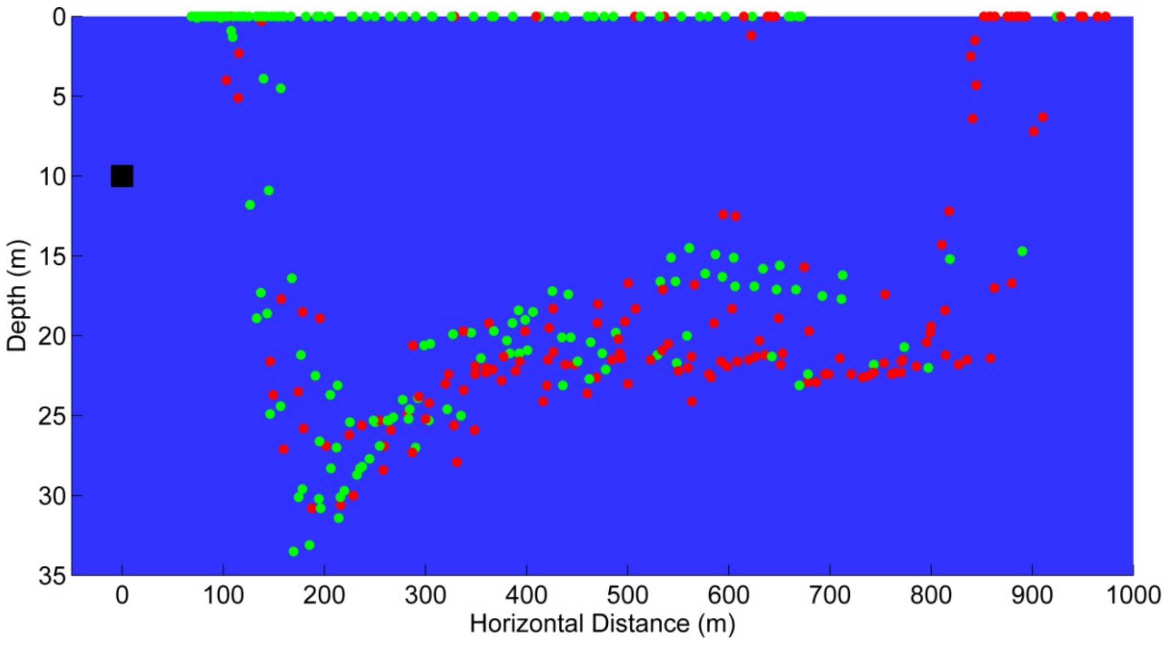


Figure 12: Section view of field trial transmission successes

The next stage of the analysis is to begin extracting the relative performance of each transmission data rate by comparing the likelihood of success achieved while using each rate as a function of channel quality. While not necessarily an accurate measure of channel quality, distance bins were selected as the first-pass measure of quality because distance or range so heavily influences a variety of acoustic loss types such as absorption, reflection, and multi-pathing. Figure 13 details the likelihood of frame success for each of the four aforementioned WHOI data rates as a function of distance between source and receiver. As expected, the lower rates (0 and 1) performed most reliably while the highest rate (5) had at best a 50% chance of success even at short range. It should be noted that the likelihood of success of rate 5 increases to a non-zero value at the 800m distance mark from its zero value at 600m. This increase is prominent because the field trials concentrated at the 0-700m range and only 8 samples fall into the rate 5 800m bin. One long-range rate 5 transmission did succeed, and with few samples in that distance bin it raised the plotted likelihood of success to 12.5% at 800m+ range. The actual likelihood of success is expected to continue to decrease with decreased channel quality.

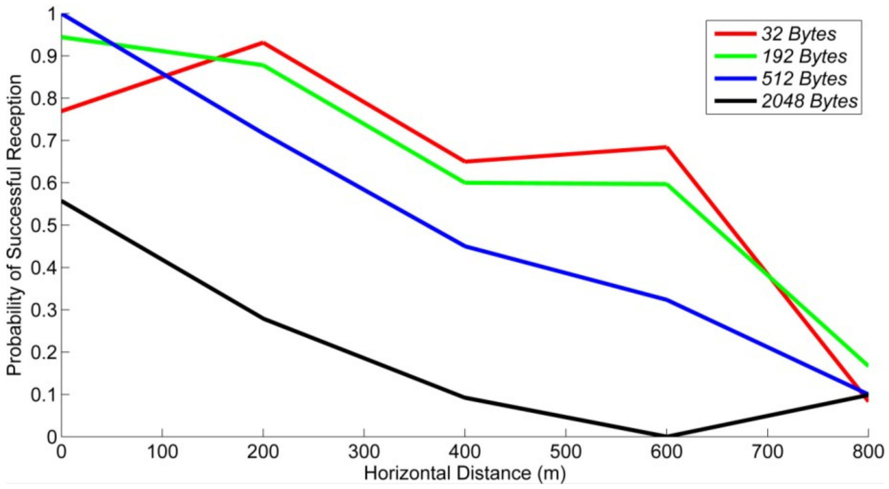


Figure 13: Likelihood of frame success vs transmission distance

The poor likelihood of success for a rate 5 packet is, however, compensated for by its massive data throughput. Data throughput per transmission is the product of packet size and likelihood of success as detailed in Figure 14. Here the rate 5 (black line) is significantly superior to other rates at the 0m and 200m bins, but its performance quickly diminishes at longer ranges due to plummeting likelihoods of success.

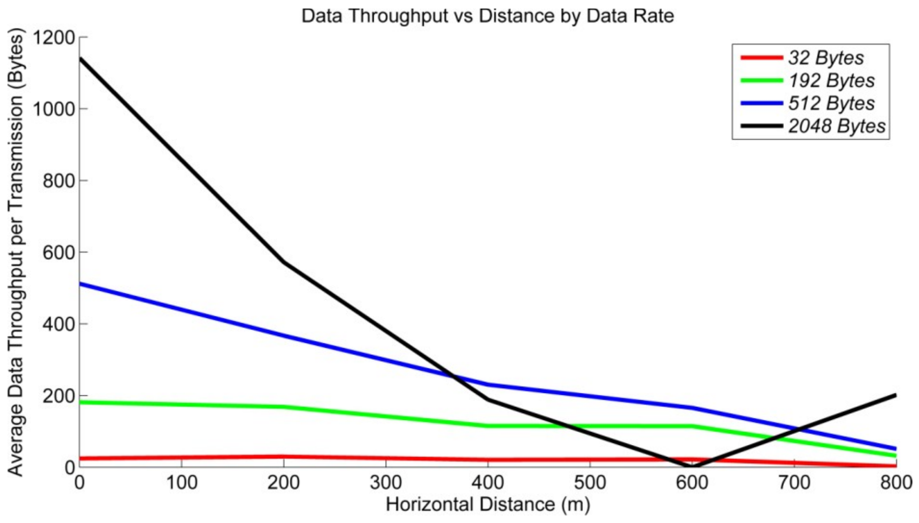


Figure 14: Mean successful data received per transmission

This preliminary analysis uses distance as a rough estimate of channel quality. In actuality, channel quality is dependent on many additional factors and can be better characterized by a more descriptive qualifier such as impulse response, multipath profile, or transmission loss. Each of these take into account a multitude of environmental and acoustic influences. Chapter 4 details the design and validation of a simulation environment that re-creates in simulation the above described field trials while measuring simulated channel quality at each transmission step. This allows for the correlation of in-water transmission performance with channel quality.

CHAPTER 4 MATLAB SIMULATOR DEVELOPMENT

A simulation environment is required to assess the performance of different adaptive communications algorithms while minimizing the need for expensive hardware and field trials. The key requirements are identified as follows:

1. Kinematics and path following for an arbitrary number of AUVs. Mission path planner should be able to parse native IVER2 mission files for full compatibility with field missions.

2. Ability to load representations of AUV test site environments. Particularly necessary is to represent the bathymetric and acoustic properties (fluid and seabed) of the Bedford Basin.

3. Integration with Acoustic Toolbox [89] ray tracing tools to provide acoustic channel quality metrics. Acoustic Toolbox is a package of physics-based acoustic performance calculators as detailed in section 4.2.2.

4. Engine that decides success or failure of WHOI Micro-Modem transmissions. Inputs must be the data rate and acoustic quality channel.

5. Ability to efficiently batch solve mission scenarios. This allows for the testing of a larger number of adaptive algorithm variants.

6. Code structure allowing for rapid development and easy results analysis and plotting.

A variety of underwater vehicle simulators are available, with the most accessible being *uSimMarine* which is integrated into the MOOS-IVP [87] autonomy software already in use on the vehicles. This simulator has vehicle kinematics, path following and acoustics toolbox integration. Loading the Bedford Basin environment and the transmission success decision engine are aspects specific to this project and would have to be added into any selected simulator. The code efficiency and ease of use requirements, however, are poorly met by MOOS-IVP's C++ environment. As a language of choice for scientific analysis and computing, MATLAB was selected for the design of a custom simulator due to its reduced development cycle and native support for high-performance multi-threaded computation. Figure 15 details the simulator layout: all blocks except the gray Acoustics Toolbox were developed by the author. The red modem controller block is the focus of this thesis – the rest of the simulator is created only to support development of the modem controller.

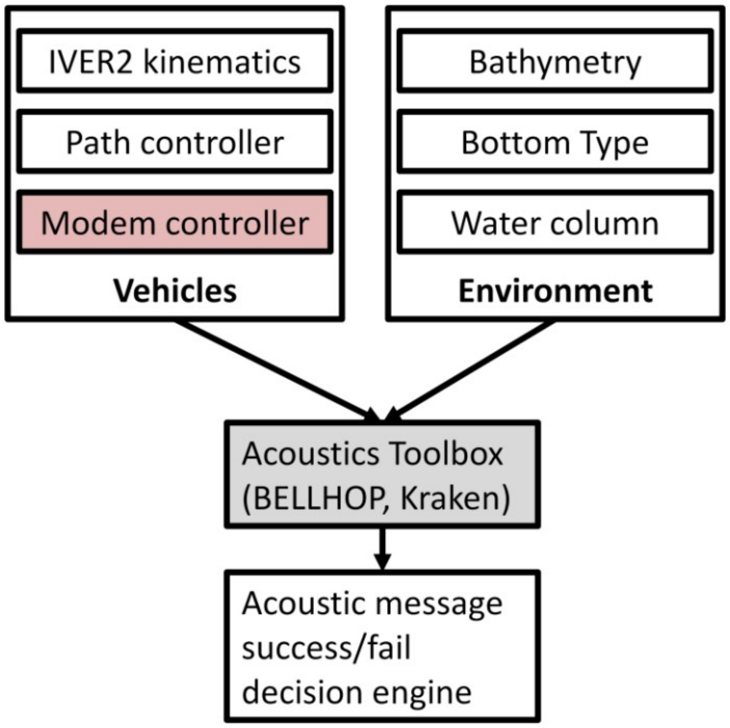


Figure 15: Simulator block diagram

The simulation design is divided into two major work areas. The development of the simulation engine itself is presented in sections 4.1 through 4.3, and the development of representative use cases and performance metrics is presented in sections 4.4 through 4.5.

4.1. VEHICLE MOTION

A vehicle motion engine is required to allow simulation of the acoustic interaction between multiple vehicles. Unlike vehicle attitude controls algorithms that operate on the order of 10-100Hz, the objective of this research is to develop controls algorithms for inter-vehicle communications that operate on the order of 0.1Hz (10s cycle times). Therefore, in comparison to the precise requirements of simulators developed for low-level vehicle control, this vehicle motion simulation needs only to accurately represent vehicle state at frequencies on the order of 0.1Hz. The motion simulation is divided into a kinematics engine and mission path planner.

4.1.1. *KINEMATICS*

The movement engine moves the simulated vehicle with realistic motions toward its next waypoint. As the intent is not to simulate the control surface to environment interaction, full vehicle dynamics are not necessary: a simple kinematics engine is capable. The vehicle's pose $(x_{i,}y_i, d_i)$ is computed using controlled variables pitch (*P*), yaw (*Y*), and speed through water pitch (v_w) via the following relationships:

$$v_v = sin\,(P) \cdot v_w \qquad \textbf{(3)}$$

$$d_i = d_{i-1} + v_v \cdot t \qquad \textbf{(4)}$$

$$v_g = cos\,(P) \cdot v_w \qquad \textbf{(5)}$$

$$x_i = x_{i-1} + sin\,(Y) \cdot v_g \cdot t \qquad \textbf{(6)}$$

$$y_i = y_{i-1} + cos\,(Y) \cdot v_g \cdot t \qquad \textbf{(7)}$$

where

$P = pitch$ (deg)

$Y = yaw$ (deg)

v_w = *speed through water* (m/s)

v_v = *vertical speed* (m/s)

v_g = *speed over ground* (m/s)

d_i = *depth* (m)

x_i = *x position* (m)

y_i = *y position* (m)

t = *kinematic timestep* (s)

The kinematics controller uses three separate PID control loops as shown in Figure 16 to represent the three primary control outputs available to the IVER2 AUV:

1. Yaw: the IVER2 AUV has two vertical control fins that control yaw. The inputs to yaw control are the next target coordinates and the current coordinates and heading.

2. Pitch: the IVER2 AUV has two horizontal control fins that control pitch and, by extension, depth. The inputs to pitch control are the current depth and depth set point, which is determined by combining the current altitude with the mission's altitude set point.

3. Speed: the IVER2 AUV has a variable speed propeller to control its forward speed. The inputs to speed control are simply desired and actual speed.

While the IVER2 AUV is also capable of combining vertical and horizontal fin controls to affect roll, the vehicle attempts to maintain zero roll so for the purposes of this simulation it is assumed that the vehicle maintains zero roll at all times.

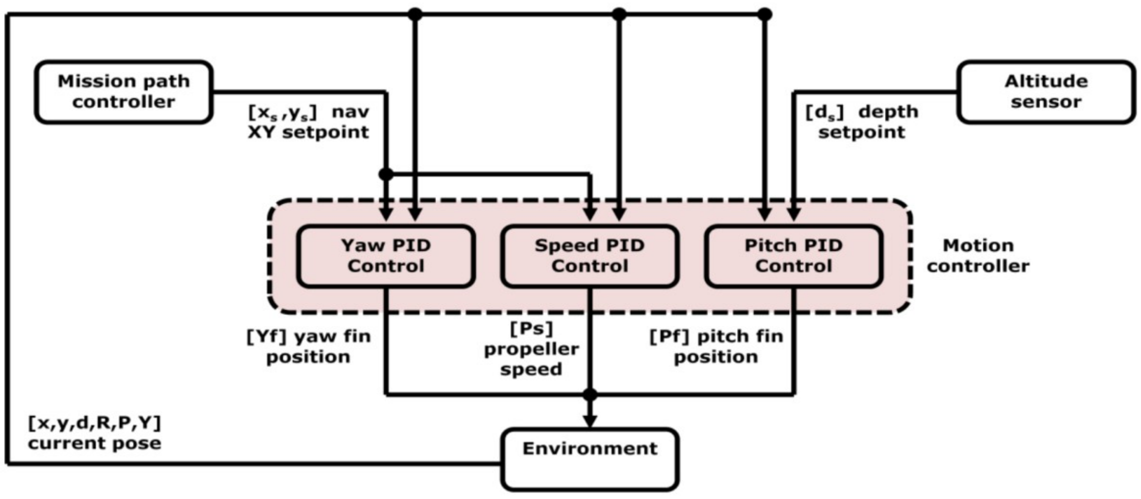


Figure 16: Vehicle motion kinematics controller

The vehicle controller was limited using the kinematics variable constraints shown in Table 3. These constraints were extracted through analysis of the IVER2 motion during the 17th October, 2013 field trial described in section 3.1.

Table 3: Kinematics parameters

Variable	**Limit**	**Notes**
Yaw Speed	$13\, deg/s$	
Yaw Acceleration	$12.9\, deg/s^2$	
Pitch Speed	$10\, deg/s$	
Pitch	$25\, deg$	Set in Vector Map mission file: default value
Forward Acceleration	$0.25\, m/s^2$	
Forward Speed	$1.54\, m/s$	Set in Vector Map mission file: typical value of 3kn

4.1.2. MISSION PATH FOLLOWING

The motion controller described in the previous section causes the simulated vehicle to travel toward the current waypoint. In order to carry out a multi-waypoint mission, a mission path controller is required to act one level above the low-level motion controller to direct the vehicle's choice of next waypoint. To allow compatibility between simulated and in-water missions, the simulated mission controller is designed to parse the *.mis* file format generated by Ocean Server's *Vector Map* mission planning software [90]. During mission execution, the mission controller simply updates the next waypoint once the vehicle reaches a threshold distance from the current waypoint. The IVER2 standard threshold radius of 5m was used.

The vehicle controllers are implemented within the larger context of the environment simulator as detailed in Section 4.2.

4.2. ENVIRONMENT

The mission environment influences both vehicle movement and inter-vehicle acoustic communication. The vehicles follow a TDMA move – transmit/receive – move – transmit/receive cycle. To capture this, the simulation of each mission alternates between two distinct modes: vehicle motion mode and acoustic transmission mode.

4.2.1. VEHICLE MOTION MODE

Each simulated vehicle's motion and mission path controllers allow it to navigate a predetermined mission plan within a simulated environment. The simulator incorporates a bathymetry map that the vehicles can use to make height from bottom measurements for realistic altitude keeping motions toward maintaining an altitude setpoint. As the majority of local DRDC field trials using the IVER2 AUVs are launched from the ACB into the Bedford Basin, the bathymetry of the Bedford Basin was used for all simulated missions [91]. An example two-vehicle mission layout overlaid on the local bathymetry is shown in Figure 17. The green lines represent vehicle 1's mission path with green dots representing waypoints; red represent vehicle 2's mission path. The black square located at the origin represents the ACB. This mission, like all simulated mission scenarios, was run at a 5m altitude keeping variable depth setting.

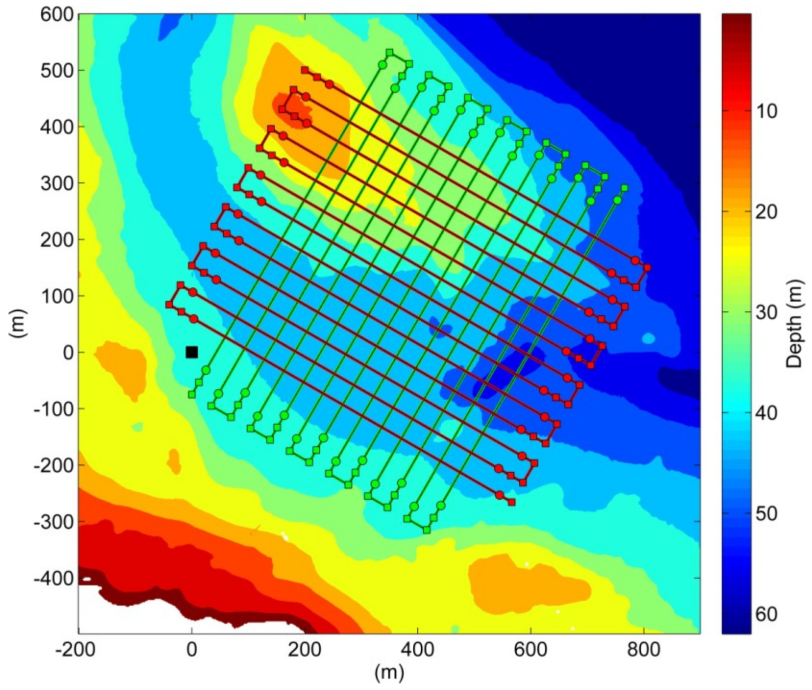


Figure 17: Example collaborative mission plan with bathymetry

4.2.2. Acoustic Transmission Mode

At the time a simulated vehicle transmits a message to its collaborators, the environment simulator switches from vehicle motion to acoustic transmission mode. In the event of a mission with $N > 2$ vehicles, all acoustic transmission mode calculations are repeated $N - 1$ times to capture every source-receiver interaction. There is no meaningful interaction between receivers in a single source $N > 2$ vehicle scenario.

The first stage to the acoustic transmission calculation is slicing the bathymetric environment to provide a 2D bottom profile section between source and receiver vehicles. This section is required for input to the 2D BELLHOP ray trace tool [89]. BELLHOP provides acoustic ray tracing which accurately takes into account a variety of environmental influences such as SSP, bottom profile, bottom composition, surface composition, and transducer beam pattern. As shown in Figure 18, in a shallow water environment the TL at any location is heavily dependent on bottom profile, leading to both multipath and dead zones. The green square on the bottom left represents the source vehicle and the red square on the top right the destination vehicle. Note how not only bottom and top reflections affect beam patterns, but also refraction from the steep sound speed gradient present at 20m depth.

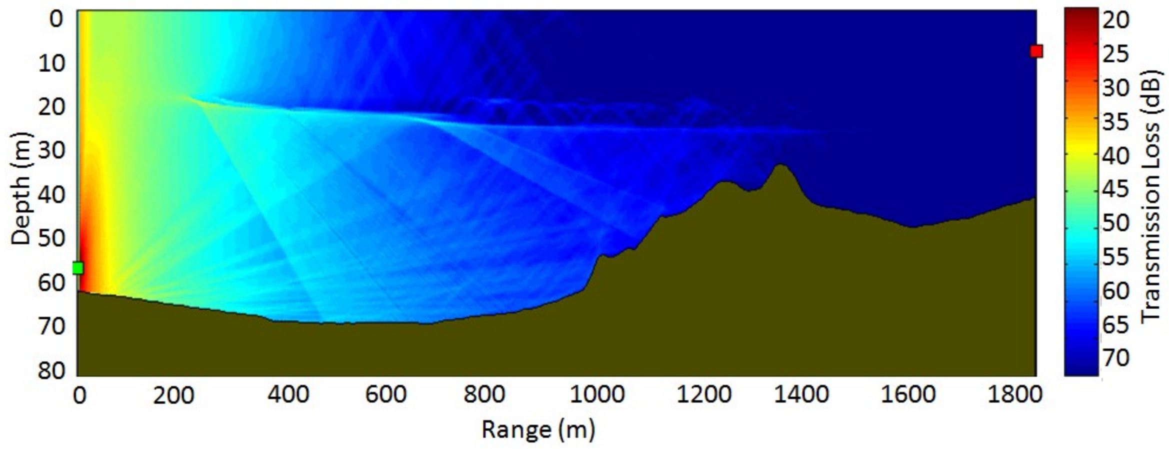


Figure 18: BELLHOP shallow water acoustic transmission loss profile

The following key parameters were used to model the Bedford Basin and WHOI Micro-Modem transducers in BELLHOP. At each transmission mode calculation, the MATLAB simulation environment automatically generates input files for computation by BELLHOP. Refer to Appendix A for an example set of auto-generated input files.

1. Bottom type. The Bedford Basin was modeled using a bottom composition of deep, soft mud as listed in a historical survey [91]. This bottom type was modeled with density $\rho = 1.6\ g/cm^3$, speed of sound $c_p = 1490\ m/s$, and bottom attenuation $\alpha_p = 1.0\ dB/\lambda$ [91].

2. Bottom profile. Generated by creating a 2D slice of the 3D bathymetric environment in the source / receiver plane, the bottom profile significantly effects reflection patterns.

3. SSP. Profiles were used from a sound speed cast taken in the 17th October, 2013 trials detailed in Section 3.1 and an interpolation of typical May profiles [91].

4. Beam type. BELLHOP offers Geometric beams as the default, with option for Gaussian, Cartesian, and Ray-Centered beams. Cartesian and Ray-Centered beams provide increased fidelity when modelling interference patterns [92], but when frequencies exceed 10KHz the interference patterns become dependent upon a level of detail that exceeds the uncertainty in environmental measurements. Gaussian beams were selected because they offer increased smoothing compared to simple geometric beams [93].

5. Calculation type. BELLHOP offers coherent and incoherent TL modes as well as an arrivals mode. Coherent preserves interference patterns while incoherent explicitly discards all interference patterns for a smoothed plot. Incoherent mode was used for TL calculations because the 25KHz frequency employed by the Micro-Modem is high enough that, as stated above, the dependencies of interference patterns cannot accurately be captured in simulation. The arrivals mode generates the amplitude, phase, and time delay of each ray incident on a receiver: this mode was used to generate an impulse response for multipath calculations.

6. Source beam pattern. The BTech model BT-1 transducer is integrated into both the static tow fish and mobile IVER2 modems. In theory this transducer produces a toroidal beam pattern that is omnidirectional in the horizontal plane but shaped in the vertical as seen in Figure 19. At each simulated transmission, the pitch angle of the source AUV is used to generate a modified 2D beam pattern to accurately represent the transducer beam pattern in the plane of calculation.

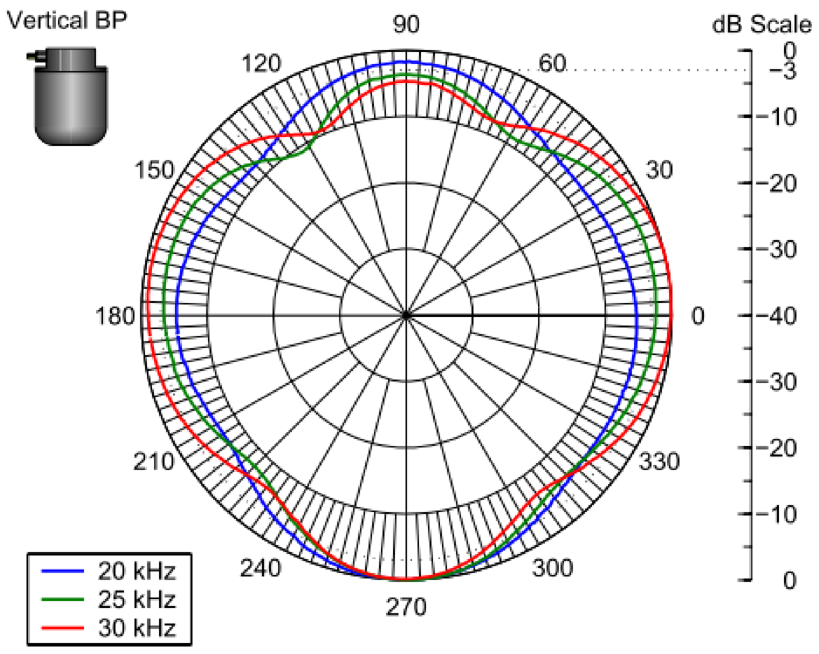


Figure 19: BT-1 transducer beam pattern vertical section [94] The green line represents the 25kHz transducer used.

7. Calculation outputs. For a given input scenario, BELLHOP can output rays, eigenrays or acoustic pressure over the range-defined calculation scope, or impulse arrivals at a particular receiver location [92]. Rays and eigenrays are useful for visualization, but not as a quantitative metric by which the channel quality can be compared. The TL – calculated from acoustic pressure – and the impulse arrivals are quantitative channel quality metrics. Both of these outputs were used in the effort to develop a model of the WHOI Micro-Modem success as a function of data rate and channel quality as described in the following section.

The results of each BELLHOP calculation are used to estimate acoustic channel quality as detailed in Section 4.3.

4.3. WHOI Micro-Modem Model

The Micro-Modem model was developed in two stages. First, the simulation environment described in the above sections was configured to replicate as exactly as possible the conditions of the 17th October, 2013 field trials. Instead of using the vehicle motion simulator, actual vehicle locations at field trial packet transmissions and receptions were directly input to the environment acoustic calculator. The Bedford Basin was modelled as described in the above section, and the SSP as measured on the 17th October, 2013 field trials was input to BELLHOP. For every field trial acoustic transmission / reception pair, the simulation duplicate of the field trial was used to calculate both acoustic pressure (TL) and arrivals (impulse response) at the receiving modem's location. This generated a dataset correlating simulated channel quality to actual modem performance.

This dataset was used to create a data rate dependent modem success model. With both TL and impulse response available, a first-pass analysis was conducted to determine which of these two channel quality metrics is more significantly correlated with modem success.

When set to calculate acoustic pressures, BELLHOP outputs a *.shd file containing acoustic pressure at nodes across the entire depth and range of the slice as seen in Figure 18. The value of acoustic pressure at the node closest to the receiving transducer is used to find the transmission loss via the following equation [89]:

$$\mathrm{TL} = -20\log_{10}|p| \tag{8}$$

where

TL = *transmission loss* (dB)

p = *acoustic pressure* (Pa)

Simulated TL, as expected, is inversely correlated with frame success as seen in Figure 20. Also as expected, the lower bandwidth data rates have a higher likelihood of success at a given TL. The shape of the four curves follow a plateau – linear decrease – plateau paradigm similar to that observed by Pompilli's deepwater WHOI simulations in

[5] as seen in Figure 21. Overall, the strong correlation seen between TL and frame success in Figure 20 conforms to expectations regarding the physical relationship between transmission strength loss and frame success. This gives confidence that the simulated TL values correctly reflect actual in-water conditions. In comparing the success vs TL results (Figure 20) to the initially derived success vs distance results (Figure 13), it can be seen that the TL produces a more defined and continually decreasing correlation. This confirms that TL is a more well-rounded indicator of channel quality than distance alone.

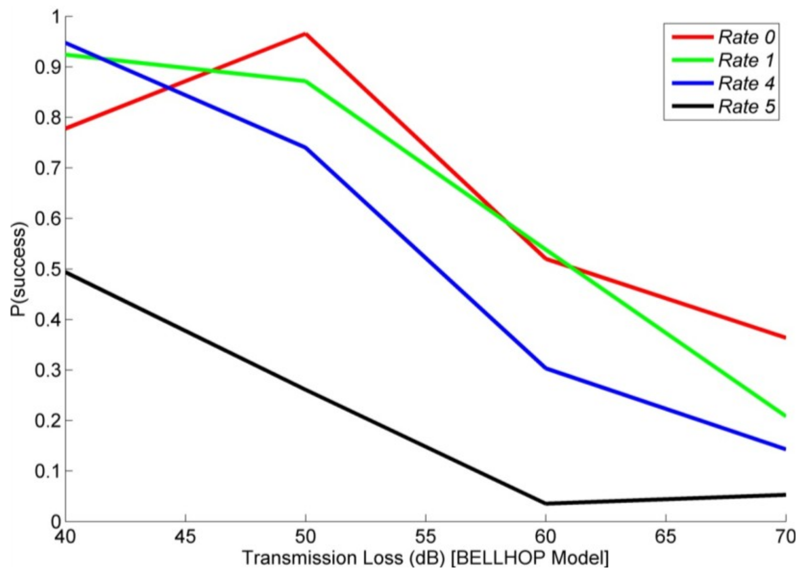


Figure 20: Frame success vs transmission loss

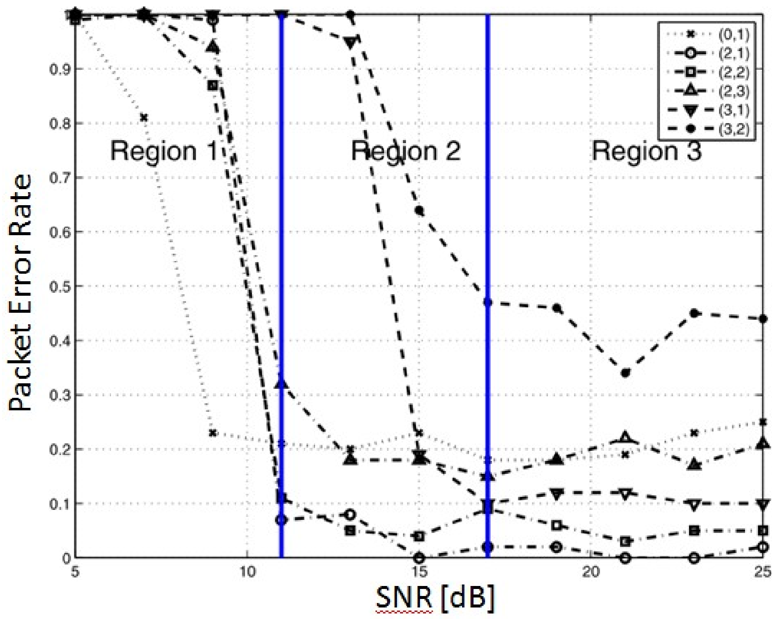


Figure 21: Reference packet error rate vs SNR [5]. Note that the graph is divided into regions for an analysis not relevant to this work.

When set to calculate impulse response, BELLHOP outputs a *.arr file detailing every ray arrival at the receiver location. Ray parameters provided include amplitude (dB), time delay (s), and number of reflections from both the top and bottom waveguides. An example impulse response plot is shown in Figure 22.

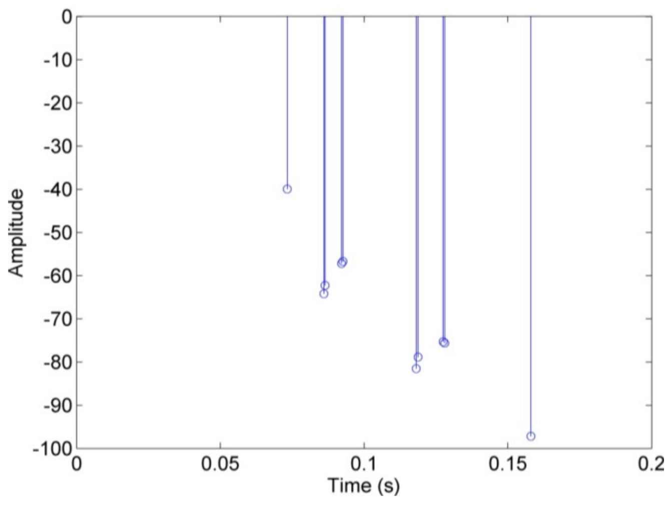


Figure 22: Typical acoustic impulse response

The impulse response multipath may be expressed using a variety of parameters, with the simplest being delay spread [95]. Two of the most common are mean delay spread and RMS delay spread. Delay spread is a measure of the time spreading of multipath responses; excess delay spread is the time difference between a given arrival and the earliest arrival. Mean delay spread is the mean of all excess delay spreads of a particular channel; RMS delay spread is the root mean square (RMS) of all channel excess delay spreads.

The WHOI Micro-Modem frame success is compared to the multipath delay spread mean and RMS in Figure 23 and Figure 24. Unexpectedly, there is a positive correlation between frame success and both delay spread mean and RMS. This does not correspond to accepted theory: increased multipath spreads should mean an increased likelihood of ISI and lower frame success. To investigate, a comparison is made between the impulse response of a 109m range (high likelihood of success) transmission and a 889m range (low likelihood of success) transmission as shown in Figure 25. In the short-range transmission several prominent rays arrive in distinct multipaths, leading to an extended delay spread. In comparison, the long range transmission only has one clumping of rays which leads to a relatively brief delay spread.

To explore this unexpected relationship further, the eigenrays were plotted for these two transmissions and are shown in Figure 26. These equal axis figures demonstrate that at near transmissions, rays impinge on the receiver from a wide range of angles. However, at distant transmissions, the shallow water environment acts as a waveguide and only allows rays with near-horizontal angles to reach the receiver. This reduces the delay spread for the long range transmissions because all rays follow similarly-angled paths, leading to similar-length paths. Short range transmissions, however, have successful arrivals from rays travelling at a large variety of angles which leads to different length paths.

While this explains the reason for the unexpected delay spread vs frame success relationship, it demonstrates that the simulated delay spread is inversely correlated with distance. These results do not give confidence that the delay spread is a quality predictor of frame success.

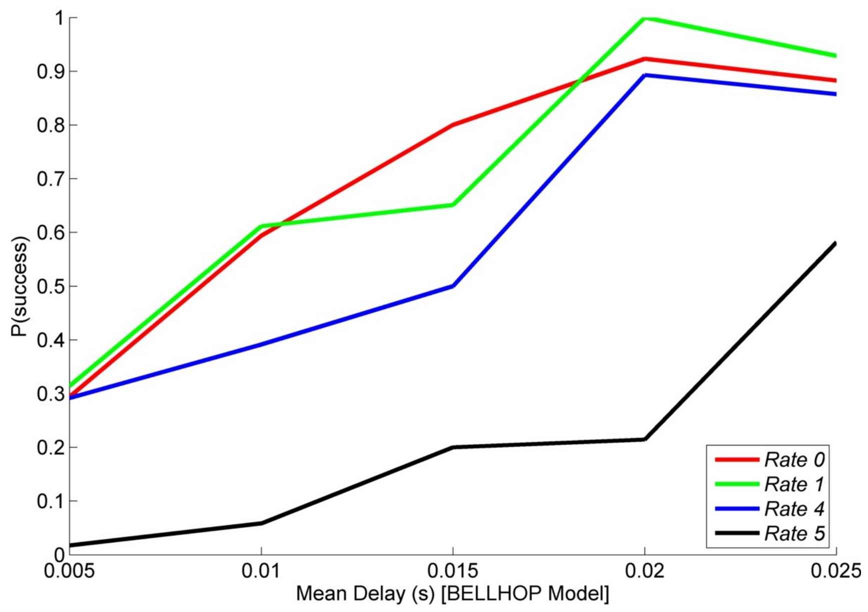


Figure 23: Frame success vs multipath delay spread mean

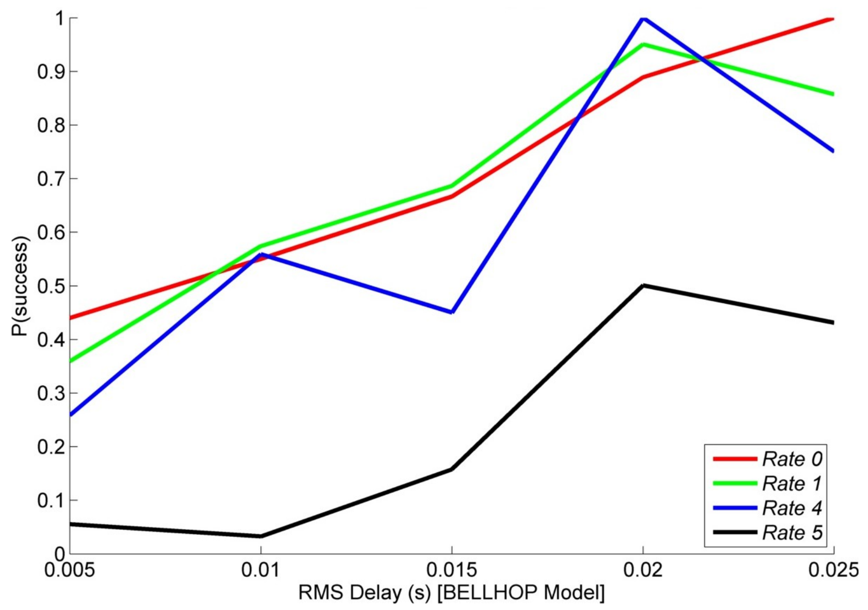


Figure 24: Frame success vs multipath delay spread RMS

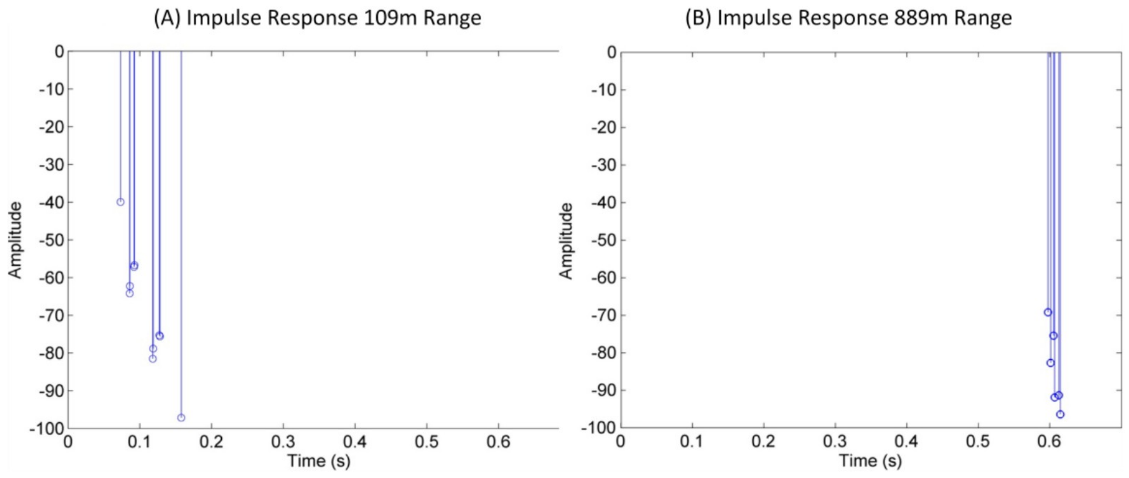
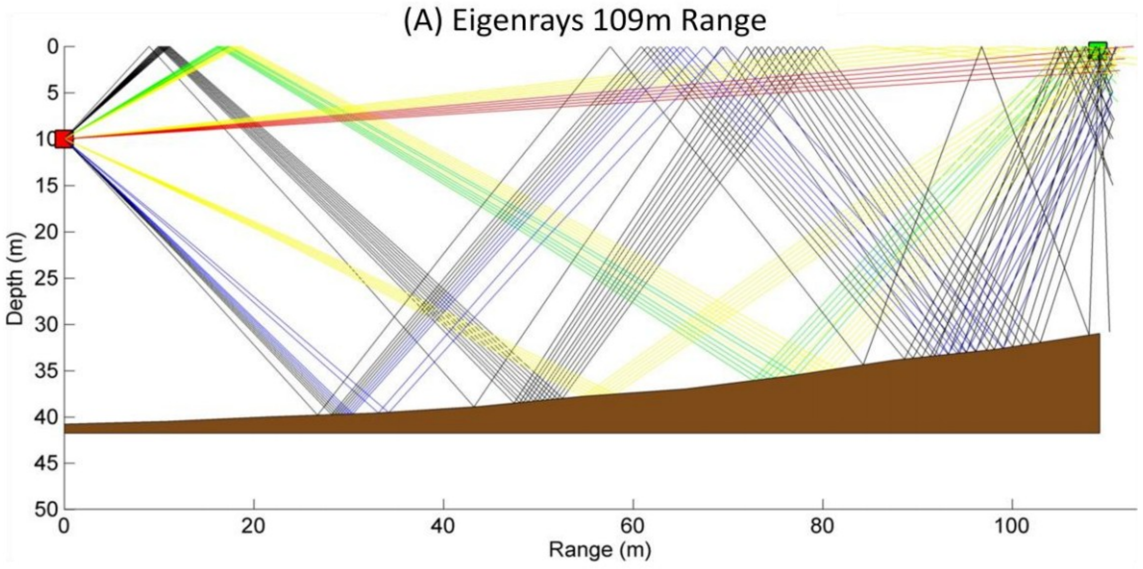


Figure 25: Comparison of short range (A) and long range (B) impulse responses



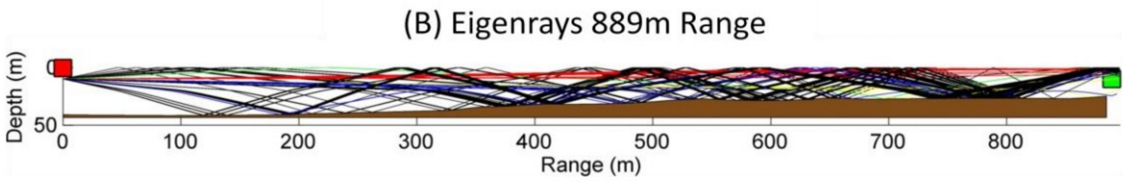


Figure 26: Comparison of short range (A) and long range (B) Eigenray plots. Plots are produced with equal scale axes to show differences in reflection angles. Red squares are source and green receivers.

One further measure of channel multipath was correlated with frame success as seen in Figure 27: the number of transmission paths. As expected, frame success does increase as the number of transmission paths decreases: fewer multipaths mean a more direct acoustic flow and less chance of ISI. However, the degree of correlation is only moderate compared to the correlation with TL. Since TL provides the best correlated and most credible predictor of frame success, it was chosen as the basis for the Micro-Modem model success predictor.

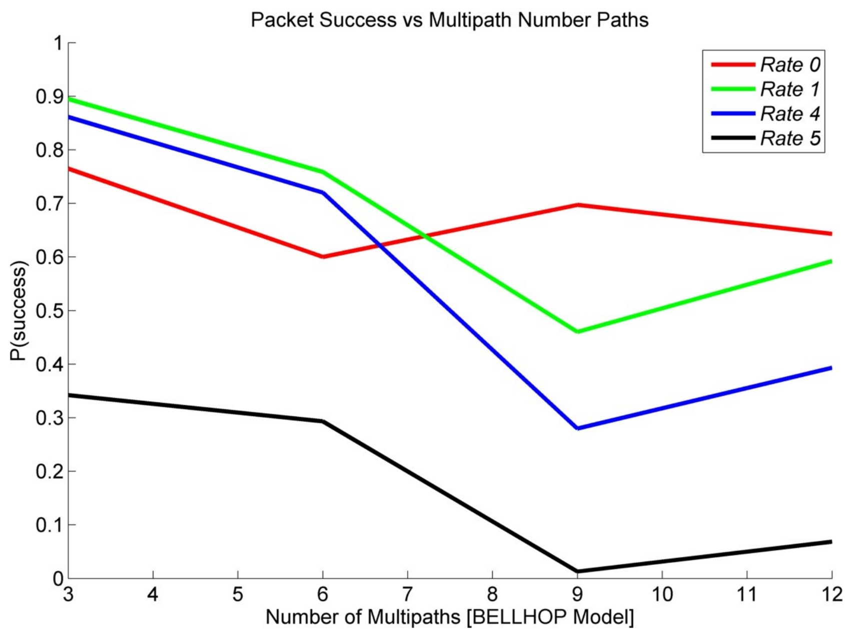


Figure 27: Frame success vs multipath path number

Based on observation of charted success vs TL curves (Figure 20) and Pompilli's related curves (Figure 21) [5], the regression was subdivided into three TL ranges. As seen in Figure 28, these are a constant best-case success rate for low TL (region *A*), a zero success rate for high TL (region *C*), and a linearly declining success rate for intermediate TL values (region *B*). The regression results as a function of data rate are displayed in Table 4 and Figure 29.

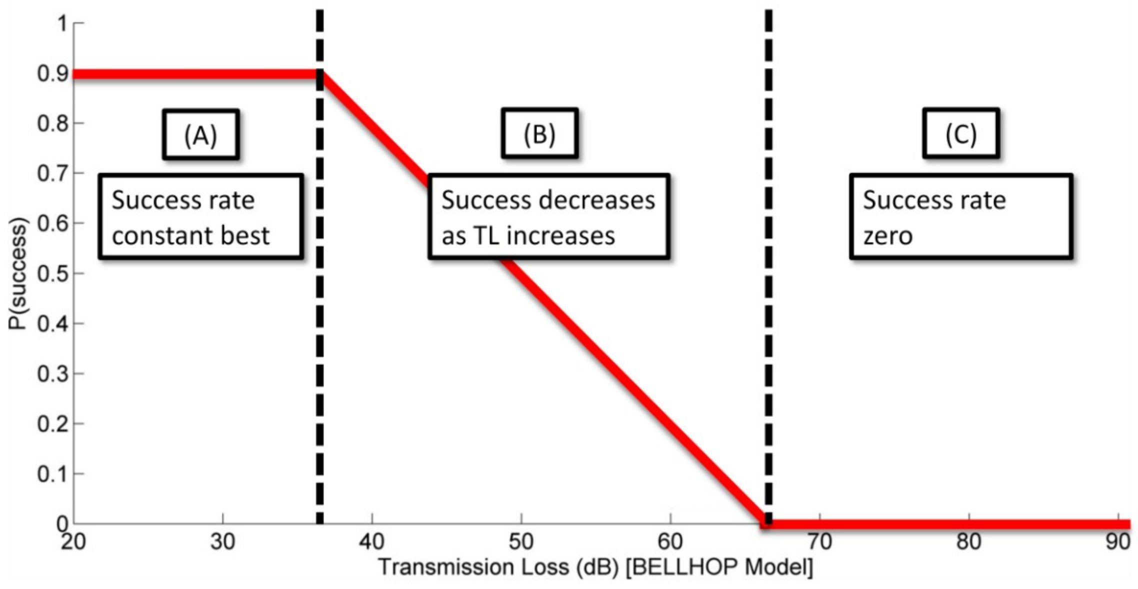


Figure 28: Three-way division of TL range for regression.

Table 4: Regression of frame success vs TL

WHOI Micro-Modem Data Rate	**Likelihood of success Region (A)**	**TL threshold Region (A) to Region (B)**	**TL threshold Region (B) to Region (C)**
0	0.80	50 dB	80 dB
1	0.90	50 dB	80 dB
4	0.80	40 dB	80 dB
5	0.45	40 dB	60 dB

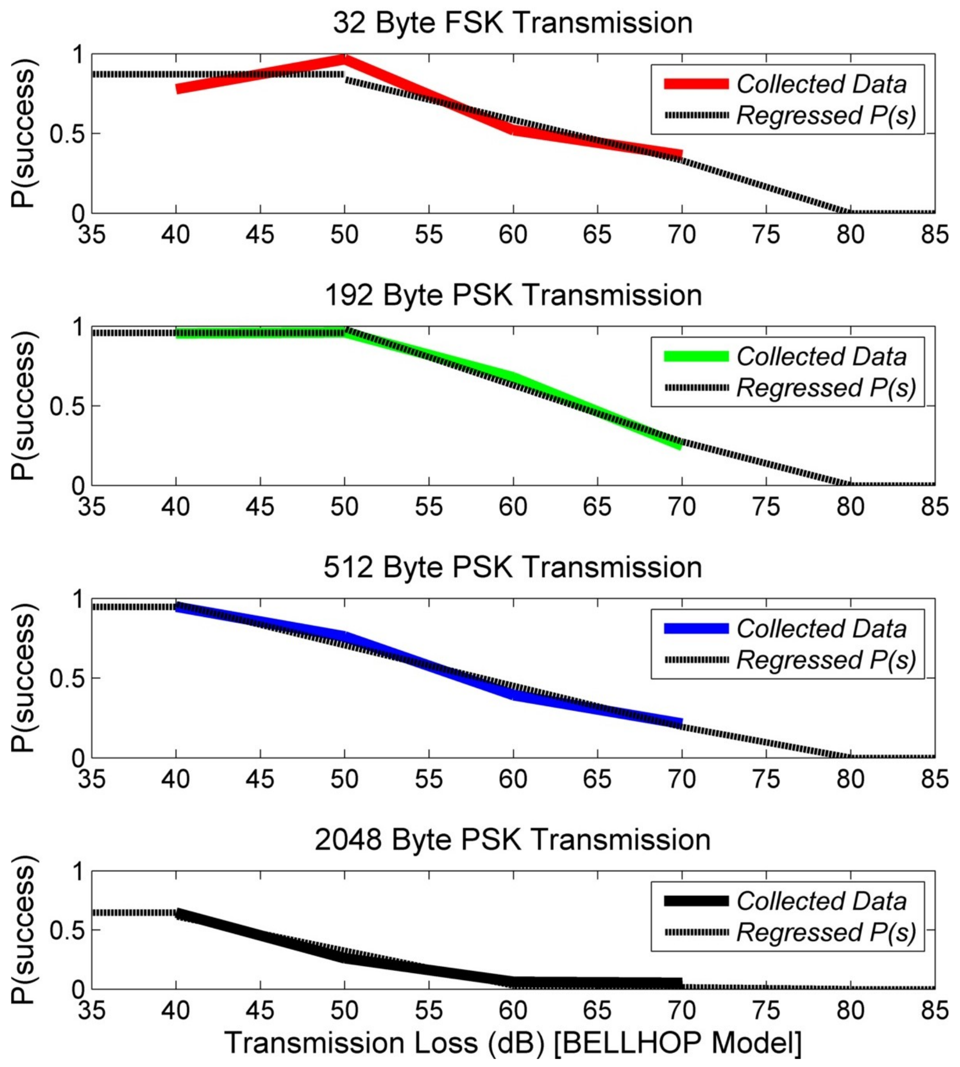


Figure 29: Comparison of WHOI Micro-Modem probability of frame success versus TL as a function of data rate. Black overlaid lines are the regressed values used in the simulator's frame success decision block.

The regression equations described in Figure 29 are used to create the frame success decision engine for the simulator. This engine probabilistically selects frame success or failure, guided by the likelihood in each case given the TL. A complete simulated mission can be run using the vehicle motion controllers to move within the physical environment until a transmission is made, at which point BELLHOP is used to calculate received TL and the frame success engine. A large number of environmental and vehicle factors influence the quality of the calculated acoustic channel: Section 4.4 investigates the relative importance of the most prominent.

4.4. Influence of Mission Factors

Each adaptive acoustic control algorithm must be tested against an identical set of simulated scenarios for performance comparison. A shallow water NMCM mission has a large number of factors that may influence acoustic performance. In order to reduce scope, the influence of four of the most significant was identified and quantified to determine which had the greatest impact on acoustic performance. The influence of each factor was tested in a simple two vehicle mission by calculating the impact of a dichotomy of conditions within each factor; *e.g.* acoustic performance with and without added ambient noise. The impact on acoustic performance was assessed in terms of bandwidth and frequency of successful communications.

4.4.1. Factors

A typical small lawnmower mission in the Bedford Basin is 500m x 500m in size, and the largest possible is approximately 2,000m x 2,000m. Figure 30 details these two missions.

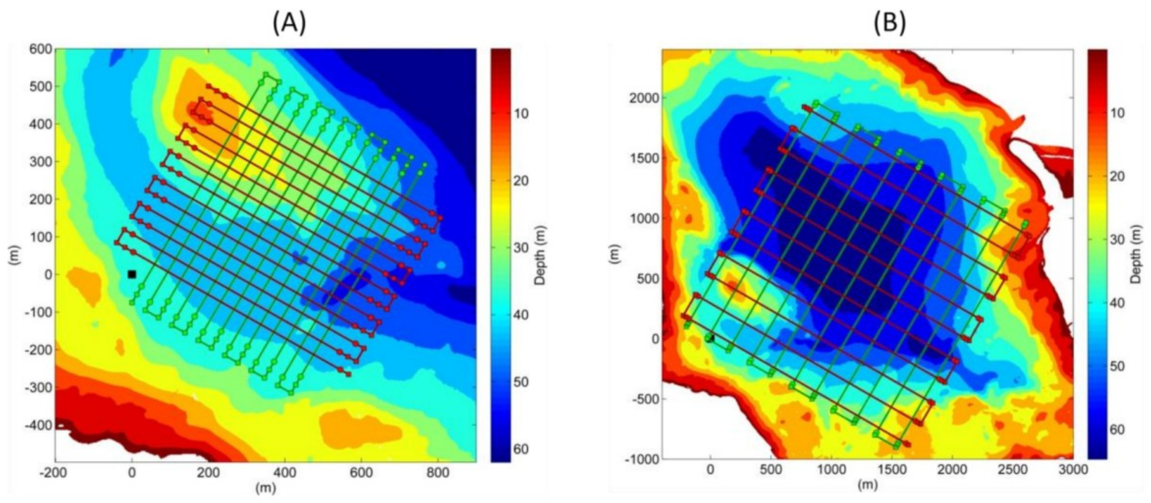


Figure 30: Mission paths used for simulation of typical small (A) and large (B) shallow-water missions within the Bedford Basin, Halifax, NS, Canada.

Ambient noise at the 17th October, 2013 sea trial is taken into account in the simulator. However, additional noise may be generated through environmental (rain, aquatic life, waves) or anthropogenic (ships, sonar) sources. Ambient noise levels of $0dB$ and $6dB$ were compared; $6dB$ is an estimate for high typical noises in populated shallow water port.

The SSP varies throughout the year and also after transient weather events such as precipitation. October (summer profile) and May (winter profile) SSPs were compared and are shown in Figure 31.

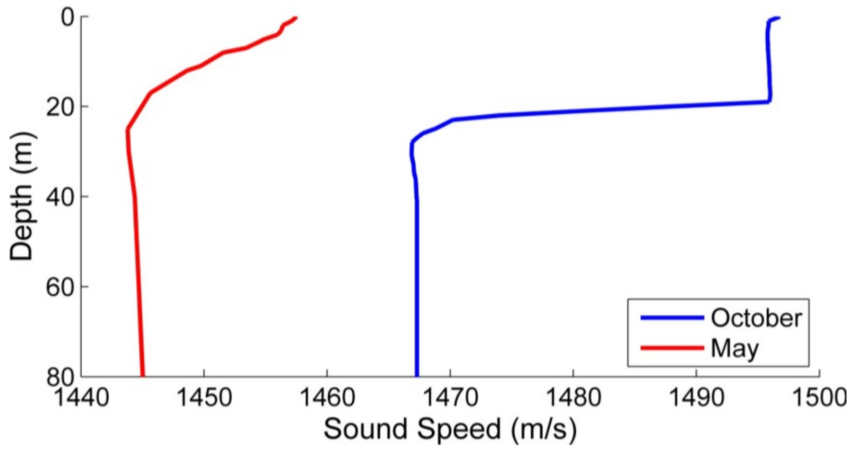


Figure 31: Seasonal SSP variability in the Bedford Basin, Halifax, NS, Canada.

During the 17th October, 2013 trial, both the IVER2 and deck modem had identical WHOI Micro-Modems and transducers. However, the transducers were mounted to structures of different geometry and each system had distinct local noise sources. The major performance difference was a tendency for the deck modem to more frequently drop part of a multi-frame packet than the IVER2. As seen in Table 5, this was seen at PSK rates 1 and 4 but not rate 5. This is suspected to be due to intermittent mechanical noise on the ACB itself such as a compressor or other impulse noise source.

Table 5: Vehicle specific partial packet loss

WHOI Data Rate	% Received Packets Partially Dropped by Barge	% Received Packets Partially Dropped by IVER2
1	38%	8%
4	32%	4%
5	38%	33%

While these particular differences in reception were specific to the hardware used on the 17th October, 2013 trial, it is reasonable to expect any exercise using heterogeneous hardware to show differences in reception patterns. This makes it useful to understand the influence of such differences on performance as any intelligent behaviour must be able to adapt to the individual hardware on which it is installed. The influence on overall acoustic performance of these two partial packet reception patterns was compared.

4.4.2. *MISSION FACTOR INFLUENCE RESULTS*

Eight simulation environments were run: each held three of the four influences constant while changing the fourth. Each environment was run with every WHOI Micro-Modem data rate. The results of the influence comparison are summarized in Figure 32. For this comparison, influence is calculated as the difference in performance between the two conditions divided by the performance of the lesser. Path Size factor represents performance difference between a 500m square side and 2,000m square side lawnmower mission path. Noise factor represents performance difference between 0 and 6 dB ambient noise levels. SSP represents performance difference between October and May SSPs in Bedford, NS, Canada. Vehicle represents difference between the WHOI WH-BT-1 transducer mounted on the IVER2 AUV and mounted on the WHOI towfish statically deployed from a barge.

The influence of mission path size and ambient noise are significantly more pronounced than those of the SSP and vehicle type. The SSP had the least influence: it dramatically influences the path of each acoustic transmission, but its influence is not consistently positive or consistently negative which reduces its long-term impact. The vehicle type also had low impact: the impact of increased partially received frames was not significant compared to the impact of larger missions or higher ambient noise.

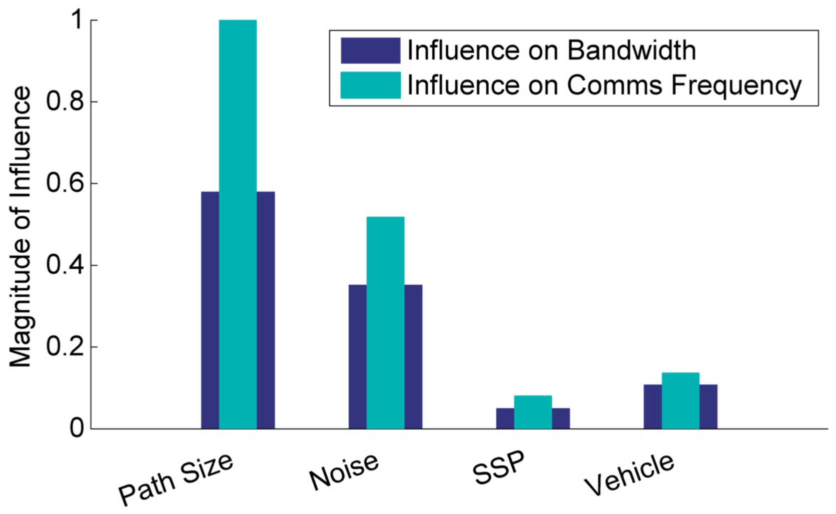


Figure 32: Influence of four mission factors on acoustic bandwidth and communication frequency.

4.4.3. Simulation Scenarios

To reduce computation time and allow for increase testing of adaptive algorithms, the number of simulation scenarios is limited to those differing by only the most influential factors. The simulation scenarios do not account for differences in SSP or vehicle type due to their low relative influence. This leaves mission size and ambient noise as differing influences on the four scenarios used for algorithm comparison as seen in Table 6.

Table 6: Simulation scenarios

Simulation Scenario	Mission Path	Ambient Noise Level	Vehicle Type	SSP
1	500m x 500m square lawnmower	0dB	IVER2	October
2	500m x 500m square lawnmower	6dB	IVER2	October
3	2,000m x 2,000m square lawnmower	0dB	IVER2	October
4	2,000m x 2,000m square lawnmower	6dB	IVER2	October

4.4.4. BATCH RUN PARAMETERS

The simulation represents the uncertainty found in field trials by using a probabilistic decision engine to determine frame success. This means that two simulated trials that follow identical mission paths under identical conditions could experience different communications performance, just as is the case with field trials. The benefit of a simulation environment is that trials can be repeated more easily than can be done with field trials. The simulator was designed to allow quick repetition of trials to allow increased precision of results.

The major computational expense in the simulator is the BELLHOP ray traces: the smaller mission path has 6,694 transmissions each requiring a full 2D raytrace, while the larger mission path has 18,106 transmissions. The longer of these requires 10+ hours to compute using a quad-core 3.5 GHz Intel i7 with 32GB RAM. However, the mission paths are governed by deterministic kinematics and ray trace calculations and will run exactly the same on each pass: the probabilistic aspect is only found in the transmission success decision engine. The simulator was designed to run each scenario once in a preparation mode and save all BELLHOP channel outputs. After this is complete, future simulation passes simply use the results of each pre-computed acoustic channel estimation instead of re-running it. This architecture allows a set of 4 simulation scenarios to be run in 30-60s; multi-thread processing is used to allow one scenario to be handled by each core simultaneously.

A comparison was run to quantify the predictability of the probabilistic results as a function of repetitions. The experiment set one simulation scenario to repeat a basic (fixed-rate) acoustic control behaviour mission for differing numbers of repetitions. Each repetition recorded the mean and standard deviation of the mission's transmission success. Once all repetitions are complete the standard deviation of the mission success parameters was computed; this process was performed for 10, 20, 40, 80, 160, and 320 repetitions as shown in Figure 33. The results show a strong reduction in deviation as the number of repetitions is increased, meaning repeating runs of each scenario definitely increases confidence in the results. This benefit must be weighed against the increased computation time requirements of repeated calculations. Forty repetitions was selected as a compromise between confidence and calculation time.

Any communication algorithm test described hereafter has been run on each of the four scenarios listed in Table 6 with each scenario individually repeated 40 times. In order for algorithm performance to be quantitatively compared, a set of metrics are established as detailed in Section 4.5.

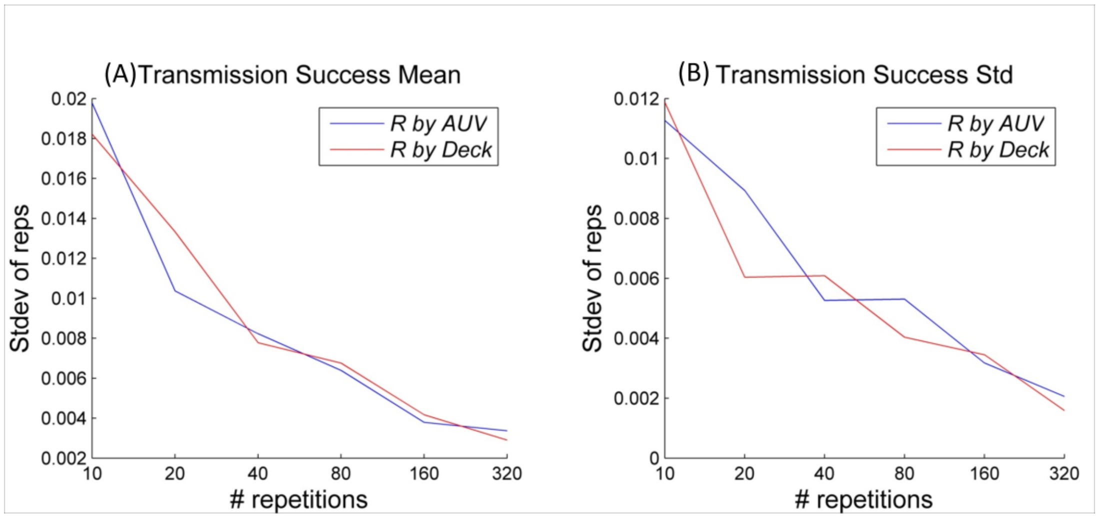


Figure 33: Standard deviation of transmission success mean vs repetitions.

4.5. Communications Performance Metrics

Section 4.4 described the standardized simulation used to test and tune adaptive algorithms. To quantify relative performance, performance criteria were established. It is important to note that performance criteria are subjective. While the performance of a communications behaviour can be quantitatively analysed in light of the performance criteria, the selection of the criteria themselves is a matter of judgement for the vehicle operator.

In this thesis, bandwidth, frequency of success, and duty cycle were selected as three criteria demonstrating the most value for high data rate NMCM missions. Bandwidth is beneficial to allow transfer of information beyond simple status updates such as SLAM mapping data or sonar images. Frequency of success is important for timely status updates to allow human operators or autonomous collaborators. Finally, low duty cycles are beneficial to enable longer missions by reducing on-board power usage. Each algorithm's performance against these criterion was normalized to allow summation for a total score as shown in Table 7. These metrics were applied to all tests throughout the adaptive behaviour design described in Section 5.

Table 7: Algorithm performance criteria

Criterion	Normalization Scoring	Value (%)
Bandwidth (B/s)	Ratio of bandwidth achieved in a scenario versus ceiling. Ceiling is defined as rate 1 transmissions sent every 10 seconds with 100% success (19.2 B/s)	50%
% Duty Cycle	One minus the ratio of transmission opportunities taken to those available.	30%
Frequency of Success	Score of one for any period between updates less than threshold, and half score for each doubling of period past threshold. Threshold determined as 60s for missions where inter-vehicle distance averages less than 500m. For greater inter-vehicle distances, threshold increases by 60s for each additional 500m.	20%

CHAPTER 5 ADAPTIVE ALGORITHM DESIGN

Adaptive AUV communication behaviors are an area of active research. Schneider integrates BELLHOP with MOOS-IVP to perform on-line ray tracing and plan the best path for acoustic connectivity with a static source [4]. Chen and Pompili optimize multi-hop packet path planning [5]. However, current work does not translate well to NMCM missions due to the multiple sensory and path planning constraints inherent in NMCM. Rapid deployment with or without prior environmental information may be required: in this case adaptive algorithms cannot depend on *a priori* knowledge of the acoustic environment. The AUV path is completely constrained by the survey requirements of the mission: the path may not be adapted for acoustic optimization. NMCM AUV sensor payloads are focused on sonar and may not incorporate salinity and temperature sensors: acoustic adaptivity cannot depend on the availability of real-time SSPs. Finally, vehicle paths may be fixed or dynamically updated: acoustic optimization may be able to use pre-determined plans but also must be able to estimate the position of collaborating vehicles dynamically if necessary.

These constraints limit the adaptive communications algorithm to controlling only the modem, not the vehicle path. Additionally, not enough sensor information is assured to attempt to directly characterize the acoustics of each transmission. Rather, it was decided to perform a 'forgetful' characterization of the overall performance of the environment as a function of Micro-Modem data rate and range. The adaptivity algorithm is broken into three modules, with the first two feeding information to the third which performs the decision making:

1. Position Estimator: estimates collaborating vehicles' position with status updates and, if available, mission plans.
2. Data Transmission Performance Tracker: provides more reliable success tracking than the modem's built in tracking feature along with significant bandwidth gains.

3. Rate Selector: decides whether to attempt a transmit and at which WHOI data rate.

5.1. Position Estimator

The position estimator is fed the self-reported position and next waypoint of collaborating vehicles each time a status update is successfully received. Between status updates, the estimator uses kinematics to predict the collaborator trajectory between current and next waypoint.

5.1.1. Estimator Design

The most basic method for estimation of collaborator positions is simply to assume that the position given in the most recent status update is current until it is replaced by the next update. While easy to implement, this does not adequately capture the location of the other vehicle for real-time channel estimation calculations.

Position estimation was significantly improved by running online a kinematics-based waypoint shadow simulation. This shadow simulation is simply the kinematics and path following simulator functions integrated into each AUV's intelligence for use online throughout missions. Each AUV runs a simulation of the position of its collaborator. The positions are updated when new status updates are received and realistically travel toward the next waypoint in between updates.

A key input to the estimator is the mission type: pre-planned or dynamic. If a vehicle has a priori knowledge of a collaborator's pre-planned mission, then it is able to track it even through long communication droughts by estimating waypoint arrival and movement through successive points. In this case, "next waypoint" transmissions need only be a one-byte numerical waypoint identifier corresponding to the coordinates of a pre-planned mission point.

With a dynamic mission type, performance is degraded but still viable. "Next waypoint" transmissions are full latitude and longitude coordinates which cost 7 bytes overhead. This performance is on par with pre-planned missions if constant contact is maintained, but if information regarding a waypoint is lost the estimator assumes the vehicle stops. This leads to error but is superior to the straight-line assumption which leads to runaway estimations.

The position estimator is only tested in simulation using the two-vehicle scenarios described in Section 4.4.3. However, it scales to any number of collaborating vehicles with no modifications save that each vehicle runs a separate simulated state estimator for all collaborators.

5.1.2. PERFORMANCE ANALYSIS

The performance of the position estimator is dependent both on the frequency of successful status updates received from collaborators and also on the presence or lack of mission preknowledge. Performance is compared to a control case where the most recent status updates are used with no live position estimation. Figure 34 details the number of position estimates versus their corresponding error for an entire mission. Error is measured through comparison of the simulator-provided ground truth and the AUV's estimated position. Acoustic communications were set to WHOI rate 1. The top three plots (1) represent scenario 1 while the lower three plots (2) represent scenario 3. The leftmost plots (A) represent best-case where AUV1 has preknowledge of AUV2 path. The center plots (B) represent worst-case where no preknowledge is available. The rightmost plots (C) represent the control where aging statuses are used instead of live position estimation.

Performance with preknowledge is excellent for both scenarios, with all errors < 25m and the vast majority < 5m as shown in. When preknowledge is lost, performance begins to be degraded with errors up to 120m in the smaller scenario 1 and 250m in the larger scenario 3. However, the large majority of errors are still < 10m and < 50m respectively. These errors are larger than with the mission preknowledge, but still significantly improved over the control results where errors grow to over 200m and 800m respectively.

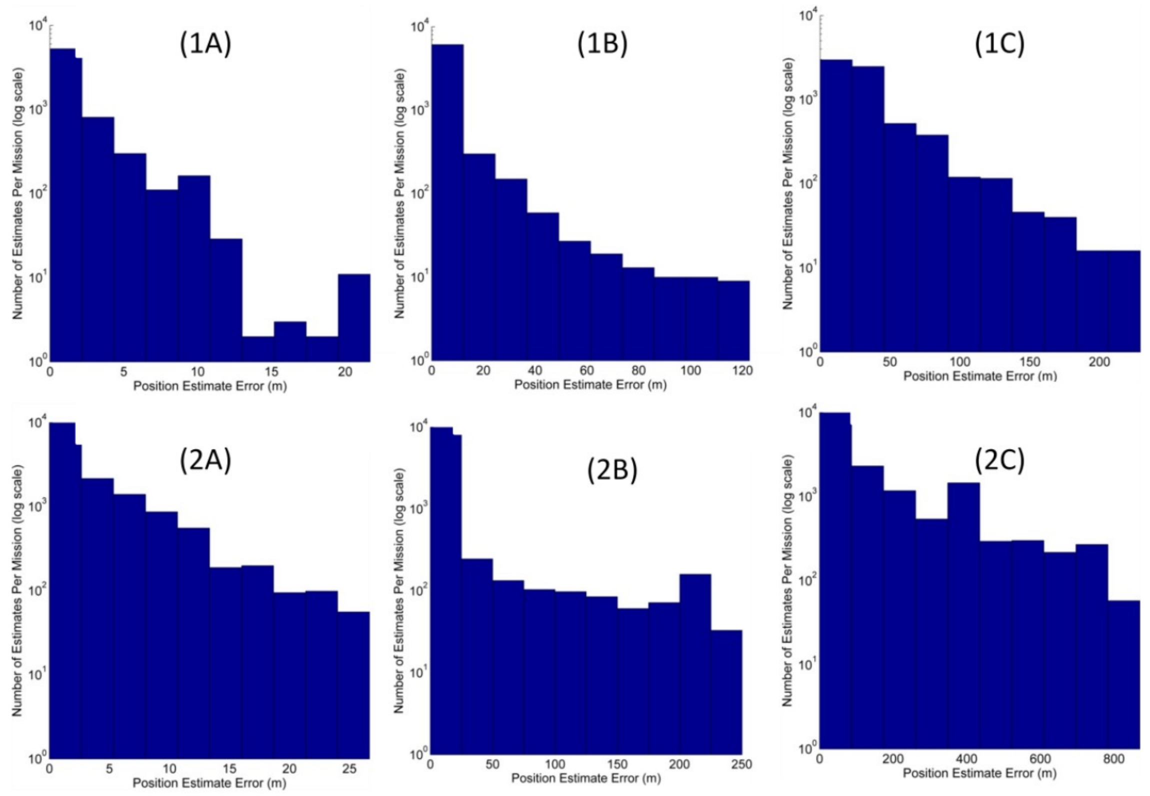


Figure 34: Position estimator error

5.2. Data Success Tracker

Transmission data success tracking is required in any collaborative AUV application where future actions depend on the success or failure of transmitted messages. Such tracking is not required in basic AUV mission scenarios: for example, a mission where AUVs simply send periodic status updates to each other [96]. The future actions of sending additional status updates are unaffected by whether or not a status update is received.

The standard option for scenarios requiring tracking is to use the WHOI Micro-Modem ACK bit. If the ACK bit setting is enabled at the time of a Micro-Modem transmission, the receiving modem automatically replies with a mini-packet confirming successful receipt. If a command is not received by a hardware asset, another can be sent until one is successful. The ACK bit's simplicity of implementation makes it particularly useful for short command sequences between heterogeneous assets using a common language such as the WHOI CCL [96].

Similar to command scenarios, data transfer applications such as collaborative SLAM require data transmission performance tracking to ensure complete transfer of critical data and to eliminate expensive redundant transmissions. ACK performs well, but its cycle time increase proves costly at high bandwidth transmissions.

The minimum cycle time for a 32 byte rate 0 FSK message is about six seconds [7][47]. However, this was applied to small mission sizes (200m path length); larger mission sizes require longer cycle time to account for acoustic travel time. 10s cycle time is commonly used for rate 0 and 1 messages [49]. Rate 5 PSK messages take even longer due to the slow serial transfers to and from the modem which are theoretically 4.9s [97]. In practice 8s was measured on the IVER2 modem. [46] uses 15 seconds for rate 5. The cycle times for this work were assumed at 10s for rates 0,1,4, and 15s for rate 5.

An ACK bit requires minimal time to generate; the cycle time allocated to it is for the acoustic travel time. The large simulated mission in Section 4.2 has maximum inter-vehicle ranges of 2800m, which is slightly less than 2.0 s of acoustic travel time. Therefore, a 2.0 s cycle time allocation for the ACK bit was assumed for cost comparisons. The bandwidth cost of using the ACK bit is proportional to the time used to send the ACK bit that could have been used to transmit data at the going rate if no ACK bit were used. This can be calculated via the following equation:

$$c = \left[\frac{d}{t_{No\ ACK}} - \frac{d}{t_{ACK}}\right] P(s|r) \tag{9}$$

where

c = *bandwidth cost* (B/s)

d = *data transferred per packet* (B)

$t_{No\ ACK}$ = *cycle time without ACK bit* (s)

t_{ACK} = *cycle time with ACK bit* (s)

$P(s|r)$ = *likelihood of successful transmission given current data rate*

This impact is captured in Figure 35 for the low TL case of highest modem success rate as detailed in Table 4. Note the significant increase in cost for higher data rate communications. As TL increases, the expected success of the modem at all rates is reduced, lowering the total but not proportional ACK bit cost.

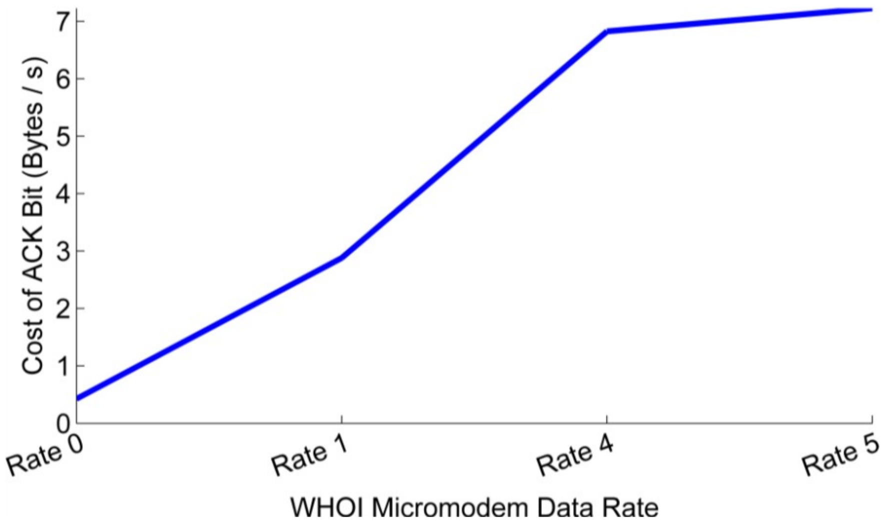


Figure 35: Cost of ACK bit

Soft data transmission performance tracking has the potential to significantly reduce bandwidth costs compared to the hardware ACK bit. Fallon and Leonard implement a cheap system for position estimation sharing [49], but is limited to synchronizing a small number of repeatedly updated data values. For tracking of arbitrary data as is required for collaborative SLAM, sonar image snippet transfers, or other high-data application, a more generic tracker is required.

5.2.1. TRACKER DESIGN

A modular, decentralized software data tracking system was developed to reduce the bandwidth cost of data tracking. The system has two components. First, a database is implemented on each vehicle to track each data frame the vehicle transmitted and whether it was successfully received by the intended recipient(s). Second, tracking headers are included with each transmission to notify collaborators when the vehicle successfully receives packets – these provide feedback similar to the ACK bit but require data overhead instead of additional cycle time.

When a vehicle (designated AUV1) first sends a packet, the frame contents are added to its tracking database and their received status is set as *unknown.* AUV1 then parses the tracking headers of messages it receives from collaborators. Once it receives confirmation that its previous transmission failed or was received, the received status of its database frame entries is accordingly updated. Once AUV1's collaborators are aware that AUV1 has received feedback on a packet, that feedback is removed from the tracking headers the collaborators transmit.

A key requirement of the data tracking system is minimized data overhead, even in poor acoustic conditions with high packet losses. This is accomplished by designing the tracking headers to explicitly report received packets, but only implicitly report unreceived packets. The tracking headers report how many communication cycles ago each successful packet was received. Implicitly, any packet sent on a communication cycle not reported is assumed failed. This method links tracking header build-up to the unlikely scenario that one collaborator consistently fails to receive packets but succeeds in sending them, rather than to the likely scenario of mutual frequent losses or long blackouts. The data tracking system process is graphically illustrated in Figure 36.

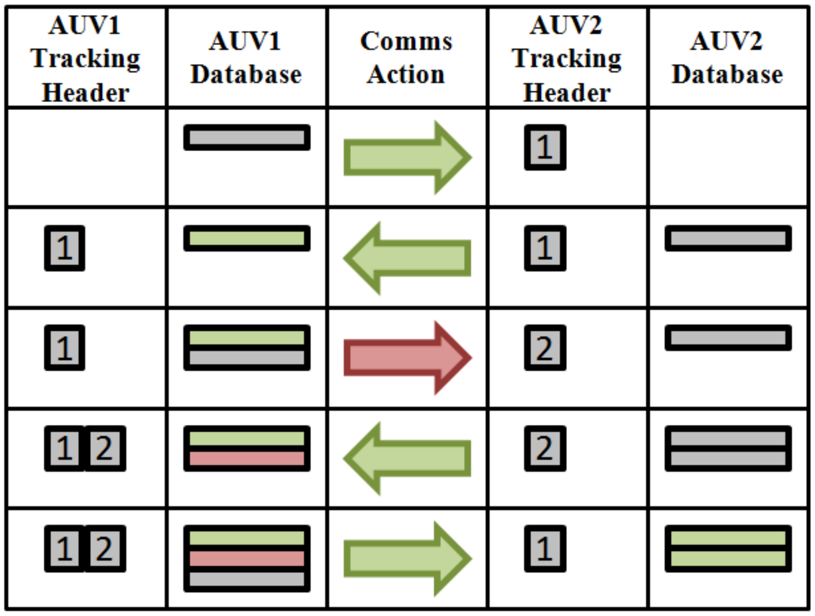


Figure 36: Snapshot of data tracking system.

Each row represents the state of two collaborating AUVs before one transmits and after the other receives. Each arrow under *Comms Action* represents a transmission, with green arrows successful and red failed. Each block under *Database* represents the host AUV's record of one frame it sent, with green representing successful receipt, red failed, and grey unknown. Each block under *Tracking Header* represents one element in the tracking header queue sent by the host AUV, with the number inside representing how many cycles ago the tracked message was received.

The data tracker was only tested for the two vehicle cases as detailed in Section 4.4.3. However, it can be expanded to a multi-vehicle case with minimal adjustment. The single collaborator case described above uses a database to track sent messages with possible values of "unknown", "received", or "failed". This single database field would be extended into an array where the status of message receipt by each collaborator is tracked independently. No additional overhead would be introduced into the acoustic headers: regardless of the number of collaborators, a TDMA network only has one agent transmitting at any given cycle. Therefore, the "cycles ago" variable by which vehicles report when packets were received would implicitly carry source agent information as the TDMA schedule would be known to all collaborators.

5.2.2. Performance Analysis

Each queue entry in the *Tracking Header* represents *cycles_ago* using an $\epsilon(0,63)$ integer (7 bit cost when DCCL encoded [88]). Each entry represents *frame_success* using a $\epsilon(0,7)$ integer (4 bit cost in DCCL) for WHOI Micro-Modem data rates 0, 1, and 4 and an $\epsilon(0,255)$ integer (9 bit cost in DCCL) for WHOI Micro-Modem data rate 5. The total size of each queue entry for WHOI modem rates 0, 1, and 4 is 13 bits, while for rate 5 the cost increases to 16 bits. The mean bandwidth cost (B/s) of the software data tracker is calculated as follows:

$$\frac{queue\ element\ size\ (B) x\ mean\ queue\ length}{cycle\ time\ (s)} \tag{10}$$

Mean and maximum tracking header queue lengths for all simulation scenarios are detailed in Figure 37: Note that each queue entry costs 1.625 Bytes for WHOI Micro-Modem rates 0, 1, and 4 while the cost increases to 2.0 Bytes for rate 5. The worst-case mean cost (scenario 4 with a 1.682 mean queue length) is used to calculate the software tracking bandwidth cost, shown compared to the ACK bit cost in Figure 38.

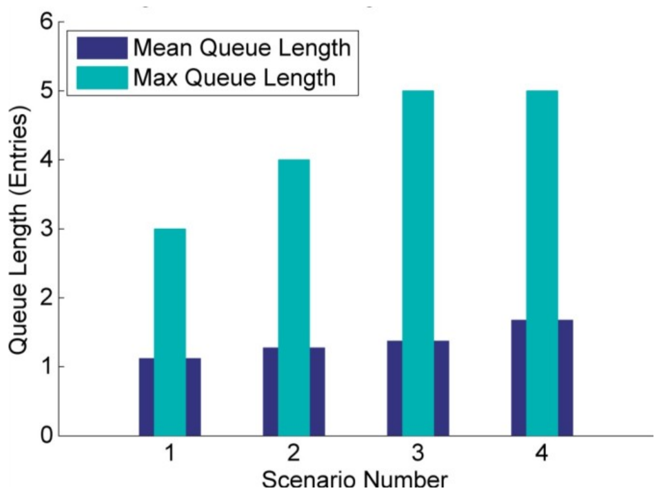


Figure 37: Tracking header queue length

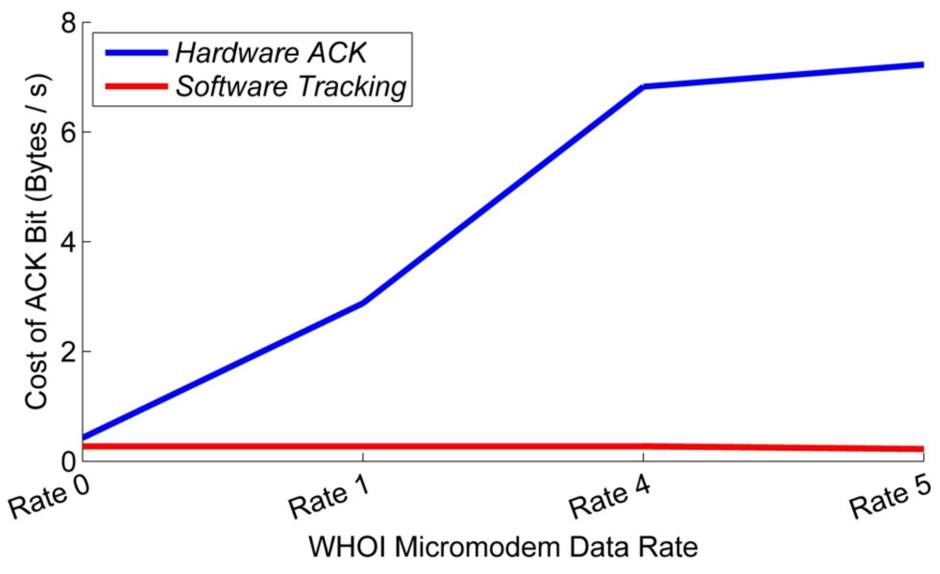


Figure 38: Cost of data tracking

The cost comparison between the software and ACK bit tracking is comparable at data rate 0, but for higher bandwidth PSK data rates the cost of software tracking remains essentially constant while the cost of the hardware ACK bit scales with bandwidth. The system provides 100% feedback on reception status within the constraint that no tracking header queue entries are allowed to age past the upper limit set on the *cycles_ago* variable. *Cycles_ago* was set to a maximum of 63 for the trials: any tracking header not received within 63 cycles is lost and the delivery status of the corresponding packet remains *unknown* forever. This maximum could be increased arbitrarily at the cost of one extra queue header bit each time it doubles. This is in comparison to the ACK bit which is susceptible to failed reception in the same way as are standard acoustic packets.

5.3. Modem Rate Selector

The rate selector is central to the design of an adaptive communications control algorithm for collaborative work. When a vehicle reaches the transmit phase of the communication cycle, the rate selector decides whether or not to transmit and, if yes, which data rate should be used. The acoustic environment is both temporally and geographically variable: to make good decisions, the rate selector must account for the goodness of the acoustic channel between source and destination vehicles at each moment of transmission.

5.3.1. *DESIGN CONSTRAINTS*

The literature mainly focuses on adaptive inter-AUV communication behaviours that use physics-based models to quantitatively estimate the actual acoustic signal loss at the receiver location. Schneider estimates TL by using BELLHOP to perform online ray tracing calculations throughout the mission [4]. Chen and Pompilli estimate the TL by using the Urick model [5] shown below to provide a basic TL approximation.

$$TL = \kappa \cdot 10log(l) + \alpha(f) \cdot l \tag{11}$$

where

κ = *spreading factor* (value set to 1.5)

$\alpha(f)$ = *frequency dependent absorption coefficient* (dB/m)

l = *distance between transmitter and receiver* (m)

Schneider's on-line ray tracing shows promise for applications in depths as shallow as 110m, but it relies on knowledge of the current SSP and assumes a flat bottom [4]. These parameters are not necessarily available for an NMCM mission. Chen and Pompilli's Urick model approximation is overly generic in that it does not take environmental and acoustic geometry into consideration. Neither method accounts for time-variant ambient noise sources.

The constraints placed by NMCM on AUV environmental knowledge reduces practicality of a meaningful and real-time characterization of TL using an on-line physical model of the environment. This led the rate selector design to move from an attempt at acoustic transmission modeling to a general and forgetful characterization of the current channel state. Of the four factors considered in Section 4.4, mission size and ambient noise have high impact on mean communications performance while that of vehicle type and SSP is lower. The low influence of the SSP means no characterization was attempted. Vehicle type also had a low influence, but its characterization is not practically discreet from a vehicle-specific characterization of ambient noise. Therefore, on-line acoustic condition characterization is focused on characterizing (A) position and (B) vehicle-specific time variant ambient noise in order to optimize (1) bandwidth, (2) frequency of success, and (3) duty cycle.

5.3.2. ARCHITECTURE

A modem rate selector has two key elements: (1) a method to characterize the current channel conditions and (2) a method to select modem settings based on the current characterization. The second element – the actual decision making process – need not be complex. A state machine with a lookup table correlating TL to optimal data rate would suffice. The value matching of the lookup table would be dependent on the relative priority of communication performance metrics and would not change throughout the mission. The adaptivity would come from reacting to the current environment with a reaction pre-computed to be optimal given the current channel conditions.

The ideal decision making engine would have access to the actual TL of the current acoustic channel and could use that to make an optimal rate selection. Unfortunately, as stated in the above section, the channel TL is not available to the NMCM AUV. However, the concept of a state machine using pre-computed lookup relations to perform optimal actions given the current environment is still viable.

The first input available for channel characterization is distance from source to receiver as fed into the engine by the position estimator. Distance is one of the key influences on TL; in lieu of actual TL calculations it may be used to approximate link quality. Therefore, a rudimentary adaptive behaviour would change the state machine lookup from TL thresholds (ideal) to distance thresholds (practical). At each decision point, the rate selector chooses the transmission type corresponding to the bin that encompasses the current distance to the intended recipient(s). However, distance bins cannot be pre-computed and statically maintained throughout a mission: a variety of factors influence TL in addition to distance. Distance thresholds pre-computed for one scenario would be poorly optimized for an environment with different noise, bottom cover, etc. Therefore, the threshold bins are preset at the mission start but must be adapted on-line throughout the mission.

The adaptation of the distance bins could be controlled by any number of online feedback methods: SNR estimation, simple TL approximation calculations, success rates, etc. Using current success rates as a feedback presents an approach that is more akin to reinforcement learning techniques than to any physical characterization. Reinforcement learning is a form of supervised learning whereby an agent makes decisions that are rewarded or punished. As the agent builds a history of decisions vs rewards, it learns which future decisions are more likely to gain reward [98]. While learning techniques can be slower than engineered controls in tightly controlled or well-understood environments, they provide versatility in highly unpredictable or poorly characterized environments such as the acoustic channel.

The reinforcement adaptation of distance threshold bins is controlled by setting desired success ratios for transmissions at each data rate. At each cycle iteration, the current success rate is compared to the desired rates and the thresholds are incremented or decremented. Acoustic conditions can change throughout the course of a day or mission, so the learning behaviour is made forgetful by only calculating the current success over the most recent subset of transmissions.

This learning is suitable for longer-term communication metrics such as mean duty cycle and mean bandwidth, but it is not well suited to the often urgent communication metric of frequent status updates. A model is required that can increasingly prioritize success, even at the cost of long-term metrics, in the event of a recent success drought. This issue is addressed by co-maintaining two arrays of distance bins corresponding to each transmission at each WHOI Micro-Modem data rate and also no transmission. One array of bins represents a conservative, high-reliability mode for status updates and the other a high-bandwidth and low duty cycle mode "data" mode to improve long-term metrics. Data bins are used by default unless performance is degraded beyond a threshold, at which point status bins are used until connectivity is regained. The architecture is shown in Figure 39 where uppercase symbols represent preset parameters and lowercase represent calculated variables.

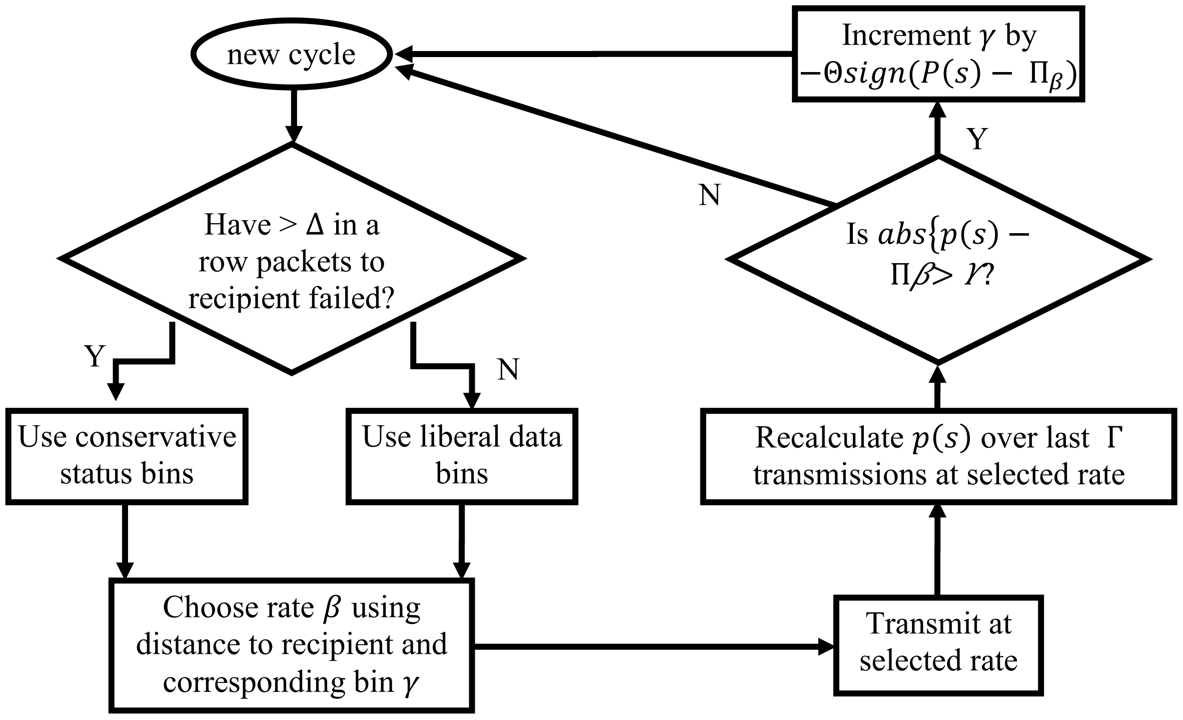

new cycle
Increment γ by $-\Theta sign(P(s) - \Pi_{\beta})$
Y
N
Have > Δ in a row packets to recipient failed?
Is $abs\{p(s) - \Pi\beta > \Upsilon$?
Y
N
Use conservative status bins
Use liberal data bins
Recalculate $p(s)$ over last Γ transmissions at selected rate
Choose rate β using distance to recipient and corresponding bin γ
Transmit at selected rate

Figure 39: Rate selector flow

Symbols used are as follows:

Δ = *status transmission threshold* (controls # sequential failed transmission attempts before moving to the low-rate bins)

β = *WHOI Micro-Modem data rate* (selected for transmission)

γ = *status bin or data bin* (which depends on the current distance between the source and receiving AUV)

$p(s)$ = *current estimation of probability of success* for β.

Γ = *memory length* (controls # previous transmissions incorporated into the estimate of current *p(s)* for β)

Π = *target success ratios* (represents desired success ratio for each Micro-Modem data rate Π_β = desired success ratio for β)

Υ = *target success tolerance* (values of $p(s)$within Υ of Π_β are considered to be at the target ratio)

Θ = *bin increment value* (if $p(s)$ is not within Υ of Π_β, γ is increased Θ in a direction to encourage $p(s)$ to approach Π_β)

The adaptive engine is driven by the need for the $p(s)$ of each data rate to approach the target success ratios Π. This method is robust to environmental change. For example, a poor acoustic environment will initially yield a $p(s)$ below the desired success ratio Π_β. In this case, the distance of bin γ will be decreased, meaning subsequent communications at that rate will take place closer to the recipient, increasing $p(s)$. The reverse occurs in an unusually good environment. It is desirable for the rate selector to target success ratios directly as $p(s|\beta)$ can be directly translated to high-level mission communication outcomes such as desired frequency of status updates, duty cycle power usage, etc.

The rate selector described herein has been only tested in the two vehicle cases described in Section 4.4.3. The basic design of the algorithm does not change when additional collaborators are added: the distance bins are still adapted using the success based learning technique. However, there is a key difference: mixing of selector decisions must be performed. When additional collaborators are added, there is an optimal solution for transmission to each collaborator; these may or may not be the same. If different optimal solutions exist, an intelligent mixer must be added to choose between them. As the number of collaborators increases, optimal decisions will increasingly favour bandwidth and update frequency rather than reduced duty cycle: each transmission provides multiple chances for success which are all forfeited in a non transmission event.

5.3.3. Simulator Driven Parameter Selection

The pre-set parameters described in the previous section significantly impact performance. Hundreds of distinct parameter combinations were run on missions using the simulation environment to determine the impact and optimal values of each. Table 8 shows the ranges tested for each parameter and the selected value that is optimal given the performance criteria established in Table 7. In the event different performance criteria are desired, the tunings may be re-done to obtain values for the new criteria.

Table 8: Rate selector parameter tuning

Parameter	Range of Values Tested in Simulation	Optimal Value
Target Success Ratio Π_5	0-0.5	0[1]
Target Success Ratio Π_4	0.2-0.7	0.3
Target Success Ratio Π_1	0.3-0.8	0.4
Target Success Ratio Π_0	0[2]	0
Memory Length Γ	10-20 transmissions	10
Target Success Tolerance Υ	0.05[3]	0.05
Bin Increment Value Θ	5-20 meters	5
Status Transmission Threshold Δ	2-4	3

[1] If cycle time is held constant, having nonzero target success ratios for data rate 5 proves superior to a zero value success ratio. However, the longer serial communication time required for rate 5 forces the entire network to jump from a 10s to 15s cycle time due to TDMA constraints. Taking this into consideration, 10s cycle times that do not use rate 5 prove superior.

[2]After performing transmission loss versus *p(s)* calculations shown in Figure 20, it was found that the *p(s)* of rate 0 communications closely matched the *p(s)* of rate 1 communications. With a factor of six improvement in bandwidth, rate 1 was always preferred over rate 0 in the rate selector design.

[3] Target success tolerance was not varied during simulations due to computing budget constraints.

The performance of the optimal parameters are compared to control communication schemes in the following sections. Three control behaviours are used. First, continual use of rate 0; this is one of the most commonly used WHOI Micro-Modem control schemes [49]. Second, continual use of rate 1; this is proposed by Fallon [49] as a bandwidth improvement over rate 0. Third, alternating use of rate 0 and rate 5: this is used by Schneider and Schmidt as a compromise between high bandwidth and reliability [3]. For reference, also included are a set of parameters that are optimal in the event rate 5 is implemented; however, the longer cycle time inherent with rate 5 reduces the effectiveness of this solution.

5.4. PERFORMANCE

The adaptive algorithm's performance was tuned using the relative valuation of the three metrics detailed in Table 7. In the following three sections, its raw performance in each of the metric categories of bandwidth, update frequency, and duty cycle are compared to that of the control behaviours.

5.4.1. BANDWIDTH

A comparison of mean successful data throughput is shown in Figure 40. The adaptive algorithm without rate 5 proves superior to all control methods, even the intermittent use of high-bandwidth rate 5. Important to note is how it uses high bandwidth rates to capitalize on the good acoustic conditions of the small, low-noise scenario 1 while reducing rates to retain connectivity in the large, noisy scenario 4. Overall, bandwidth performance is increased over the best control by between 24% for scenario 4 and 106% for scenario 1.

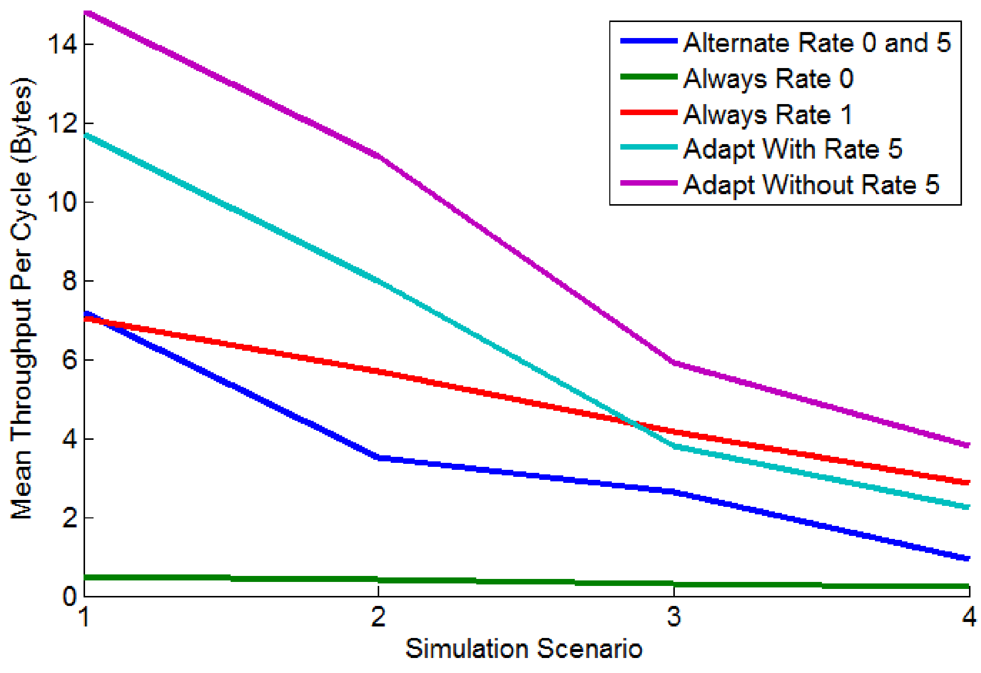


Figure 40: Adaptive algorithm bandwidth performance

5.4.2. STATUS UPDATE PERIOD

Mean period between successful transmissions was examined as shown in Figure 41. The adaptive algorithm without rate 5 proves superior to alternating of rates 0 and 5, but inferior to the other controls by between 33% and 78%. While not ideal, this is acceptable because of performance gains in the other two categories. Adaptive rate control does not improve the underlying goodness of the acoustic channel; it merely optimizes the way in which it is used. Specifically, it allows the operator to achieve a desired combination of performance in the mutually detrimental categories of bandwidth, update frequency, and power usage. This particular optimization prioritized bandwidth and low power usage while ensuring the frequency of status updates was not unreasonably low.

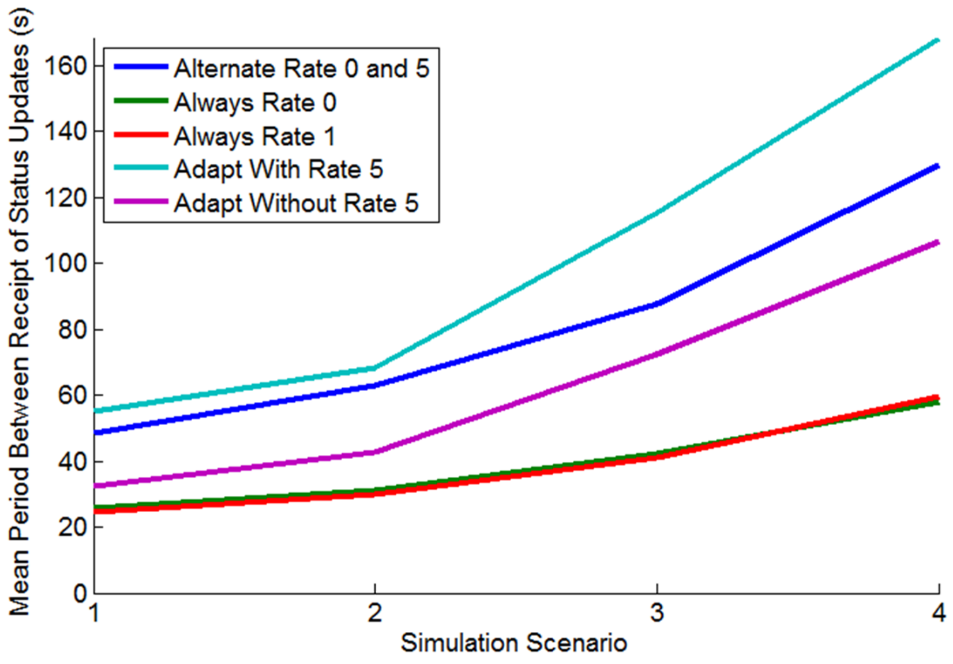


Figure 41: Adaptive algorithm status period performance

5.4.3. Power Usage

Mean duty cycles are shown in Figure 42. Both adaptive algorithms are comparable to the continual duty cycle control methods in the small mission size, but drop to 62% of the control values for the larger mission. This is because in smaller missions good connectivity can almost always be achieved, while in the larger missions the power cost frequently exceeds the slight chance of success.

Steady-state modem power use is so low compared to transmit power that overall communication system power draw downscales linearly with transmission duty cycle. In addition to costing power, modem transmissions corrupt sidescan sonar images through acoustic interference and also increase the potential for a clandestine AUV to be detected by hostile sensors. Reducing the communications duty cycle not only benefits power draw but also these other key NMCM aspects.

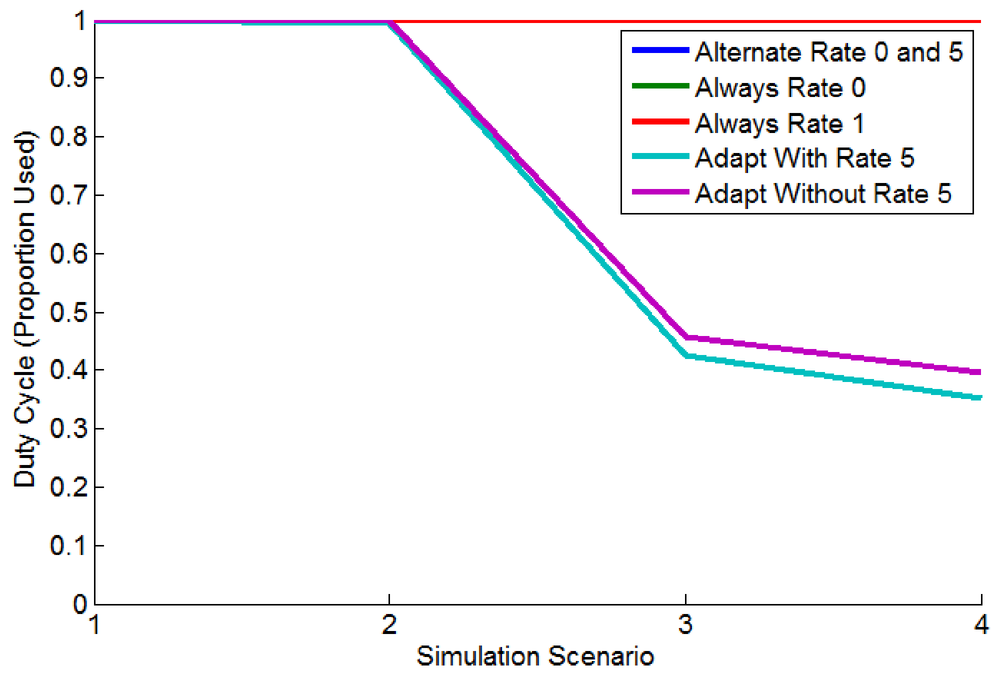


Figure 42: Adaptive algorithm duty cycle performance

5.5. Discussion of Results

This work was begun with the objective of developing a method to use existing underwater communication systems more intelligently to optimize use of the acoustic channel. Specifically, this method must apply to shallow water multi-AUV NMCM operations with the goal of creating a framework on which data intensive collaborative SLAM systems may be run. This broad objective is developed and tested for using the hardware available at DRDC. Therefore, this work focuses on the optimal use of the WHOI Micro-Modem on IVER2 or similar vehicles running the MOOS software. Field trials were used to characterize the Micro-Modem's performance as a function of data rate. This was used to develop a simulation environment in which all algorithms were developed and tested. Unfortunately, logistical realities precluded the final validation on field hardware in a timely fashion, meaning that the following results are generated through simulation.

The first optimization feature implemented is to replace the Micro-Modem's built in transmission success feedback message with a more lightweight software success tracker as described in Section 5.2. The built in feature works well but extends the 10 second TDMA cycle time to 12 seconds; a costly 17% time increase per unit of data. The software feedback tracking system does not extend cycle time and instead operates with a mean 2.7 Byte overhead. This is significantly cheaper than the cycle time increase, particularly for higher data rates. Additionally, the software tracker features 100% packet feedback in comparison to the generally reliable hardware feedback message which is nonetheless still susceptible to packet loss in adverse conditions. The only trade-off is complexity of implementation that requires a full autonomy system controlling the modem, as compared to the simple interface offered by the hardware acknowledgement.

This feature is significant because a data-intensive multi-agent system requires data tracking for optimal performance, particularly given the often poor connectivity of underwater networks. If a data packet is not received, it may need to be repeated until it is. Conversely, it would be wasteful for a received packet to be repeated unnecessarily. However, the hardware tracking feature significantly cuts into the data throughput budget: using the software tracker significantly improves bandwidth in all scenarios – both fixed and adaptive rate control – with no cost to connectivity or duty cycle.

The second optimization feature implemented is the adaptive rate selector as detailed in Section 5.3. Unlike the data tracker discussed above, the adaptive rate selector cannot unilaterally improve performance in all metrics it affects. Each WHOI Micro-Modem data rate has a distinct relationship between reliability and bandwidth, with higher bandwidths typically providing lower reliability. An intelligent rate selector, instead of improving every communication metric, acts to tune the rate selections to optimize favoured metrics. The three key metrics for NMCM missions, identified in Section 4.5, are: mean bandwidth (50% value), mean period between status updates (20%

value), and mean duty cycle (30% value). Given the relative values of metrics, an adaptive behaviour can be tuned to optimize total value, typically by increasing performance in one or two metrics while limiting the performance hit to the third metric. The behaviour optimized for the above metrics yielded results as detailed in Table 9, Table 10, and Table 11.

Table 9: Results summary for data throughput

	Performance: adaptive behaviour	**Performance: best control case**	**Percent adaptive different from control**
Most favorable acoustic conditions	14.8 B/s	7.2 B/s	106% better data
Least favorable acoustic conditions	3.6 B/s	2.9 B/s	24% better data

Table 10: Results summary for connectivity (period between status updates)

	Performance: adaptive behaviour	**Performance: best control case**	**Percent adaptive different from control**
Most favorable acoustic conditions	32.4s	24.4s	33% longer between status updates
Least favorable acoustic conditions	102.6s	57.8s	78% longer between status updates

Table 11: Results summary for duty cycle (proportion of available transmit opportunities used)

	Performance: adaptive behaviour	Performance: best control case	Percent adaptive different from control
Most favorable acoustic conditions	1.00	1.00	same power used
Least favorable acoustic conditions	1.00	0.38	62% less power used

The data throughput is the most heavily weighted metric due to its importance for implementing the high-bandwidth underwater SLAM algorithm. The adaptive behaviour more than doubles total throughput in good acoustic conditions, but the length of period between status updates is increased by a third, reducing connectivity. For favourable acoustic conditions both the control and adaptive behaviours transmit at 100% duty cycle: likelihood of success is high enough that not transmitting is too costly to the bandwidth and connectivity metrics. In poor acoustic conditions the adaptive behaviour still outperforms control but only by 24%. This comes at a cost of a 78% longer period between status updates. If a lowered duty cycle were of no value, data rates in adverse conditions would be slightly increased from these results, and connectivity would be less poor. However, given the valued power savings of not transmitting, any packet with a sufficiently low likelihood of success becomes a duty cycle liability: the adaptive behaviour reduced its duty cycle (power usage) by 62% in adverse acoustic conditions.

The importance of tunable communication metric optimization is that it provides a flexibility not previously implemented in collaborative AUV missions. Current, fixed low-rate behaviours are optimized only for connectivity. The framework presented in Section 5.3 allows operators the ability to customize acoustic operations. The application addressed herein is NMCM operations, where bandwidth is the most important metric but connectivity and duty cycle are also important. Other potential applications include long-range large area mapping that may prioritize energy savings and connectivity over bandwidth, or a statically deployed sensor suite that values bandwidth to the exclusion of the other metrics.

It should be noted that the field trials and simulations discussed herein were conducted with a two agent setup in order to reduce scope for initial development. However, all developed applications are scalable to the multi-vehicle case with either few or no modifications. This is critical as interest in fleet-based AUV activities continues to climb. It is highly recommended that development continue and multi-vehicle implementations of the algorithms described herein be performed in simulation and field trial.

In summary, the results generated from testing the adaptive behaviour and data tracking systems demonstrate key advances in line with the initial objective of optimizing the way existing modem hardware interacts with the acoustic channel.

CHAPTER 6 CONCLUDING MATTER

6.1. CONCLUSIONS

An adaptive framework has been presented that provides AUV operators a method to optimize acoustic modem performance in NMCM and other shallow-water collaborative missions.

Field trials were conducted to characterize the relative likelihood of success between the four WHOI Micro-Modem data rates in a two vehicle scenario. The results of this are used to develop a MATLAB vehicle and acoustics simulation environment for behaviour development. A forgetful adaptive behaviour is implemented to characterize the current quality of the acoustic environment, and transmission decisions are based on adaptive distance thresholds. Live state estimation is applied to collaborating vehicles to predict their current positions during communication droughts.

The framework has been applied to a variety of operating environments and yielded promising results in simulated two vehicle collaborative AUV missions. Bandwidth performance is increased by between 24% for poor condition to over 106% for excellent condition scenarios. Duty cycles remain similar for good conditions but are reduced by 62% in poor conditions, providing power savings. These improvements do come at a cost, with the period between status updates falling between 33% and 78%. The framework is tested between two vehicles in the simulations presented, but may be scaled to larger fleets in practise.

In summary, a communications groundwork for the support of data intensive multi-agent applications in a shallow water NMCM environment has been developed. This framework significantly improves the net data transfer between collaborating vehicles and also provides valuable lost data feedback. When paired with an intelligent data queuing system this framework has the potential to allow for the first successful implementation of underwater SLAM or a related distributed online mapping system.

6.2. Recommendations for Future Work

The adaptive framework has been ported from MATLAB simulation and integrated with MOOS-IVP enabled IVER2 AUVs. However, little in-water work has commenced due to lack of vehicle availability. Proposed steps for future development work are as follow:

1. Perform in-water debugging of MOOS code implementation. The control algorithm code has been ported from MATLAB to MOOS and installed on a single IVER2. The code has been tested thoroughly in a small acoustic test tank but only for one mission run in open water. That mission encountered two identified issues (and likely additional unidentified ones) that must be addressed before performance validation may begin.

    a. The clock on the IVER2 backseat computer runs slow; manual pre-missions synchronization is required (imprecise) and the clock slows appreciably throughout the mission. This causes a relative drift in the IVER2's TMDA cycle and increases the chance of packet collision. The clock must be fixed to provide accurate time.

    b. The trial was conducted to compare an implementation with mission preknowledge to one without. Therefore, the IVER2 had no preknowledge of where the deck modem was located. The deck modem had full preknowledge of the IVER2 mission path. Throughout the trial the algorithm on the deck functioned as intended. However, the IVER2 chose to not transmit at all. This is because it started with an inaccurate position estimate for the deck, thinking it was well outside of range. A bug prevented it from updating its estimate of the deck position when it received a new status message, leading it to continue thinking the deck was out of range for the mission duration. This bug correction must be validated in another field trial.

2. Perform first sea trials with a single AUV and single stationary transponder configuration to mimic the two vehicle setup used in simulation. Evaluate the adaptive behaviour's impact on bandwidth, connectivity, and duty cycle in a variety of conditions and environments. Make any modifications necessary to the learning agent to optimize results.

3. Scale from the dual-vehicle scenario discussed in this work to the multi-vehicle scenario. The algorithms detailed herein may be scaled to $N > 2$ vehicles, but to reduce scope only $N = 2$ vehicle cases were analyzed in this thesis. The move to multiple vehicle scenarios would take place in two stages.

    a. As described in this work, it is most efficient to thoroughly test and optimize an intelligent communications scheme in simulation before field trials. This is the recommended path for multi-agent communication protocol implementations. The simulation environment described in Section 4 should be used to compare the performance of a $N \epsilon \{3:M\}$ multi-vehicle fleet where M is a reasonable upper fleet size.

    b. The simulation results should be confirmed through multi-vehicle field trials. DRDC currently has three IVER2 AUVs, one Micro-Modem equipped Unmanned Surface Vehicle (USV), and one deck modem for a total of five possible agents for multi-vehicle trials. Logistic realities will likely preclude the use of all five simultaneously, but a three or four agent scenario is reasonable.

4. Investigate the potential large-fleet payoff of an intelligent message routing scheme similar to what Pompilli [5] implements on glider fleets. As fleet size increases, it becomes problematic for a transmitting agent to find a communication rate optimal for all of its recipients. Instead of attempting to optimize transmission rates for all recipients, it may be more efficient to optimize for the recipients most

likely to succeed and incorporate routing information to allow data forwarding to more distant collaborators in future transmissions.

REFERENCES

[1] S. Sariel, T. Balch and N. Erdogan, "Naval Mine Countermeasure Missions," *Robotics & Automation Magazine, IEEE,* vol. 15, pp. 45-52, 2008.

[2] M. Seto, L. Paull and S. Saeedi, "Introduction to Autonomy for Marine Robots," in *Marine Robot Autonomy*, M. Seto, Ed., Springer, 2012, pp. 1-40.

[3] T. Schneider and H. Schmidt, "Unified Command and Control for Heterogeneous Marine Sensing Networks," *Journal of Field Robotics,* vol. 27, no. 6, pp. 876-889, 2010.

[4] T. Schneider and H. Schmidt, "Model-Based Adaptive Behavior Framework for Optimal Acoustic Communication and Sensing by Marine Robots," *Oceanic Engineering, IEEE Journal of ,* vol. 38, no. 3, pp. 522-533, 2013.

[5] B. Chen, P. Hickey and D. Pompili, "Trajectory-Aware Communication Solution for Underwater Gliders Using WHOI Micro-Modems," in *Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, 2010. pp. 1-9.

[6] J. Crowell, "Small AUV for Hydrographic Applications," in *OCEANS 2006*, 2006. pp. 1-6.

[7] N. Hallin, H. Taheri, J. Horn, M. ORourke and D. Edwards, "Architecture and applications of Language-Centered Intelligence for unmanned underwater vehicles," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 2011. pp. 3130-3136.

[8] P. Newman, "MOOS - Mission Orientated Operating Suite," Boston, USA.

[9] M. Quigley, B. Gerkey, K. Conle, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "ROS: an open-source Robot Operating System," in *In ICRA Workshop on Open Source Software* , 2009.

[10] A. Makarenko, A. Brooks and T. Kaupp, "Orca: Components for Robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'06)*, Beijing, China, 2006.

[11] R. Eustice, H. Brown and A. Kim, "An overview of AUV algorithms research and testbed at the University of Michigan," in *Proceedings of the IEEE/OES Autonomous Underwater Vehicles Conference*, 2008. pp. 1-9.

[12] J. C. Kinsey, R. M. Eustice and L. L. Whitcomb, "A Survey of Underwater Vehicle Navigation: Recent Advances and New Challenges," in *Conference of Manoeuvering and Control of Marine Craft*, Lisbon, Portugal, 2006.

[13] L. Medagoda, S. B. Williams, O. Pizarro and M. V. Jakuba, "Water column current aided localisation for significant horizontal trajectories with Autonomous Underwater Vehicles," in *OCEANS 2011*, 2011. pp. 1-10.

[14] M. V. Jakuba, O. Pizarro and S. B. Williams, "High resolution, consistent navigation and 3D optical reconstructions from AUVs using magnetic compasses and pressure-based depth sensors," in *OCEANS 2010*, 2010. pp. 1-9.

[15] M. Hunt, W. Marquet, D. Moller, K. Peal, W. Smith and R. Spindel, "An Acoustic Navigation System," Woods Hole, 1974.

[16] D. R. C. Philip, "An Evaluation of USBL and SBL Acoustic Systems and the Optimization of Methods of Calibration - Part I," *The Hydrographic Journal,* vol. 108, pp. 18-25, April 2003.

[17] E. Gallimore, J. Partan, I. Vaughn, S. Singh, J. Shusta and L. Freitag, "The WHOI micromodem-2: A scalable system for acoustic communications and networking," in *OCEANS 2010*, 2010. pp. 1-7.

[18] R. M. Eustice, H. Singh and L. L. Whitcomb, "Synchronous-clock, one-way-travel-time acoustic navigation for underwater vehicles," *J. Field Robot.,* vol. 28, pp. 121--136, jan 2011.

[19] R. M. Eustice, L. L. Whitcomb, H. SINGH and M. Grund, "Experimental Results in Synchronous-Clock One-Way-Travel-Time Acoustic Navigation for Autonomous Underwater Vehicles," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007. pp. 4257-4264.

[20] D. Ribas, P. Ridao, J. Neira and J. D. Tardos, "SLAM using an Imaging Sonar for Partially Structured Underwater Environments," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006. pp. 5040-5045.

[21] I. Ruiz, S. de Raucourt, Y. Petillot and D. M. Lane, "Concurrent mapping and localization using sidescan sonar," *Oceanic Engineering, IEEE Journal of,* vol. 29, pp. 442-456, 2004.

[22] S. Barkby, S. Williams, O. Pizarro and M. Jakuba, "An efficient approach to bathymetric SLAM," in *Intelligent Robots and Systems, 2009, IEEE/RSJ International Conference on*, 2009. pp. 219-224.

[23] S. B. Williams, O. Pizarro, M. V. Jakuba, I. Mahon, S. D. Ling and C. R. Johnson, "Repeated AUV surveying of urchin barrens in North Eastern Tasmania," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010. pp. 293-299.

[24] S. B. Williams, P. Newman, G. Dissanayake and H. Durrant-Whyte, "Autonomous underwater simultaneous localisation and map building," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 2000. pp. 1793-1798 vol.2.

[25] S. Barkby, S. B. Williams, O. Pizarro and M. V. Jakuba, "Bathymetric SLAM with no map overlap using Gaussian Processes," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011. pp. 1242-1248.

[26] Department of the Navy United States of America, "The Navy Unmanned Undersea Vehicle (UUV) Master Plan," 2004.

[27] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *Robotics Automation Magazine, IEEE,* vol. 13, pp. 99 -110, june 2006.

[28] M. Csorba, "Simultaneous Localisation and Map Building," Oxford.

[29] M. W. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *Robotics and Automation, IEEE Transactions on,* vol. 17, pp. 229 -241, jun 2001.

[30] J. W. Fenwick, P. M. Newman and J. J. Leonard, "Cooperative concurrent mapping and localization," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002. pp. 1810-1817 vol.2.

[31] A. Melim and M. West, "Towards autonomous navigation with the Yellowfin AUV," in *OCEANS 2011*, 2011. pp. 1 -5.

[32] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2002. pp. 593-598.

[33] A. Kim and R. Eustice, "Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009. pp. 1559-1565.

[34] R. Eustice, H. Singh, J. Leonard, M. Walter and R. Ballard, "Visually navigating the RMS Titanic with SLAM information filters," in *in Proceedings of Robotics: Science and Systems*, 2005. pp. 57-64.

[35] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, Boston: MIT Press, 2005.

[36] S. Williams, "Efficient Solutions to Autonomous Mapping and Navigation Problems," Sydney, 2001.

[37] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *Robotics Automation Magazine, IEEE,* vol. 13, pp. 108 -117, sept. 2006.

[38] J. J. Leonard and H. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," in *Proceedings of the Ninth International Symposium on Robotics Research*, London, Springer-Verlag, pp. 169-176.

[39] P. Newman, "On The Structure and Solution of the Simultaneous localization and map building problem," Sydney, Australia, 1999.

[40] J. L. Leonard, R. N. Carpenter and H. J. Feder, "Stochastic mapping using forward look sonar," *Robotica,* vol. 19, pp. 467--480, aug 2001.

[41] P. Newman and J. Leonard, "Pure range-only sub-sea SLAM," in *Proceedings of the IEEE Conference on Robotics and Automation (ICRA '02)*, 2003. pp. 1921-1926 vol.2.

[42] I. Mahon and S. Williams, "SLAM using natural features in an underwater environment," in *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, 2004. pp. 2076-2081 vol. 3.

[43] A. Burguera, Y. Gonzalez and G. Oliver, "Underwater SLAM with robocentric trajectory using a mechanically scanned imaging sonar," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011. pp. 3577 -3582.

[44] D. Ribas, P. Ridao, J. D. Tardos and J. Neira, "Underwater SLAM in a marina environment," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007. pp. 1455-1460.

[45] A. C. Koh, W. S. Wijesoma, S. L. Pua, K. W. Lee and B. Kalyan, "Shallow waters SLAM experiments on meredith AUV using forward looking sonar," in *OCEANS 2009, MTS/IEEE Biloxi - Marine Technology for Our Future: Global and Local Challenges*, 2009. pp. 1-6.

[46] F. Maurelli, P. Patron, J. Cartwright, J. Sawas, Y. Petillot and D. Lane, "Integrated MCM missions using heterogeneous fleets of AUVs," in *OCEANS, 2012*, 2012. pp. 1 -7.

[47] T. Bean, G. Beidler, J. Canning, D. Odell, R. Wall, M. O'Rourke, M. Anderson and D. Edwards, "Language and Logic to Enable Collaborative Behavior among Multiple Autonomous Underwater Vehicles," *International Journal of Intelligent Control and Systems,* vol. 13, no. 1, pp. 67-80, March 2008.

[48] A. Bahr, J. J. Leonard and M. F. Fallon, "Cooperative Localization for Autonomous Underwater Vehicles," *The International Journal of Robotics Research,* vol. 28, pp. 714-728, June 01 2009.

[49] M. F. Fallon, G. Papadopoulos and J. J. Leonard, "A measurement distribution framework for cooperative navigation using multiple AUVs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010. pp. 4256 -4263.

[50] S. B. Williams, G. Dissanayake and H. Durrant-Whyte, "Towards multi-vehicle simultaneous localisation and mapping," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, 2002. pp. 2743-2748.

[51] E. Nettleton, P. Gibbens and H. F. Durrant-Whyte, "Closed form solutions to the multiple platform simultaneous localisation and map building (SLAM) problem," in *Proc. SPIE 4051, Sensor Fusion: Architectures, Algorithms, and Applications IV, 428*, 2000. pp. 428--437.

[52] M. Walter and J. Leonard, "An Experimental Investigation of Cooperative SLAM," in *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, 2004.

[53] S. Thrun and Y. Liu, "Multi-robot slam with sparse extended information filters," in *Robotics Research, The Eleventh International Symposium, ISRR*, 2003.

[54] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani and H. Durrant-Whyte, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *The International Journal of Robotics Research,* vol. 23, pp. 693-716, August 01 2004.

[55] A. Howard, "Multi-robot Simultaneous Localization and Mapping using Particle Filters," *The International Journal of Robotics Research,* vol. 25, pp. 1243-1256, December 01 2006.

[56] L.-L. Ong, B. Upcroft, T. Bailey, M. Ridley, S. Sukkarieh and H. Durrant-Whyte, "A decentralised particle filtering algorithm for multi-target tracking across multiple flight vehicles," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006. pp. 4539-4544.

[57] B. Johnson, N. Hallin, H. Leidenfrost, M. ORourke and D. Edwards, "Collaborative mapping with autonomous underwater vehicles in low-bandwidth conditions," in *OCEANS 2009*, 2009. pp. 1-7.

[58] J. V. Diosdado and I. T. Ruiz, "Decentralised Simultaneous Localisation and Mapping for AUVs," in *OCEANS 2007*, 2007. pp. 1-6.

[59] E. Nettleton, S. Thrun, H. Durrant-Whyte and S. Sukkarieh, "Decentralised SLAM with Low-Bandwidth Communication for Teams of Vehicles," *Field and Service Robotics,* vol. 24, pp. 179--188, 2006.

[60] M. Pfingsthorn, A. Birk and H. Bulow, "An efficient strategy for data exchange in multi-robot mapping under underwater communication constraints," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010. pp. 4886-4893.

[61] J. J. Sojdehei, P. N. Wrathall and D. F. Dinn, "Magneto-inductive (MI) communications," in *OCEANS 2001*, 2001. pp. 513-519 vol.1.

[62] D. B. Kilfoyle and A. B. Baggeroer, "The state of the art in underwater acoustic telemetry," *Oceanic Engineering, IEEE Journal of,* vol. 25, pp. 4-27, 2000.

[63] J. Preisig, "Acoustic propagation considerations for underwater acoustic communications network development," *SIGMOBILE Mob. Comput. Commun. Rev.,* vol. 11, pp. 2--10, oct 2007.

[64] H. Riksfjord, O. T. Haug and J. M. Hovem, "Underwater Acoustic Networks - Survey on Communication Challenges with Transmission Simulations," in *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, 2009. pp. 300-305.

[65] M. Chitre, S. Shahabudeen, L. Freitag and M. Stojanovic, "Recent advances in underwater acoustic communications & networking," in *OCEANS 2008*, 2008. pp. 1-10.

[66] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski and K. Ball, "The WHOI micro-modem: an acoustic communications and navigation system for multiple platforms," *OCEANS, 2005. Proceedings of MTS/IEEE,* pp. 1086-1092 Vol. 2, 2005.

[67] D. Green, "Beyond underwater acoustic communications," in *OCEANS 2011*, 2011. pp. 1-6.

[68] D. Green, "Acoustic modems, navigation aids, and networks for undersea operations," in *OCEANS 2010*, 2010. pp. 1-6.

[69] M. Chitre, S. Shahabudeen and M. Stojanovic, "Underwater Acoustic Communications and Networking: Recent Advances and Future Challenges," *Marine Technology Society Journal,* vol. 42, no. 1, pp. 103-116, Spring 2008.

[70] M. Stojanovic, L. Freitag, J. Leonard and P. Newman, "A network protocol for multiple AUV localization," in *OCEANS 2002*, 2002. pp. 604-611 vol.1.

[71] E. M. Sozer, M. Stojanovic and J. G. Proakis, "Underwater acoustic networks," *Oceanic Engineering, IEEE Journal of,* vol. 25, pp. 72-83, 2000.

[72] J. Preisig, "Performance analysis of adaptive equalization for coherent acoustic communications in the time-varying ocean environment," *Journal of the Acoustic Society of America,* vol. 118, no. 1, pp. 263-278, July 2005.

[73] S. Shahabudeen, M. Chitre and M. Motani, "A multi-channel MAC protocol for AUV networks," in *OCEANS 2007*, 2007. pp. 1-6.

[74] J. Albiez, S. Joyeux and M. Hildebrandt, "Adaptive AUV mission management in under-informed situations," in *OCEANS 2010*, 2010. pp. 1-10.

[75] A. Guerrero-Gonzalez, F. Garcia-Cordova and J. Gilabert, "A biologically inspired neural network for navigation with obstacle avoidance in autonomous underwater and surface vehicles," in *OCEANS 2011*, 2011. pp. 1-8.

[76] G. Antonelli, F. Caccavale, C. Sansone and L. Villani, "Fault diagnosis for AUVs using support vector machines," in *Robotics and Automation, 2004. ICRA IEEE International Conference on*, 2004. pp. 4486-4491 vol.5.

[77] M. Seto, "Application of On-board Evolutionary Algorithms to Underwater Robots to Optimally Re-plan Missions with Energy Constraints," *Journal of Robotics,* vol. 2012, January 2012.

[78] P. Ridao, J. Yuh, J. Batlle and K. Sugihara, "On AUV control architecture," in *Intelligent Robots and Systems, 2000. (IROS). IEEE/RSJ International Conference on*, 2000. pp. 855-860 vol.2.

[79] M. Seto, J. Hudson and Y. Pan, "Three-Dimensional Path-Planning for a Communications and Navigation Aid Working Cooperatively with Autonomous Underwater Vehicles," in *Proceedings of the 2011 International Conference on Autonomous and Intelligent Systems*, Burnaby, BC, Canada, 2011. pp. 51-62.

[80] M. L. Seto, "On-line Learning with Evolutionary Algorithms towards Adaptation of Underwater Vehicle Missions to Dynamic Ocean Environments," in *IEEE Computer Society*, 2011. pp. 235-240.

[81] M. L. Seto and H. Li, "On-board AUV autonomy through adaptive fins control," in *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, 2010. pp. 933-939.

[82] L. Paull, S. Saeedi, H. Li and V. Myers, "An information gain based adaptive path planning method for an autonomous underwater vehicle using sidescan sonar," in *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, 2010. pp. 835-840.

[83] R. Sutton, C. Johnson and G. N. Roberts, "Depth control of an unmanned underwater vehicle using neural networks," in *OCEANS 1994*, 1994. pp. 121-125 vol.3.

[84] M. Carreras, J. Yuh, J. Batlle and P. Ridao, "A behavior-based scheme using reinforcement learning for autonomous underwater vehicles," *Oceanic Engineering, IEEE Journal of,* vol. 30, pp. 416-427, 2005.

[85] H. Kawano and T. Ura, "Dynamics control algorithm of autonomous underwater vehicle by reinforcement learning and teaching method considering thruster failure under severe disturbance," in *ntelligent Robots and Systems, 2001.IEEE/RSJ International Conference on*, 2001. pp. 974-979 vol.2.

[86] L. Freitag, "Micro-Modem Software Interface Guide," Woods Hole Oceanographic Institution, Woods Hole, 2013.

[87] M. Benjamin, P. Newman, H. Schmidt and J. Leonard, "An Overview of MOOS-IvP and a Users Guide to the IvP Helm Autonomy Software," MIT, Cambridge, 2010.

[88] T. Schneider and H. Schmidt, "The Dynamic Compact Control Language: A Compact Marshalling Scheme for Acoustic Communications," in *OCEANS 2010 IEEE*, Sydney, 2010. pp. 1-10.

[89] M. Porter, "Bellhop Code," 2014. [Online]. Available: http://oalib.hlsresearch.com/Rays/index.html. [Accessed September 2014].

[90] "Iver2 AUV & Spares," OceanServer, [Online]. Available: http://www.oceanserver-store.com/iver2auv.html. [Accessed 15 November 2014].

[91] M. Seto, Bedford Basin Composition, private communication: email, 2014.

[92] O. Rodriguez, "General description of the BELLHOP ray tracing program," Faro, 2008.

[93] M. Porter, "The BELLHOP Manual and User's Guide," La Jolla, 2011.

[94] BTech Acoustics, LLC, "Model BT-1RCL," Barrington, 2010.

[95] K. Pahlavan and A. Levesque, Wireless Information Networks, Hoboken: Wiley, 2005.

[96] R. Stokey, L. Freitag and M. Grund, "A Compact Control Language for AUV Acoustic Communication," in *OCEANS 2005*, 2005. pp. 1133-1137.

[97] "Micro-Modem Overview," Woods Hole Oceanographic Institution, 2006. [Online]. Available: http://acomms.whoi.edu/umodem/. [Accessed 2 November 2014].

[98] E. Alpaydin, Introduction to Machine Learning, Cambridge: MIT Press, 2010.

APPENDIX A EXAMPLE BELLHOP INPUT FILES

filename1382016839-1.env: Environmental description file.

```
'Auto-generated ENV File for BELLHOP'          !TITLE
25000.0          !FREQUENCY
1            ![UNUSED]
'CVWT'              ! [(SSP INTERP METHOD)(TOP LAYER)(BOTTOM ATTENUATION
UNITS)(VOLUME ATTENUATION)]
0  0  60.0,        ![UNUSED] [UNUSED] [MAX DEPTH]
0.0  1496.8  /! [DEPTH] [SOUND SPEED]
1.0  1495.9  /! [DEPTH] [SOUND SPEED]
2.0  1495.9  /! [DEPTH] [SOUND SPEED]
3.0  1495.8  /! [DEPTH] [SOUND SPEED]
4.0  1495.8  /! [DEPTH] [SOUND SPEED]
5.0  1495.8  /! [DEPTH] [SOUND SPEED]
6.0  1495.8  /! [DEPTH] [SOUND SPEED]
7.0  1495.8  /! [DEPTH] [SOUND SPEED]
8.0  1495.8  /! [DEPTH] [SOUND SPEED]
9.0  1495.9  /! [DEPTH] [SOUND SPEED]
10.0  1495.9  /! [DEPTH] [SOUND SPEED]
11.0  1495.9  /! [DEPTH] [SOUND SPEED]
12.0  1495.9  /! [DEPTH] [SOUND SPEED]
13.0  1495.9  /! [DEPTH] [SOUND SPEED]
14.0  1495.9  /! [DEPTH] [SOUND SPEED]
15.0  1495.9  /! [DEPTH] [SOUND SPEED]
16.0  1496.0  /! [DEPTH] [SOUND SPEED]
16.9  1496.0  /! [DEPTH] [SOUND SPEED]
17.9  1496.0  /! [DEPTH] [SOUND SPEED]
19.0  1495.9  /! [DEPTH] [SOUND SPEED]
20.0  1488.1  /! [DEPTH] [SOUND SPEED]
21.0  1480.8  /! [DEPTH] [SOUND SPEED]
21.9  1474.1  /! [DEPTH] [SOUND SPEED]
22.9  1470.2  /! [DEPTH] [SOUND SPEED]
23.9  1469.5  /! [DEPTH] [SOUND SPEED]
24.9  1468.8  /! [DEPTH] [SOUND SPEED]
25.9  1467.8  /! [DEPTH] [SOUND SPEED]
26.9  1467.3  /! [DEPTH] [SOUND SPEED]
27.9  1466.9  /! [DEPTH] [SOUND SPEED]
28.9  1466.9  /! [DEPTH] [SOUND SPEED]
29.9  1466.9  /! [DEPTH] [SOUND SPEED]
30.9  1466.9  /! [DEPTH] [SOUND SPEED]
31.9  1466.9  /! [DEPTH] [SOUND SPEED]
32.9  1467.0  /! [DEPTH] [SOUND SPEED]
33.9  1467.0  /! [DEPTH] [SOUND SPEED]
34.9  1467.1  /! [DEPTH] [SOUND SPEED]
35.9  1467.2  /! [DEPTH] [SOUND SPEED]
36.9  1467.2  /! [DEPTH] [SOUND SPEED]
37.9  1467.2  /! [DEPTH] [SOUND SPEED]
38.9  1467.2  /! [DEPTH] [SOUND SPEED]
39.9  1467.3  /! [DEPTH] [SOUND SPEED]
```

```
40.9  1467.3  /! [DEPTH] [SOUND SPEED]
50.0  1467.3  /! [DEPTH] [SOUND SPEED]
60.0  1467.3  /! [DEPTH] [SOUND SPEED]
'A*'     0.0  !  [(BOTTOM  LAYER)(EXTERNAL  BATHYMETRY]  [SIGMA  BOTTOM
ROUGHNESS]
60.0  1490.0 0 1.6 1.0 /     ![MAX DEPTH] [SOUND SPEED BOTTOM] [UNUSED]
[DENSITY BOTTOM] [ATTENUATION BOTTOM]
1                 ! [NUMBER SOURCES]
0.5 /             ! [SOURCE DEPTHS 1:N (m)]
1                 ! [NUMBER RECEIVER DEPTHS]
10.0 /    ! [RECEIVER DEPTHS 1:N (m)]
1                 ! [NUMBER RECEIVER RANGES]
0.0992 /    ! [RECEIVER RANGES 1:N (km)]
'AB*'                    !  [(RUN  TYPE)(BEAM  TYPE)(EXTERNAL  SOURCE  BEAM
PATTERN)]
0                 ! [NUMBER OF BEAMS]
-89.0 89.0 /        ! [MIN ANGLE] [MAX ANGLE]
0.0 60.6 0.1002  ! [STEP SIZE] [DEPTH BOX] [RANGE BOX]
```

filename1382016839-1.bty: Bathymetric bottom profile description file

```
10
0.0000  22.2
0.0110  24.0
0.0220  26.0
0.0331  30.1
0.0441  32.8
0.0551  34.4
0.0661  36.8
0.0772  38.3
0.0882  39.7
0.0992  40.8
```

filename1382016839-1.sbp: Source beam pattern description file

```
19
-135.0  -6.90
-120.0  -9.70
-105.0  -5.50
-90.0  -3.70
-75.0  -5.50
-60.0  -9.70
-45.0  -6.90
-30.0  -3.80
-15.0  -2.80
0.0  -2.90
15.0  -3.70
30.0  -5.30
45.0  -6.80
60.0  -4.50
75.0  -1.20
90.0  0.00
105.0  -1.20
120.0  -4.50
135.0  -6.80
```

APPENDIX B EXAMPLE MATLAB SIMULATION CODE

vehicle.m: Class defining a simulated AUV

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% vehicle class                                                            %
%                                                                          %
% (c) Dainis Nams 2013/2014 all rights reserved                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

classdef vehicle < handle

    % ******The following properties can be set only by class methods
*************************************
    properties (GetAccess = private)
    end

    % ****** The following properties are hidden but can be called
****************************************
    properties (Hidden)
    end

    % ****** The following properties remain constant
*****************************************************
    properties (Constant)

        % vehicle kinematics variables. Customized to DRDC IVER2 vehicles.
        maxLinAccel         = 0.25; % IVER2 max linear acceleration
(m/s^2)
        maxPitchRate        = 10.0; % IVER2 rate of delta pitch
(deg/s^2)
        maxYawAccel         = 12.9; % IVER2 rate of delta yaw
(deg/s^2)
        maxYawRate          = 13.0; % IVER2 maximum yaw rate; related to min
turn radius and speed  (deg/s)
        kinematicTimestep   = 0.1;  % Time by which kinematic and PID equations
are incremented     (s)

        % vehicle PID controller gains
        KpSpeed = 1;    % Proportional constant for speed control
        KdSpeed = 0;    % Derivative constant for speed control
        KpYaw   = 1;    % Proportional constant for yaw control
        KdYaw   = 0;    % Derivative constant for yaw control
        KpPitch = 5;    % Proportional constant for pitch control
        KdPitch = 15;   % Derivative constant for pitch control

        % plot output controllers
        lengthHistory = 10; % how many old parameter values to plot
    end

    % ****** The following properties are visible and accessible by
all************************************
    properties

        % vehicle state variables
        roll = 0;           % current roll; this won't change in scope of
simulation        (deg)
```

```
        pitch = 0;          % current pitch; + up, - down
(deg)
        heading = 0;        % current heading
(deg)
        speed = 0;          % current water speed (not same as groundspeed)
(m/s)
        groundSpeed = 0;    % current speed over ground (used for calculating x
and y)      (m/s)
        verticalSpeed = 0;  % current speed up/down through the water column
(m/s)
        linAccel = 0;       % current water acceleration
(m/s^2)
        dPitch = 0;         % current rate of pitch change
(deg/s)
        dYaw = 0;           % current rate of heading change
(deg/s)
        aYaw = 0;           % current yaw acceleration
(deg/s^2)
        x;                  % current x coordinate in geodesy frame
(m)
        y;                  % current y coordinate in geodesy frame
(m)
        lat;                % current latitude
(dd.dd)
        lon;                % current longitude
(dd.dd)
        depth = 0           % depth from surface
(m)
        rateOfDive = 0;     % vertical speed in water column
(m/s)
        altitude = -1;      % altitude from seafloor
(m)
        t =0;               % time from start of mission
(s)

        %mission object
        mission;

        % vehicle hardware parameters
        sourceBeamPattern = struct; % contents of *.sbp file with dB loss vs
angle of vehicle's transducer
        vehicleType;                % whether the vehicle is an AUV or a BARGE
determines if it updates position

        % mission configuration variables
        maxPitch;               % maximum pitch allowed; set by mission
(deg)
        waypointTolerance;      % required proximity to complete waypoint ; set
by mission      (m)
        missionComplete = 0;    % maintains a record as to whether or not the
mission is compelte

        % next waypoint properties (used for navigation)
        nextWPLat;              % latitude of next waypoint
(dd.dd)
        nextWPlon;              % longitude of next waypoint
(dd.dd)
        nextWPX;                % x position of next waypoint in geodesy frame
(m)
```

```
        nextWPY;                % y position of next waypoint in geodesy frame
(m)
        nextWPDepth;            % depth as specified by next waypoint; -1 if
altitude instead   (m)
        nextWPAltitude;         % altitude as specified by next waypoint; -1 if
depth instead   (m)
        nextWPSpeed;            % speed craft should use to travel to next
waypoint             (m/s)
        nextWPNumber;           % number of next waypoint

        % controller setpoints
        setpointSpeed   = 0;    % this overrides nextWPSpeed if craft has to
turn tightly       (m/s)
        setpointHeading = 0;    % calculated from current heading, position, and
next waypoint  (deg)
        setpointPitch   = 0;    % this is calculated based on required depth /
altitude         (deg)
        setpointDepth   = 0;    % adjusted to cover either altitude or depth
following          (m)

        % time history arrays for output to plotting tools
        historyX                = [];
        historyY                = [];
        historySpeed            = [];
        historyDepth            = [];
        historyPitch            = [];
        historySeabed           = [];
        historySetpointSpeed    = [];
        historySetpointDepth    = [];
        historySetpointPitch    = [];

    end

   % Define events
    events
    end

   %Define class methods
    methods

        % ****** Initilizer. Reads in mission file and populates vehicle object
***************************
        function auv = vehicle(mission, vehicleType, xStart, yStart)
            auv.vehicleType = vehicleType;
            switch vehicleType
                case 'AUV'
                    % get mission configuration variables from mission
                    auv.mission = mission;
                    auv.maxPitch = auv.mission.getMaxPitchAngle();
                    auv.waypointTolerance = auv.mission.getWaypointTolerance();

                    % set AUV position
                    auv.x = xStart;
                    auv.y = yStart;

                    % initialize current waypoint variables
                    [auv.nextWPX auv.nextWPY auv.nextWPDepth auv.nextWPAltitude
auv.nextWPSpeed auv.nextWPNumber] = auv.mission.getNextWaypoint();

                case 'BARGE'
```

```
                    % set parameters for static barge transducer
                    auv.x = 0;
                    auv.y = 0;
                    auv.depth = 10;
                    %display(auv.mission.missionComplete)
                    auv.mission.missionComplete = true;
                    display('auv mission')
                    display(auv.mission.missionComplete)
            end
            auv.readSourceBeamPattern;
        end

        % ****** Functions to access vehicle information
****************************************************

        % access single vehicle state parameters
************************************************************
        function [x] = getX(auv)
        % Returns vehicle x
            x = auv.x;
        end

        function [y] = getY(auv)
        % Returns vehicle y
            y = auv.y;
        end

        function [h] = getHeading(auv)
        % Returns vehicle heading
            h = auv.heading;
        end

        function [s] = getSpeed(auv)
        % Returns vehicle x
            s = auv.speed;
        end

        function [p] = getPitch(auv)
        % Returns vehicle pitch
            p = auv.pitch;
        end

        function [d] = getDepth(auv)
        % Returns vehicle depth
            d = auv.depth;
        end

        function [a] = getAltitude(auv)
        % Returns vehicle altitude
            a = auv.altitude;
        end

        function [s] = getSeabed(auv)
        % Returns vehicle altitude
            s = auv.altitude + auv.depth;
        end

        % access single vehicle control parameters
        function [sp] = getSetpointHeading(auv)
        % Returns vehicle heading
```

```
            sp = auv.setpointHeading;
        end

        function [sp] = getSetpointSpeed(auv)
        % Returns vehicle speed setpoint
            sp = auv.setpointSpeed;
        end

        function [sp] = getSetpointDepth(auv)
        % Returns vehicle vehicle depth setpoint
            sp = auv.setpointDepth;
        end

        function [sp] = getSetpointPitch(auv)
        % Returns vehicle vehicle depth setpoint
            sp = auv.setpointPitch;
        end

        % access history of vehicle state parameters
        function [h] = getHistoryX(auv)
        % Returns history of vehicle x
            auv.historyX = auv.updateHistory(auv.historyX, auv.x);
            h = auv.historyX;
        end

        function [h] = getHistoryY(auv)
        % Returns history of vehicle y
            auv.historyY = auv.updateHistory(auv.historyY, auv.y);
            h = auv.historyY;
        end

        function [h] = getHistorySpeed(auv)
        % Returns history of vehicle speed
            auv.historySpeed = auv.updateHistory(auv.historySpeed, auv.speed);
            h = auv.historySpeed;
        end

        function [h] = getHistoryDepth(auv)
        % Returns history of vehicle depth
            auv.historyDepth = auv.updateHistory(auv.historyDepth, auv.depth);
            h = auv.historyDepth;
        end

        function [h] = getHistoryPitch(auv)
        % Returns history of vehicle pitch
            auv.historyPitch = auv.updateHistory(auv.historyPitch, auv.pitch);
            h = auv.historyPitch;
        end

        function [h] = getHistorySeabed(auv)
        % Returns history of vehicle seabed height
            auv.historySeabed = auv.updateHistory(auv.historySeabed,
auv.getSeabed);
            h = auv.historySeabed;
        end

        function [h] = getHistorySetpointSpeed(auv)
        % Returns history of vehicle speed setpoint
            auv.historySetpointSpeed =
auv.updateHistory(auv.historySetpointSpeed, auv.setpointSpeed);
```

```
            h = auv.historySetpointSpeed;
        end

        function [h] = getHistorySetpointDepth(auv)
        % Returns history of vehicle depth setpoint
            auv.historySetpointDepth =
auv.updateHistory(auv.historySetpointDepth, auv.setpointDepth);
            h = auv.historySetpointDepth;
        end

        function [h] = getHistorySetpointPitch(auv)
        % Returns history of vehicle pitch setpoint
            auv.historySetpointPitch =
auv.updateHistory(auv.historySetpointPitch, auv.setpointPitch);
            h = auv.historySetpointPitch;
        end

        % returns a source beam pattern struct rotated according to current
vehicle pitch
        function [sbp] = getPitchedSourceBeamPattern(auv)
            % initializers
            sbp = struct;
            angle   = [];

            % twist beam pattern
            for i = 1:auv.sourceBeamPattern.num
                angle = [angle auv.sourceBeamPattern.angle(i)-auv.pitch ];
            end

            % output variables
            sbp.num = auv.sourceBeamPattern.num;
            sbp.angle = angle;
            sbp.dBLoss = auv.sourceBeamPattern.dBLoss;
        end

        % ****** Functions to update vehicle information
***************************************************

        function [] = updatePosition(auv,deltaT, env)
        % Updates vehicle state info. Input is timestep and global environment
(for sensors)

            if ~strcmp(auv.vehicleType,'BARGE')

                % Update position by auv.kinematicTimestep seconds until
"deltaT" increment is reached
                for i = 1:deltaT/auv.kinematicTimestep

                    % only move AUV if it has not completed mission (park on
completion)
                    if ~auv.mission.isMissionComplete()

                        % Altitude sensor
                        auv.altitude = env.getMissionDepth(auv.x,auv.y) -
auv.depth;

                        % Heading control --------------------------------------
--------------------------------
                        % calculate heading setpoint
                        deltaX = auv.nextWPX - auv.x;
                        deltaY = auv.nextWPY - auv.y;
                        auv.setpointHeading = mod(-
(rad2deg(atan2(deltaY,deltaX)) - 90),360);

                        % get error in heading current vs setpoint
                        errorAngle = auv.heading - auv.setpointHeading;
                        errorAngle = mod((errorAngle + 180), 360) - 180;

                        auv.dYaw = -errorAngle * auv.KpYaw + auv.KdYaw *
auv.dYaw;

                        if abs(auv.dYaw) > auv.maxYawRate
                            auv.dYaw = auv.maxYawRate * sign(auv.dYaw);
                        end

                        % Pitch control ----------------------------------------
--------------------------------

                        if auv.nextWPDepth ~= -1  % Depth target
                            auv.setpointDepth = auv.nextWPDepth;
                        elseif auv.nextWPAltitude ~= -1 % Altitude target
                            auv.setpointDepth = env.getMissionDepth(auv.x,auv.y)
- auv.nextWPAltitude;
                        end
                        errorDepth = auv.depth - auv.setpointDepth;

                        auv.dPitch = errorDepth * auv.KpPitch + auv.KdPitch *
auv.verticalSpeed;
                        if abs(auv.dPitch) > auv.maxPitchRate
                            auv.dPitch = auv.maxPitchRate * sign(auv.dPitch);
                        end

                        % Speed control ----------------------------------------
--------------------------------
                        auv.setpointSpeed = auv.nextWPSpeed * (1 -
0.2*abs(auv.dYaw)/auv.maxYawRate);
                        errorSpeed = auv.setpointSpeed - auv.speed;
                        auv.linAccel = errorSpeed * auv.KpSpeed + auv.linAccel *
auv.KdSpeed;
                        if abs(auv.linAccel) > auv.maxLinAccel
                            auv.linAccel = auv.maxLinAccel * sign(auv.linAccel);
                        end

                        % Update position --------------------------------------
--------------------------------
                        auv.speed       = auv.speed + auv.linAccel *
auv.kinematicTimestep;

                        auv.pitch = auv.pitch + auv.dPitch *
auv.kinematicTimestep;
                        if abs(auv.pitch) > auv.maxPitch
                            auv.pitch = auv.maxPitch * sign(auv.pitch);
                        end

                        auv.groundSpeed = cosd(auv.pitch) * auv.speed;

                        auv.verticalSpeed = -sind(auv.pitch) * auv.speed;

                        distanceTravelled = auv.groundSpeed *
auv.kinematicTimestep;
```

```
                        auv.depth = auv.depth + auv.verticalSpeed *
auv.kinematicTimestep;
                        % in case bounces "above surface" in PID control of
surfacing
                        if auv.depth < 0
                            auv.depth = 0;
                        end

                        auv.heading = auv.heading + auv.dYaw *
auv.kinematicTimestep;
                        if auv.heading > 360
                            auv.heading = auv.heading - 360;
                        elseif auv.heading < 0
                            auv.heading = auv.heading + 360;
                        end

                        % lookup table for custom ATAN2 heading -> x & y
                        if auv.heading < 90
                            auv.x = auv.x + sind(auv.heading) *
distanceTravelled;
                            auv.y = auv.y + cosd(auv.heading) *
distanceTravelled;

                        elseif auv.heading < 180
                            auv.x = auv.x + cosd(auv.heading-90) *
distanceTravelled;
                            auv.y = auv.y - sind(auv.heading-90) *
distanceTravelled;
                        elseif auv.heading < 270
                            auv.x = auv.x - sind(auv.heading-180) *
distanceTravelled;
                            auv.y = auv.y - cosd(auv.heading-180) *
distanceTravelled;
                        else
                            auv.x = auv.x - cosd(auv.heading-270) *
distanceTravelled;
                            auv.y = auv.y + sind(auv.heading-270) *
distanceTravelled;
                        end

                        %check whether waypoint is reached
                        auv.isWaypointCompleted();

                    end % if mission not completed yet

                    % advance time by one step
                    auv.t = auv.t + auv.kinematicTimestep;
                end % for each kinematic timestep
            end % if vehicleType ~= BARGE
        end % function

        % ****** Functions internal to vehicle
************************************************************

        function [] = isWaypointCompleted(auv)
        % Checks whether or not the vehicle is close enough to the current
waypoint
```

```
            distanceToWaypoint = sqrt((auv.x - auv.nextWPX)^2 + (auv.y -
auv.nextWPY)^2);
            if distanceToWaypoint < auv.waypointTolerance

                % let mission know that the waypoint has been completed
                auv.mission.waypointCompleted();

                % have we completed the mission?
                if auv.mission.isMissionComplete()
                    auv.missionComplete = 1;
                else
                    % get next waypoint
                     [auv.nextWPX auv.nextWPY auv.nextWPDepth auv.nextWPAltitude
auv.nextWPSpeed auv.nextWPNumber] = auv.mission.getNextWaypoint();
                end
            end
        end

        function [h] = updateHistory(auv, history, newValue)
        % returns an updated scrolling history list
            if length(history) < auv.lengthHistory;
                h = [history newValue];
            else
                h = [history(2:auv.lengthHistory) newValue];
            end
        end

        % reads source beam pattern file from \sourceBeamPatterns
        function readSourceBeamPattern(auv)

            % static filename because we only use the one transducer
            filename = 'BT1.sbp';
            fid = fopen([ fileparts(mfilename('fullpath'))
'\sourceBeamPatterns\' filename ]);

            % get first line which is number of entries
            line = fgetl(fid);
            auv.sourceBeamPattern.num = str2double(line);

            % loop through remaining lines and read [angle (deg)] [dB loss]
values
            line    = fgetl(fid);
            angle   = [];
            dB      = [];
            while length(line) > 1
                lineDelimited = getSubstring(line, ' ');
                angle = [angle str2double(lineDelimited(1)) ];
                dB    = [dB str2double(lineDelimited(2)) ];
                line = fgetl(fid);
            end
            fclose(fid);
            auv.sourceBeamPattern.angle = angle;
            auv.sourceBeamPattern.dBLoss = dB;
        end
    end % methods
end % classdef
```

APPENDIX C EXAMPLE MOOS AUTONOMY CODE

pChangeDataRate.cpp: MOOS program to adjust WHOI Micro-Modem data rate

```
/******************************************************************
 *  Author: Dainis Nams
 *
 ******************************************************************/

using namespace std;
#include <MOOSLIB/MOOSLib.h>
#include "pChangeDataRate.h"
#include <sys/time.h>
#include <ctime>
#include "/home/namzy/moos-ivp/ivp/src/lib_mbutil/MBUtils.h"
#include "goby/moos/moos_protobuf_helpers.h"

using goby::acomms::operator<<;

pChangeDataRate::pChangeDataRate(){

}

pChangeDataRate::~pChangeDataRate(){

}

bool pChangeDataRate::Iterate(){
  getTimeIntoCycle();
  m_Comms.Notify("SECONDS_TO_NEXT_CYCLE",secsToNext,MOOSTime());

 return true;
}

bool pChangeDataRate::OnConnectToServer(){

  registerVariables();

return true;
}

bool pChangeDataRate::OnStartUp(){

  getTimeIntoCycle();

    //PULL FROM MISSION FILE
    if (!m_MissionReader.GetConfigurationParam("slot_length",
slot_length))
        slot_length = 15;
    cout<<slot_length<<endl;
    if (!m_MissionReader.GetConfigurationParam("vehicle_id",
vehicle_id))
        vehicle_id = 1;

    if (vehicle_id == 1) partner_id = 2;
    if (vehicle_id == 2) partner_id = 1;

    registerVariables();
    queue_size = .8;

return true;
}

bool pChangeDataRate::OnNewMail(MOOSMSG_LIST &NewMail){


   MOOSMSG_LIST::iterator p;
    for(p = NewMail.begin();p!=NewMail.end();p++)
    {
        CMOOSMsg &  rMsgIn = *p;

//-------subscribe to requested data rates and reset queue with new
transmit data rates
      if(rMsgIn.GetKey() == "DATA_RATE")
      {
          dataRate = rMsgIn.GetDouble();
          assignMACSlot(dataRate, slot_length);
      }
    }
   return true;
}

// Register (subscribe) MOOSDB variables
void pChangeDataRate::registerVariables()
{
    m_Comms.Register("DATA_RATE",0);
}

// Insert a new AMAC queue slot at the front of the queue
void pChangeDataRate::assignMACSlot(int rate, int length)
{
    //--publish message to replace current MAC slot
    goby::acomms::protobuf::MACUpdate mac_update_msg;

    mac_update_msg.set_dest(vehicle_id);

mac_update_msg.set_update_type(goby::acomms::protobuf::MACUpdate::ASSIGN
);

    //--define the new MAC slot according to the ModemTransmission proto
file
    goby::acomms::protobuf::ModemTransmission* first_slot =
mac_update_msg.add_slot();
    first_slot->set_src(1);
    first_slot->set_dest(0);
    first_slot->set_rate(rate);
    first_slot-
>set_type(goby::acomms::protobuf::ModemTransmission::DATA);
    first_slot->set_ack_requested(false);
    first_slot->set_slot_seconds(length);

    goby::acomms::protobuf::ModemTransmission* second_slot =
mac_update_msg.add_slot();
    second_slot->set_src(2);
    second_slot->set_dest(0);
    second_slot->set_rate(rate);
    second_slot-
>set_type(goby::acomms::protobuf::ModemTransmission::DATA);
    second_slot->set_ack_requested(false);
    second_slot->set_slot_seconds(length);

    std::string serialized_update;
    serialize_for_moos(&serialized_update, mac_update_msg);
    m_Comms.Notify("ACOMMS_MAC_CYCLE_UPDATE",
serialized_update,MOOSTime());

}
// return the number of seconds into current cycle
double pChangeDataRate::getTimeIntoCycle()
{
   // current date/time based on current system
   time_t now = time(0); // # of seconds since January 1,1970
   tm *ltm = localtime(&now);

    int cycle_duration = slot_length*2;
    double secondsSinceDayStart = (3 + ltm->tm_hour) * 3600 + (ltm-
>tm_min) * 60 + ltm->tm_sec;
    if (ltm->tm_hour >= 21) // case: after 12AM, subtract 24 hours
      secondsSinceDayStart = (ltm->tm_hour - 21) * 3600 + (ltm->tm_min)
* 60 + ltm->tm_sec;
    int cycles_since_day_start_ =
(floor(secondsSinceDayStart/cycle_duration) + 1);
    double secsToNextSinceDayStart =
cycles_since_day_start_*cycle_duration;
    secsToNext = secsToNextSinceDayStart - secondsSinceDayStart;

    cout << "Time: "<< 3 + ltm->tm_hour << ":" << ltm->tm_min << ":" <<
ltm->tm_sec <<" SECONDS TO NEXT CYCLE: "<<secsToNext<<endl;
}
```