Vehicle autonomy under the Arctic ice : environmental adaptation through model-aided machine learning

by

Oscar Alberto Viquez R.

S.B. 2013, Mechanical Engineering, Massachusetts Institute of Technology
S.M. 2017, Mechanical Engineering, Massachusetts Institute of Technology

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Science

at the

Massachusetts Institute of Technology

May 2022

Author . . . . . .
Department of Mechanical Engineering, MIT
April 11th, 2022

Certified by . . . . . .
Henrik Schmidt
Professor of Mechanical and Ocean Engineering, MIT
Thesis Supervisor

Accepted by . . . . . .
Nicolas Hadjiconstantinou
Professor of Mechanical Engineering, MIT
Chair, Department Committee on Graduate Students

Vehicle autonomy under the Arctic ice : environmental adaptation through model-aided machine learning

by

Oscar Alberto Viquez R.

Submitted to the Department of Mechanical Engineering on April 11th, 2022,
in partial fulfillment of the requirements for the degree of Doctor of Science

Abstract

The use of autonomous vehicles has been growing across the globe, driven by their ability to meet the diverse needs of industry and scientific applications. Terrestrial and aerial uncrewed vehicles typically benefit from high-throughput communication systems which enable accurate positioning and operator input; Autonomous Underwater Vehicles (AUVs), however, generally require a higher degree of autonomy as they must rely on much more limited communication links and lack access to global navigation satellite systems (GNSS) while underway. This distinction becomes especially important in hazardous environments like the Arctic Ocean, where surface ice may impede an AUV from breaching to regain access to position and controller updates. Instead, underwater vehicles in ice-covered environments require a higher level of autonomous decision-making, and rely on a combination of self-contained sensors and acoustic positioning networks for navigation – but the latter generally rely on a deterministic conversion of acoustic travel times to ranges, failing to capture the natural variability of the acoustic environment.

This dissertation demonstrates the application of physics-based machine learning techniques as an alternative to deterministic solutions for environmental adaptation in unmanned vehicle autonomy. This is achieved by gradually incrementing the complexity of the adaptation problem: first, the tasks of behavior identification and riverbed characterization are tackled with a classification approach; next, an embedded acoustics model is used in place of the conventional linear model for acoustic positioning, and a feature design approach is employed to improve the performance of this embedded range estimation; last, a pseudo-tomographic approach based on neural network techniques is proposed as a complement to compressive sensing, to enable exploratory environmental adaptation onboard AUVs.

The improvements to acoustic positioning are validated against data collected in the Beaufort Sea in March of 2020, where the presence of the Beaufort Lens combines with the surface ice covering the Arctic Ocean to create an ideal setting in which to demonstrate the importance of environmental adaptation. These capabilities may impact monitoring efforts in the area, which has seen increased interest from fishing, trade and military operations, and is of significant importance to understanding climate conditions.

Thesis Supervisor: Henrik Schmidt
Title: Professor of Mechanical and Ocean Engineering

Acknowledgments

"As we express our gratitude, we must never forget that the highest appreciation is not to utter words, but to live by them."

– John F. Kennedy, *Proclamation 3560*

To begin writing this portion of my dissertation is to embark on a Herculean task, for I have been fortunate to receive the support of many kind souls throughout my time at MIT. There are many more names worthy of mention than I can capture in a few pages, but I'll try my best.

This extensive list begins with my advisor, Henrik. In welcoming me to the lab, he granted me the adventure of a lifetime. I was lucky to join him in various deployments in open waters, and to travel with him to the Arctic for the experiment at the heart of this dissertation – not to mention the joy of all our small-scale deployments in the Charles river! I am grateful to count him as a mentor; and I am glad I got to be part of the community he built in the lab. Under Henrik's guidance, I got to work with some of the brightest, most supportive collaborators and friends. I am indebted to Eesh, Rui, Bradli, Dan, Kristen, Supun, Nick, Misha, Paul, and so many more, for shaping not just my time in graduate school, but my entire outlook for future collaborations – you all set the highest of bars.

I am, of course, extremely grateful to my committee members. Erin, whose curiosity and passion – and just enough pragmatism – were always a fountain of inspiration for continuous growth and discovery; I must add that her support in field experiments cannot go unstated. And Peko, whose refreshing perspective challenged me to distill my work into its most fundamental form.

I'd also like to extend my gratitude to Mike, who for the past few years trusted me to join him in training the new generation of leaders in marine autonomy. He nurtured my reserved inclination for mentorship into a true passion. I can only hope I may continue to serve our community as he does every day.

I must also thank Geoff, who has kept our lab running smoothly over the years. Likewise, I extend my gratitude to Leslie, Una, Saana, Janice, and the rest of the department's staff of the past few years. The world-within-a-world that is MechE would not function without them.

I am also grateful to the many friends I have made along this journey. Whether it was camping and climbing in the White Mountains or in Acadia; building something new at one of the machine shops on campus; catching up during coffee hour... You are too many to mention by name; but you are also the reason I've made it through this process with (most of) my sanity.

I am extremely grateful to my fiancée, Nicole. Her company and her support over the past couple of years made writing this thesis during COVID a much more bearable endeavor. I am also grateful to my brother, Jose, and my sister, Kbi, who have been there all along, poking fun when things were good and lending a shoulder when the going got tough.

Lastly, this degree belongs to my parents as much as it does to me. From very early on, they nurtured a sense of curiosity that extended well beyond their own comfort zones. Without their faithful support and their encouragement to take things apart and ask too many questions, I wouldn't be where I am today.

Funding Sources

This work was supported by:

Office of Naval Research
Award Number: N00014-19-1-2741
Environmentally Adaptive Autonomy for Under-Ice Acoustic Navigation and Communication

Office of Naval Research
Award Number: N00014-19-1-2716
TFO: Realism and Uncertainties in Navy Decision Aids

Office of Naval Research
Award Number: N00014-17-1-2474
Environmentally Adaptive Acoustic Communication and Navigation in the new Arctic

Defense Advanced Research Projects Agency / US Army Contracting Command - New Jersey
Award Number: W15QKN-15-1-0001
Environmentally Adaptive Off-Board Acoustic Sensing Concept for the Rapidly Changing Arctic Ocean

Office of Naval Research / University of California-San Diego
Award Number: N00014-16-1-2129 > N00014-13-1-0632 / PO #S9000381, Sub #43019208
MURI: The Information Content of Ocean Noise: Theory and Experiment - Imaging the Changing Arctic with Ice Noise

US Air Force / Lincoln Laboratory
Award Number: FA8721-05-C-0002 / PO 7000339130
Biomimetic Adaptive Forward-Looking Sonar for Object Recognition

Defense Advanced Research Projects Agency / Applied Physical Sciences Corp.
Award Number: HR0011-18-C-0008 / APS-18-03
Tactical Exploitation of the Acoustic Channel (TEAC)

Contents

Contents 9

List of Figures 15

List of Tables 31

1 Introduction 35

1.1 A new and changing Arctic . . . 36

1.2 The need for real-time environmental adaptation . . . 39

1.3 Problem statement and outline . . . 43

2 Background 45

2.1 Estimating the speed of sound in water . . . 46

2.2 Acoustic propagation . . . 53

2.2.1 Ray methods . . . 54

2.2.2 The Eikonal equation . . . 56

2.2.3 Multipath propagation . . . 58

2.3 Acoustic tomography and geophysical inversion . . . 61

2.3.1 The linearized inverse problem . . . 64

2.4 Positioning . . . 66
2.4.1 The Global Navigation Satellite System (GNSS) . . . 69
2.4.2 Acoustic positioning . . . 71
2.5 Autonomy . . . 72
2.5.1 Middleware autonomy . . . 72
2.5.2 Communications . . . 75
2.6 Summary . . . 76

3 A Machine Learning primer 77
3.1 A brief history of Machine Learning . . . 78
3.2 Behavior classification : understanding your signals . . . 80
3.2.1 First stage : baseline observations . . . 81
3.2.2 Second stage : reflecting on the challenges . . . 89
3.2.3 Third stage : recurring themes as signals . . . 91
3.3 Environment characterization : exploiting *a priori* information . . . 96
3.3.1 Assembling candidate models . . . 97
3.3.2 Virtual sampling as a stochastic observer . . . 101
3.3.3 Collecting field measurements . . . 103
3.3.4 Statistical performance . . . 104
3.4 Summary . . . 108

4 ICEX-20 : Experimental system 111
4.1 Core systems . . . 112
4.1.1 The Virtual Ocean Autonomy Testbed . . . 112
4.1.2 NETSIM : Hardware-In-The-Loop simulation . . . 116
4.1.3 ICNN : The Integrated Communications and Navigation Network . . . 119

4.1.4 AUV *Macrura* : Autonomy Payload 120
4.2 Observable information from the vehicle perspective 123
4.2.1 The Sound Speed Domain 124
4.2.2 The Acoustic Domain 126
4.3 Simulation work 127
4.4 Summary 134

5 ICEX-20 FIELD REPORT AND DATA REVIEW 135
5.1 Experiment Timeline 136
5.1.1 Getting to camp – and coming back 136
5.1.2 Datasets acquired 138
5.2 Overview of the acoustic communications logs 140
5.2.1 Modem Error Codes 140
5.2.2 Impulse Response Estimates 142
5.3 Coordinated operations and the TDMA scheme 144
5.3.1 Time synchronization 145
5.3.2 Effects of clock synchronization error: acoustic transmissions timeline . 146
5.3.3 (More) Effects of clock synchronization error: differences between transmission and reception events 148
5.3.4 Time skew in the ICEX-20 data 151
5.4 A closer look at the acoustic comms data 153
5.4.1 Transmission events 154
5.4.2 Reception events 156
5.5 ICNN tracking data 164
5.6 Summary 167

6 Exploiting the multi-path structure 169
6.1 Minimum bounce criteria . . . 171
6.1.1 Modem-to-modem connections . . . 173
6.1.2 On recorded data vs bottom-bounce predictions . . . 176
6.2 Nearest bounce criteria . . . 177
6.2.1 Validating the nearest bounce criteria . . . 179
6.2.2 Performance of the nearest bounce and minimum bounce criteria . . . 180
6.2.3 A case study of the performance improvement . . . 182
6.2.4 Comparison with an eigenray-search approach . . . 186
6.3 Effect of the NBC on the navigation paradigm . . . 186
6.3.1 Baseline performance : *in-situ* trilateration . . . 190
6.3.2 Comparing the different ranging approaches . . . 194
6.4 AUV navigation performance . . . 198
6.4.1 General data features . . . 198
6.4.2 Position corrections posted by the ICNN . . . 201
6.5 Summary . . . 204

7 Machine learning, revisited 207
7.1 ML in environmental and ocean sciences . . . 208
7.1.1 The Landslide Hazard Assessment for Situational Awareness . . . 209
7.1.2 Learning data relations in Ocean Acoustics . . . 210
7.2 Physically Informed Neural Networks . . . 212
7.2.1 A brief discussion on types of learning . . . 212
7.2.2 Conventional neural network architecture . . . 214
7.2.3 How PINNs compare to other NNs . . . 215
7.2.4 Solving the Factored Eikonal equation with PINNs . . . 217

7.3 Augmenting the model-aided environment estimation framework 219
7.3.1 The baseline EOF solution 219
7.3.2 Accounting for acoustic time-of-travel data 221
7.4 Exploring the potential of PINNs for field use 223
7.4.1 PINN architecture 224
7.4.2 A PINN case study in 3 parts 225
7.5 Environment estimation as a data assimilation problem 231
7.5.1 A simplified inversion model 232
7.5.2 Travel time sample selection 233
7.5.3 Learning the travel-time observation matrix 235
7.6 Summary 238

8 Conclusions 239
8.1 Summary of contributions 239
8.2 Opportunities for improvement and future work 241
8.2.1 Data management in the field 241
8.2.2 Parallelizing the acoustic modeling tools 243
8.3 Closing remarks 246

A Python MOOS and the LAMSS Python toolkit 249
A.1 The Python-MOOS bindings 250
A.1.1 Installation 250
A.1.2 Understanding the bindings 251
A.1.3 Getting started with Python-MOOS 253
A.1.4 Using Python-MOOS as part of a class 254

A.2 The LAMSS Python plotting utilities . . . 259
A.2.1 The uPlot superclass . . . 262

B LAMSS Docker 265
B.1 From VMs to containers . . . 266
B.2 The LAMSS Docker solution . . . 267

C Additional Python tools for LAMSS 271
C.1 The BELLHOP utility library . . . 271
C.2 The LAMSS utility library . . . 273
C.2.1 The Acoustic Comms tools . . . 273
C.2.2 The navigation and general log tools . . . 275

Acronyms 279

Glossary 281

Bibliography 283

List of Figures

1.1 Estimated age of sea ice in the Arctic during the month of March, shown biennially from 1984 to 2020. Based on data from the National Snow and Ice Data Center. . 38

1.2 Comparison of the generic Arctic environment (top) and a model based on field measurements from ICEX-16 (bottom). Sound speed profiles are shown on the left, and incoherent transmission loss estimates are shown on the right, for a source depth of 33 meters. 41

2.1 Models for acoustic propagation in the ocean. 54

2.2 A ray bending through a stack of layers, each with an increasing sound speed $c_i < c_{i+1}$. 56

2.3 Equal time fronts (black) and time field gradient (blue/arrows) for a linear sound speed profile and a source located at 1000m depth. 58

2.4 Two rays illustrate distinct viable solutions for the Eikonal equation in a simple uniform half-space. 60

2.5 A comparison of medical and ocean acoustic tomography. a) A sample volumetric model produced by Cone-beam Computed Tomography (CBCT), which is often used in medical applications to plan surgical procedures. b) Locations of two experiments employing ocean acoustic tomography in the North Atlantic, overlaid on a snapshot of sound speed at 300m depth derived from a high-resolution numerical ocean model. These volumetric reconstructions enable users to explore distinct cross-sections of a target volume, compared to non-separable 2D projections. Image credits: a) Jose D. Viquez, DDS, FRCD(C); b) Brian Dushaw, Ph.D. (released to public domain). 63

2.6 Spherical positioning uses travel time information directly, and thus requires knowledge about the system's synchronization. a) Single-beacon range shown as red circle; simple ranging is possible due to available timing information, but distinct positions cannot be isolated. b) Two-beacon position, with red circles marking true position (filled) and ambiguous solution (unfilled). c) 3-beacon solution shows how additional measurements help break the ambiguity and reduce uncertainty about the true solution. 67

2.7 Hyperbolic positioning uses *differences* in travel times, and does not require knowledge about the system's synchronization. a) A minimum of two beacons are needed to produce a time difference and a single hyperbola. b) Three beacons can produce two hyperbolas, using one node as a reference point for the time difference. c) In the 3-beacon system, the position estimate becomes underdetermined along the baseline extensions, where the hyperbolas become tangential (red line). 68

2.8 The MOOS autonomy middleware uses a centralized publish-subscribe architecture. 73

2.9 The MOOS-IvP autonomy software suite introduces the IvP Helm, which enables complex decision-making through multi-object optimization. 74

3.1 An illustration of fundamental concepts in beamforming. a) A 10-element line array in 2D space, with a plane wave input; the plane wave is traveling in the direction **a** (system state shown for $t = 0$). b) The signal recorded by each receiver as a function of time. c) The signal phase recorded for each receiver when processing a snapshot of the data (as shown in b). 85

3.2 Magnitude of the beam pattern obtained by scanning across steering angles θ_s, for the setup shown in Fig. 3.1. 88

3.3 Sample bearing and time-to-intercept signals produced by ships moving along a transect and a loiter pattern. The starting positions and initial steps are shown in green, while the closest point along both paths are marked with a red cross. In the time-series data, the intercept time for both transect and loiter are shown as a solid red vertical line, while additional approaches from the loitering vehicle are shown with dashed red lines. Bearing is given relative to the array's broadside, aligned with the vertical axis. 92

3.4 Behavior classification from bearing and time-to-intercept (TTI) signals detected at a receiving array; signals are produced by an external vehicle moving relative to the array. This comparison illustrates the challenges that come with conventional "feature design" solutions, as the signals exhibit distinct patterns which are generally visible to the human eye, but require additional processing before they may be classified by a computational algorithm. 94

3.5 Reference surveys for the Charles River Basin. (a) Soil surveys provided by the Natural Resources Conservation Service of the US Department of Agriculture can be used to identify the presence of materials such as sand (shown), clay, and silt in local substrates; the Merrimac-Urban land complex (blue), for example, has a weighted average of 82.8% sand content across all depth layers. (b) Bathymetric surveys such as the one shown, by the MIT Sea Grant College Program and the Charles River Alliance of Boaters, inform decisions about a propagation model's bottom boundary. 99

3.6 Acoustic models for riverbed characterization. (a) Transmission loss chart for the full field, assuming an isovelocity profile with $c = 1451$ and a uniform sand bottom starting at 5.8m (19ft). (b) Power detected by a receiver at a depth of 1 meter, as a function of range, for the three riverbed materials evaluated by the classifier. . . . 100

3.7 Schematic of the model preparation process for virtual sampling. 101

3.8 The power predicted by the acoustic propagation model (a) is randomly sampled, and noise is added to reflect sensor and process error. Each set of random samples is discretized in range to construct a virtual sampling vector (b), which is added to the training set. This process is repeated multiple times per model to ensure the training set contains enough sample vectors, and sufficient information about the model, to be used by the classifier. 102

3.9 Equipment used to collect acoustic data for the environment classifier. (a) Components for a stand-alone acoustic array that generally reproduces the capabilities available onboard an AUV; shows part of the anchoring system used to stabilize the array orientation. (b) Equipment loaded on an electric motorboat prior to deployment; shows the stand-alone array unit and the components that make up a traceable acoustic source node. 103

3.10 Schematic of the bootstrapping approach to determine the classifier performance. The solution consists of a data-driven voting system based on multiple random iterations of the k-NN classifier. 105

3.11 Performance of a k-NN classifier for riverbed material characterization. (a) Classification votes from 1200 repetitions with k=1 nearest neighbors and n=200 random data samples per trial. (b) Comparison of a random test vector built with n=200 samples from field measurements, versus one feature vector per class from the training set. The training set is assembled by virtual sampling from the acoustic models. 106

3.12 Performance of a k-NN classifier for bathymetry estimation. (a) Classification votes from 1200 repetitions with k=1 nearest neighbors and n=200 random data samples per trial. (b) Comparison of a random test vector built with n=200 samples from field measurements, versus one feature vector per class from the training set. The training set is assembled by virtual sampling from the acoustic models. 107

3.13 Performance of a k-NN classifier for riverbed density estimation. (a) Classification votes from 1200 repetitions with k=1 nearest neighbors and n=200 random data samples per trial. (b) Comparison of a random test vector built with n=200 samples from field measurements, versus one feature vector per class from the training set. The training set is assembled by virtual sampling from the acoustic models. 107

4.1 The Virtual Ocean Simulator runs in simulation and field deployments alike. In the field, real vehicles rely on a lower-fidelity version of the simulator to inform decisions that may be impacted by the environmental conditions. 114

4.2 The Virtual Ocean Simulator uses a nested modeling approach that exploits the timescales of change of the different features. Slow-changing features such as those reported by the different ocean models are sampled less frequently, while fast-changing features such as the element-level signals (which change as the vehicle travels) are sampled much more frequently. 115

4.3 NETSIM expands on the Virtual Ocean Simulator by using an instance of the simulator running a high-fidelity ocean model as a substitute for the physical ocean. Software-based and hardware-in-the-loop implementations are available. 117

4.4 NETSIM introduces physical modems as the key hardware-in-the-loop component. 118

4.5 Overview of the Integrated Communications and Navigation Network (ICNN). . 120

4.6 Overview of the system architecture for a payload autonomy implementation. . . 121

4.7 Payload Autonomy stack for AUV *Macrura*. (a) Side view of the distributed computing assembly that enables *Macrura*'s autonomy. (b) The author of this thesis, working on the payload in preparation for field testing ahead of ICEX-20. 122

4.8 Overview of the internal components in the payload autonomy module. 123

4.9 The set of Empirical Orthogonal Functions (EOFs) used as the basis for data compression during ICEX-20 field operations. These EOFs were computed from a subset of WHOI ITP data filtered by proximity to the operations region and the time of year. 126

4.10 Effect of sound speed model on the ray tracing output (middle) and corresponding impulse response estimate (right) used as a basis for ranging. Sound speed profiles are taken from HYCOM (left, dashed) and the Virtual Ocean framework (left, solid). 128

4.11 Exploration of the impulse response estimate based on the weight of a single EOF. The weight is set to a value between -5 and 5, for a total of 15 evenly spaced entries. 129

4.12 Exploration of the impulse response estimate based on the mixture of 2 EOFs. As in the single-EOF example, each weight is set to a value between -5 to 5, for a total of 15 evenly spaced entries each. The 225 combinations are ordered by setting both weights to their first (lowest) value, then scanning across one weight's values as a fine step (same color), and across the other weight's values as a coarse step (change of color). 131

4.13 Sound speed profiles (a) and normalized model-based impulse response estimates (c-e) for different environment approximations; impulse response from physical modem (b) was obtained from NETSIM, for a simulated vehicle located at 7.1km range and 70m depth. 133

5.1 Gantt chart showing the timeline (UTC) of ICEX 2020 experiments. 139

5.2 Timeline of acoustic transmission and reception events collected during ICEX 2020, identified by experimental subset. 141

5.3 Failure mode distribution for acoustic reception events at each node in the network. Failure modes are categorized by each event's PSK error code, as reported by the WHOI Micromodem. Events with bad frames represent a notable portion of the data. 141

5.4 Failure mode distribution for acoustic reception events at each node in the network, for each experimental subset. The particular distributions change with the varying spatial arrangements, after breaking down the data set into smaller windows. However, it can be observed that the events with bad frames continue to represent a notable portion of each subset. 142

5.5 Number of impulse response estimates collected, compared to number of events detected per node. Limitations in the radio infrastructure used to communicate with remote nodes required that the collection of impulse response estimates be disabled on those modems, while in-situ mission reconfiguration impacted the collection of samples from the camp's modem. 144

5.6 Visualization of the TDMA scheme used for ICEX 2020. The first slot in each 4-minute cycle is reserved to ping U.S. Navy submarines in the area as part of coordinated operations. The remaining slots are assigned to the topside and vehicle systems in alternating fashion. The 15-second slot width supports operations in a 10 x 10 km grid, with enough buffer time at the end of each slot to clear late-arrival paths before the next transmission. 145

5.7 Acoustic transmission: timeline of clock-related events. Transmissions are generally robust to modem clock misalignment, since they depend on: (1) the timing of the transmit command coming from the topside computer, and (2) the pulse-per-second signal produced by the node's time-keeping solution (GPS receiver or synchronized CSAC). The modem clock provides timestamps for transmission statistics, but does not control the transmission events. 147

5.8 Acoustic reception: timeline of clock-related events. The modem uses its internal clock to produce timestamps for reception events. Misalignment of the modem clock relative to GPS time jeopardizes the best opportunity for high-precision timing, as the serial interface to the topside computer may introduce significant additional delays. 149

5.9 Diagram illustrating the conventional serial interface to acoustic modems. When communicating over a wired interface, each modem uses a designated interface and there is less chance of failure due to dropped packets. 150

5.10 Diagram illustrating the radio link as a substitute to the serial interface for remote acoustic modems. The latency of the radio interface increases the chances for delayed or dropped packets, impacting the reliability of the interface. 152

5.11 Time skew by node, as collected from explicit time queries sent to the node's Micromodem. The skew is given with respect to the control computer's time; operator workstation for surface nodes and payload computer for the vehicle. 153

5.12 Distribution of transmission events along the TDMA cycle, based on the timestamps reported by the modem in the transmission statistics message (left) and on the time of the log entry, given by the control computed (right). When using the date and time recorded within the modem's transmission statistics, some events that are apparently not aligned with the expected TDMA cycle stand out; when looking at the same data relative to the log times, it becomes clear that the transmissions sent out from the modem at camp were not, in fact, delayed. This timing mismatch provides additional insight into the health of the ICNN's synchronization, beyond that attained from explicit time queries. 155

5.13 Time of arrival of reception events by node, given in the context of the TDMA scheme. Transmissions are expected to occur 1 second after the start of each window (shaded regions). Successful arrivals are front-loaded within each TDMA slot; late arrivals – with travel times of 4-6 seconds, for example – fail primarily due to some number of bad data frames in the signal, but appear to contain valid headers. 157

5.14 Time of arrival of reception events during the modem test experiment. The static nature of the experiment is captured by the narrow and generally consistent peaks. This set reveals few arrivals with travel times greater than 4 seconds. 159

5.15 Time of arrival of reception events during the vertical line array experiment. The VLA experiment consisted of a series of shorter static experiments, where the vehicle was held at different depths for about 30 minutes at a time. The effect of changing depth is reflected in the appearance of the late arrivals within the vehicle's transmission window, while the quasi-static nature of the experiment is again reflected by the narrow peaks for successful arrivals. 160

5.16 Time of arrival of reception events during AUV operations. The dynamic nature of AUV operations is captured by the widening of the peaks for successful arrivals, and the effect of changing vehicle depth remains visible through the distribution of late arrivals. 161

5.17 Time of arrival of reception events during the final phase of AUV operations, where the vehicle had stalled. This set effectively amounted to another static experiment; the arrival structure once again returns to tall, narrow peaks. With the vehicle settled in place under the surface ice, there are once again few arrivals with travel times greater than 4 seconds. 162

5.18 Vehicle track during an 11km run, based on the vehicle's onboard sensor fusion system. The vehicle ingests error correction updates from the ICNN, and combines them with the onboard DVL and INS measurements, as well as a self-correcting hydrodynamic model. 165

5.19 ICEX-20 vehicle deployment (a) and recovery (b) sites. 166

6.1 Effect of sound speed model on the ray tracing output (middle) and corresponding impulse response estimate (right) used as a basis for ranging. Sound speed profiles are taken from HYCOM (left, dashed) and the Virtual Ocean framework (left, solid). Replicates Fig. 4.10 locally, for reader's convenience. 170

6.2 Overview of the modem connections recorded during the modem test experiment. Reception event counts are shown as fractions, with the number of successfully decoded entries on top and the total number of detections on the bottom. 174

6.3 Timeline of events and configuration during the modem test experiment. The group velocity estimates are based on either the baseline sound speed profile, or one obtained by fitting the model to a CTD cast with the EOF framework. Transmitter and receiver activity is categorized by depth layer, with the camp modem lowered to approx 20 meters depth, and the buoys operating at either 30 or 90 meters. 175

6.4 Runtime simulations are based on the Virtual Ocean Simulator, which uses a nested modeling approach that exploits the timescales of change of the different features. The acoustic model is evaluated at an intermediate timescale, producing data for a local grid near the target node. The element-level impulse response estimate is computed by plane wave expansion of this pre-computed grid. 179

6.5 Ray traces for a source depth of 20 meters (Camp), under different environmental models. The eigenrays produced by ray tracing are matched to field measurements by the estimated travel time, to illustrate the most likely paths taken by the acoustic signals. Bottom bounce paths are not shown. 187

6.6 Ray traces for a source depth of 30 meters (North and South buoys), under different environmental models. The eigenrays produced by ray tracing are matched to field measurements by the estimated travel time, to illustrate the most likely paths taken by the acoustic signals. Bottom bounce paths are not shown. 188

6.7 Ray traces for a source depth of 90 meters (East and West buoys), under different environmental models. The eigenrays produced by ray tracing are matched to field measurements by the estimated travel time, to illustrate the most likely paths taken by the acoustic signals. Bottom bounce paths are not shown. 189

6.8 Distribution of vehicle position corrections computed by the ICNN during the modem test experiment. The corrected position estimates are the trilateration results computed by the ICNN, which are in turn derived from the recorded times of arrival and respective model-aided range conversions; position corrections are then transmitted as marginal differences with respect to the position estimate transmitted by the vehicle. 191

6.9 Acoustic positioning via trilateration, using model-aided range estimation. The reception events shown here occurred during the modem test experiment, while the northernmost buoy (h2) was acting as the virtual vehicle. The event contains a viable 3-beacon solution with a correction of 20.4 meters from the initial position estimate transmitted by the virtual vehicle. The associated trilateration error had a magnitude of 11.6 meters. 193

6.10 Trilateration with 3 beacons. The Minimum Bounce approach is unable to accurately estimate the effective group velocity for one of the acoustic links, leading to a large positioning error that can be easily recognized in this view. The Nearest Bounce approach is able to recognize the alternate path and adjust accordingly. . . 195

6.11 Trilateration with 3 beacons. The difference in performance of the various algorithms is not apparent in this view; however, the Nearest Bounce approach clearly outperforms its counterparts in approaching the GPS solution. 195

6.12 Distribution of vehicle position corrections computed in post-processing for the Minimum Bounce and Nearest Bounce algorithms. The MBC is shown as implemented during ICEX-20, with the local plane wave expansion, as well as in the simplified form used throughout this analysis. 197

6.13 Overview of the modem connections recorded during the experiments involving vehicle operations (VLA, tethered and untethered runs). Reception event counts are shown as fractions, with the number of successfully decoded entries on top and the total number of detections on the bottom. The vehicle depth is variable, and spans 0-125 meters. 199

6.14 Distribution of the reception events recorded across the ICNN, arranged by the number of individual detections available per individual transmission event. Shows only instances where the AUV is the transmitting node. 201

6.15 Distribution of vehicle position corrections computed by the ICNN during AUV operations. The corrected position estimates are the trilateration results computed by the ICNN, which are in turn derived from the recorded times of arrival and respective model-aided range conversions. 202

6.16 Distribution of vehicle position corrections computed in post-processing for the Minimum Bounce and Nearest Bounce algorithms. The MBC is shown as implemented during ICEX-20, with the local plane wave expansion, as well as in the simplified form used throughout this analysis. 203

6.17 Distribution of vehicle position corrections computed in post-processing for the Minimum Bounce and Nearest Bounce algorithms. The MBC is shown as implemented during ICEX-20, with the local plane wave expansion, as well as in the simplified form used throughout this analysis. 204

7.1 Diagrams illustrating the key differences between PINNs used for data-driven solution of PDEs (top), versus the conventional label-matching model for artificial neural networks (bottom). 216

7.2 Diagram illustrating the PINN framework to learn the factored Eikonal equation. 225

7.3 Learning the travel time field for a Munk-shaped environment (case 1). The sampled field is limited to 1 km in range and 500 m in depth; the sound speed minimum is located at 100 m depth. The top and bottom plots show two distinct repetitions of the training process; some variability appears across repetitions, as expected. In general terms, though, the learning process and results are fairly repeatable for this example. 226

7.4 Ray tracing solution for direct paths in the Munk-shaped profile with the minimum at 100m, computed via BELLHOP. Traces with steep launch angles and surface bounces are not shown. 227

7.5 Learning the travel time field for a Munk-shaped environment (case 2). The sampled field extends to 2 km in range and 4 km in depth; the sound speed minimum is located at 500 m depth. With increasingly sparse sampling of the function space, the learning process becomes less consistent across repetitions; one instance may produce a smooth travel time field (top) while another execution of the same setup may produce a jagged field (bottom). 228

7.6 Learning the travel time field for a Munk-shaped environment (case 3). The sampled field is further extended to 6 km in range and 5 km in depth; the sound speed minimum is located at 500 m depth. With even sparser sampling of the function space, the results of the training process become less reliable still. Once again, top and bottom illustrate two distinct repetitions of the training process, one producing a smooth field and the other a jagged field. 229

7.7 Ray tracing solution for direct paths in the Munk-shaped profile with the minimum at 500m, computed via BELLHOP. Traces with steep launch angles and surface bounces are not shown. The extents are matched to the longest-range scenario (case 3); the bounds marking the solution for case 2 are also shown. 230

7.8 Coverage of the range-depth domain for one-way travel time samples collected by AUV *Macrura* (top) and the ICNN buoys connected to the topside computer (bottom), for acoustic links between topside and the vehicle; buoy-to-buoy samples are not shown. Ranges are given relative to the ICNN buoys, with the different depths illustrated. The markers indicating the range and depth of *Macrura* are color-coded by the depth layer of the ICNN transmitter or receiver involved in the link. 234

7.9 Sample selection from the set of acoustic data recorded by the topside-controlled ICNN. A k-means approach with 8 clusters is used to identify representative regions in the set. 235

7.10 CTD casts collected during ICEX-20, along with the EOF fit obtained via conventional least-squares. The rows of the travel time observation matrix B are scaled and shifted to align with the data, illustrating how the training process captures perturbations of the EOF-fitted profile. 237

A.1 Diagram illustrating the lamssDC server architecture. Plotting applications inherit from the uPlot superclass and send requisite data to the lamssDC server via a websocket connection. 261

List of Tables

2.1 Numerical coefficients for the Chen and Millero equation for the speed of sound in seawater 51

3.1 Parameters for acoustic modeling, by material 99

6.1 Range estimation error for events recorded during the modem test experiment, based on data and predictions produced *in situ*. Metrics are reported with respect to the absolute error. 176

6.2 Range estimation error for events recorded during the modem test experiment, which may be considered consistent with bottom-bounce paths. Metrics are based on data and predictions produced *in situ*, and are reported with respect to the directional error. 176

6.3 Range estimation error for 1232 acoustic communication events recorded during the modem test experiment, using supplementary simulations. Performance improvements of the Nearest Bounce Criteria over the Minimum Bounce Criteria approach an order of magnitude. Metrics are reported with respect to the absolute error. 181

6.4 Event data collected from the ICEX-20 modem test data logs for two buoy-to-buoy connections: between (1) the East and West buoys, and (2) the South and North buoys. Note that *in-situ* simulation data is only available for the 11 events between the East and West buoys; the predicted one-way travel time (OWTT) for these events is roughly half of the measured value. 183

6.5 Range estimation error for the 11 events recorded traveling from the easternmost buoy to the westernmost buoy during the modem test experiment. Metrics are reported with respect to the directional error. 184

6.6 Range estimation error for the 10 events recorded traveling from the southernmost buoy to the northernmost buoy during the modem test experiment. Metrics are reported with respect to the directional error. 184

6.7 Predicted travel times by number of boundary interactions (bounces), for the EOF fit environment. 185

6.8 Trilateration data for the example scenario shown in Figure 6.9, based on the Minimum Bounce Criteria and data collected during the modem test experiment. The estimated ranges used for the propagation model are based on the latest position estimates received from the virtual vehicle at the time of model execution. 191

6.9 Trilateration results for the example scenario shown in Figure 6.10. The limitations of the Minimum Bounce Criteria lead to errors in the order of 1 kilometer. . . . 196

6.10 Trilateration results for the example scenario shown in Figure 6.11. The Nearest Bounce Criteria outperforms the Minimum Bounce Criteria. 196

6.11 Distribution of reception events, arranged by the number of times a given receiver detects the same transmission event. The instances where a beacon detects the same transmission only once are generally successful receptions; instances where the beacon reports three or more detections for the same transmission window are generally associated with an error such as incomplete data frames or headers. . . 200

A.1 A sampling of basic bindings handled by the `python-moos` project. Parent namespaces are given in parenthesis for each group; the `msg` entry is an item from the vector returned by `pymoos.comms.fetch()`. 252

B.1 Ports exposed in LAMSS Docker image. 269

1 Introduction

> *"If I were to choose a single phrase to characterize the first century of modern oceanography, it would be 'a century of undersampling.' The most profound effect of satellite oceanography has not been the resulting new sensor packages (and these have been remarkable), nor the global coverage, but rather that for the first time ocean processes were adequately sampled."*
>
> – Walter Munk[1]

Lake Geneva was the setting for the earliest recorded attempt to measure the speed of sound in water. The experiment, conducted by Daniel Colladon and Charles Sturm in 1827, produced a value of 1435 m/s for water at 8°C – a result that has long been regarded as remarkably for how close it is to modern values. It was nearly a century after this first measurement, during World War I, that the first investigations concerned with the particulars of sound propagation underwater were first conducted.[2]

The field of ocean acoustics has evolved greatly from its early days. Continued research efforts over the years yielded multiple theoretical approaches for the problem of propagation modeling. Field and theoretical experiments yielded multiple formulations for the value of the speed of sound. Skylab's RADSCAT and then SeaSat, both launched in the 1970's, marked the beginning of satellite oceanography.[3] As renown physical oceanographer Walter Munk said, the arrival of satellite tech-

[1] Halpern, "1. Oceanography before, and after, the Advent of Satellites".
[2] Lasky, "Review of undersea acoustics to 1950"; Urick, *Sound propagation in the sea*; Jensen, Kuperman, Porter, and Schmidt, *Computational ocean acoustics*.
[3] Monaldo, Jackson, and Pichel, "SEASAT TO RADARSAT-2: RESEARCH TO OPERATIONS".

nology had a profound impact in our understanding of the oceans because it provided us with an unprecedented standard of sampling for the processes affecting the world's largest water masses. And yet, advances in machine learning techniques over the past few decades have wholly redefined the concept of "adequate sampling."

1.1 A new and changing Arctic

The Arctic has captivated the minds of mankind for millennia – and what else could be expected of a place mysterious enough to inherit three names from Ancient Greece? Pytheas, a Greek geographer, is said to have sailed north until he encountered a frozen ocean, where man could "neither sail nor walk." From here, he is said to have sailed back south to find land, and the place he encountered he called Thule - the land beyond beyond all known lands. Of the two other names, this can be said: that one was quite an apt description of the reality we know today, and that the other held the power of inspiration over those who would risk it all for a utopian vision. These names were Arctic, from *Arktikos* (αρκτικοσ), meaning "of the great bear"; and Hyperborea, the region beyond the kingdom of Boreas (the god of the north wind).[4]

The cartographer Mercator, on the basis of stories that made their way to him, drew the heart of the Arctic on his maps as a place called the Lodestone Mountain, capable of ripping the nails out of ships with its magnetic pull; or alternatively as a maelstrom so powerful, not even the strongest winds could save a ship caught in its grip. But, where Pytheas encountered a frozen ocean, Mercator drew instead an open ocean surrounding those iconic landmarks which embodied the ocean's attraction turned fatal. This peculiar choice persisted through the centuries, and inspired generations of explorers to see beyond legend; to seek not the utopia of the Hyperboreans or the excitement of ocean-borne crucibles, but the more realistic dream of a northern passage.[5]

[4] Schulz, "Literature's Arctic Obsession".
[5] Schulz, "Literature's Arctic Obsession".

This vision of a blue Arctic – of one without a blanket of ice spread atop it – is perhaps one of the great ironies about this northernmost domain. It has been centuries since Mercator's maps inspired the pursuit of a northern passage, which has only in recent decades become a likely reality for our future. Figure 1.1 shows how the age of sea ice has changed over the years, according to satellite imagery as well as satellite tracking of buoys placed on the sea ice, as part of the International Arctic Buoy Program.[6]

The observations from satellite data collected by the NSIDC are well-complemented by the findings presented by Chen, where a detailed comparison of two sets of acoustic measurements (collected in the Beaufort Sea during the spring of 1994 and spring of 2016) shows that changes in the Arctic soundscape are indeed consistent with two distinct models for surface source distributions. The data from the Sea Ice Mechanics Initiative of 1994 (SIMI-94) was found to be consistent with a model of uniformly distributed sources, where many sustained transients are generally expected – the analogy here being that thicker, stronger ice is generally associated with this model for its ability to sustain the kind of loading that would produce these transient signals. In contrast, the data collected by MIT's Laboratory for Autonomous Marine Sensing Systems (LAMSS) during the US Navy's Ice Exercise of 2016 (ICEX-16) was determined to lack much of the transient characteristics of the SIMI-94 data, and to be more spatially discrete; that is, the field measurements from 2016 are better described by the simpler model of a single source at some range from the receiver. The latter model would be consistent with localized phenomena such as ridge formation, and allows for the argument that thinner, weaker ice will tend to break apart with relative ease and therefore will not sustain the transient signals of the uniform source distribution model.[7]

[6] Tschudi, Meier, Stewart, Fowler, and Maslanik., *EASE-Grid Sea Ice Age, Version 4*; Tschudi, Meier, Stewart, Fowler, and Maslanik., *Polar Pathfinder Daily 25 km EASE-Grid Sea Ice Motion Vectors, Version 4*.

[7] Chen, "Ambient Acoustics as Indicator of Environmental Change in the Beaufort Sea: Experiments & Methods for Analysis".

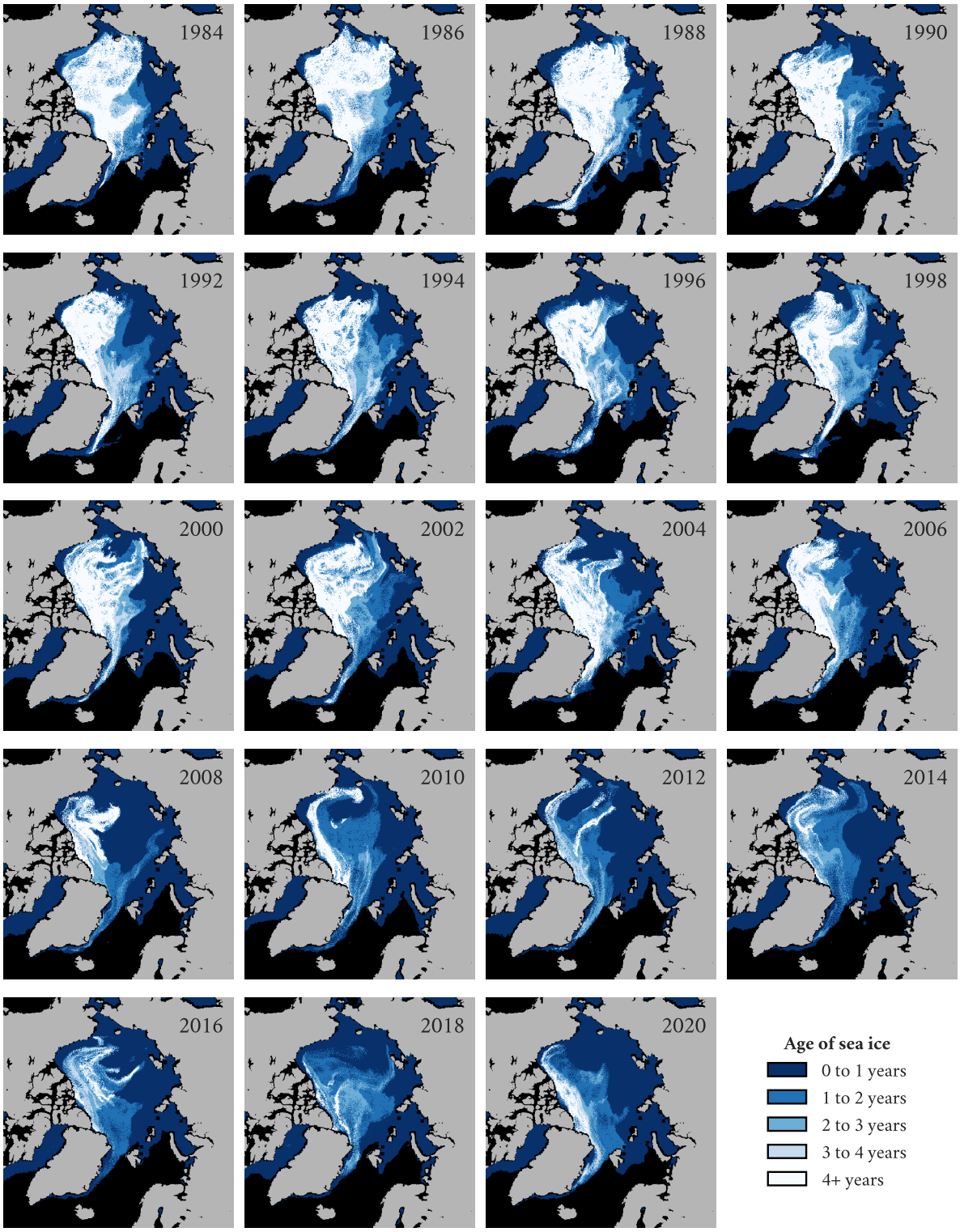


Figure 1.1: Estimated age of sea ice in the Arctic during the month of March, shown biennially from 1984 to 2020. Based on data from the National Snow and Ice Data Center.

1.2 The need for real-time environmental adaptation

The changes in the age, thickness, and distribution of the ice cover atop the Arctic Ocean directly impact the soundscape of those polar waters; but the ice alone does not account for all that has changed in the Arctic. Another phenomenon that has impacted the Arctic is the appearance of a warm layer of water coming in from the Pacific – sitting at a depth of around 70-80 meters, this layer of warm water directly affects the sound speed profile and thus the performance of acoustics-driven sensing, navigation and communications infrastructure. This phenomenon, known as the *Beaufort Lens*, was observed by Woods Hole Oceanographic Institution (WHOI) in 2013, through a network of specialized remote sensors designed to collect environmental measurements; the WHOI Ice Tethered Profiler (ITP) program. The Beaufort Lens was also observed during underwater vehicle operations conducted by MIT LAMSS as part of ICEX-16[8] and has been the subject of multiple studies to date; an interesting example of work on the Beaufort lense is a recent study that evaluated ocean circulation models against acoustic measurements from the Canada Basin Acoustic Propagation Experiment (CANAPE) and data from the ITP program in order to assess the effects of variablility in the Pacific summer water layer and the ice cover on underwater sound ducting at large scales.[9]

The impact of the Beaufort Lens on acoustic communications is illustrated in Figure 1.2, for a 3.5kHz source at a depth of 33 meters. The top half illustrates a generic Arctic environment based on historical trends; this environment, with a generally increasing sound speed value as a function of depth (top left), is what was generally expected for ICEX-16. Even with some small variability added to this generally increasing environment, the operational paradigm indicated that a vehicle

[8] Schmidt and Schneider, "Acoustic communication and navigation in the new Arctic — A model case for environmental adaptation".

[9] Duda, Zhang, and Lin, "Effects of Pacific Summer Water layer variations and ice cover on Beaufort Sea underwater sound ducting".

operating within the upper 200 meters[10] should be able to maintain a reliable acoustic path to a source at 33m depth, so long as the range between the two nodes remained within 6-7 kilometers (top right). Furthermore, the 6-7km range predicted in the operational paradigm would have been consistent with historical performance.[11] As the range extends beyond 7km in this generic Arctic environment, it becomes increasingly difficult to discern particular paths between the source and some receiver in the transmission loss field, and this property of the model translates to the expectation that communication performance will be significantly degraded at these longer ranges.

The bottom half of Figure 1.2 shows the sound speed profile based on measurements collected at Camp Sargo, the name given to the ice camp during ICEX-16. These in-situ measurements captured the Beaufort lens, which is reflected in the sound speed profile as a local maximum (bottom left) – a feature often described as a knee, for its appearance when plotted. Where the generally increasing sound speed profile of the generic Arctic produced upward-refracting paths, the appearance of a local maximum in the ICEX-16 data tends to divide the water column into two ducts: one in the surface layer, above the knee; and another one that tends to trap sound below the knee, roughly between 75m and 300m depth in the case shown. The transmission loss field for the ICEX-16 environment captures these ducts rather clearly (bottom right).

A key difference between these two exemplary environments is that, while the generic Arctic case had reliable paths out to 6-7km, the ICEX-16 case exhibits a deep shadow zone – a region with very large transmission loss, which is to say, where very little energy from the source can be detected – stretching between the 2km and 5km range markers. If the vehicle is constrained to operate within the upper 200m of the water column, then this shadow zone means the vehicle

[10] Mechanical design for underwater operations is a fascinating domain of engineering, which includes considerations such as whether to use air-filled pressure vessels or oil-filled, pressure-compensated chambers. The vehicle used for ICEX-16 and ICEX-20, for example, holds two air-filled pressure vessels that contains the vehicle's computers; these air-filled vessels play a part in the research vehicle's depth rating being limited to about 200 meters. However, further discussion about mechanical design is beyond the scope of this thesis.

[11] Schmidt and Schneider, "Acoustic communication and navigation in the new Arctic — A model case for environmental adaptation".

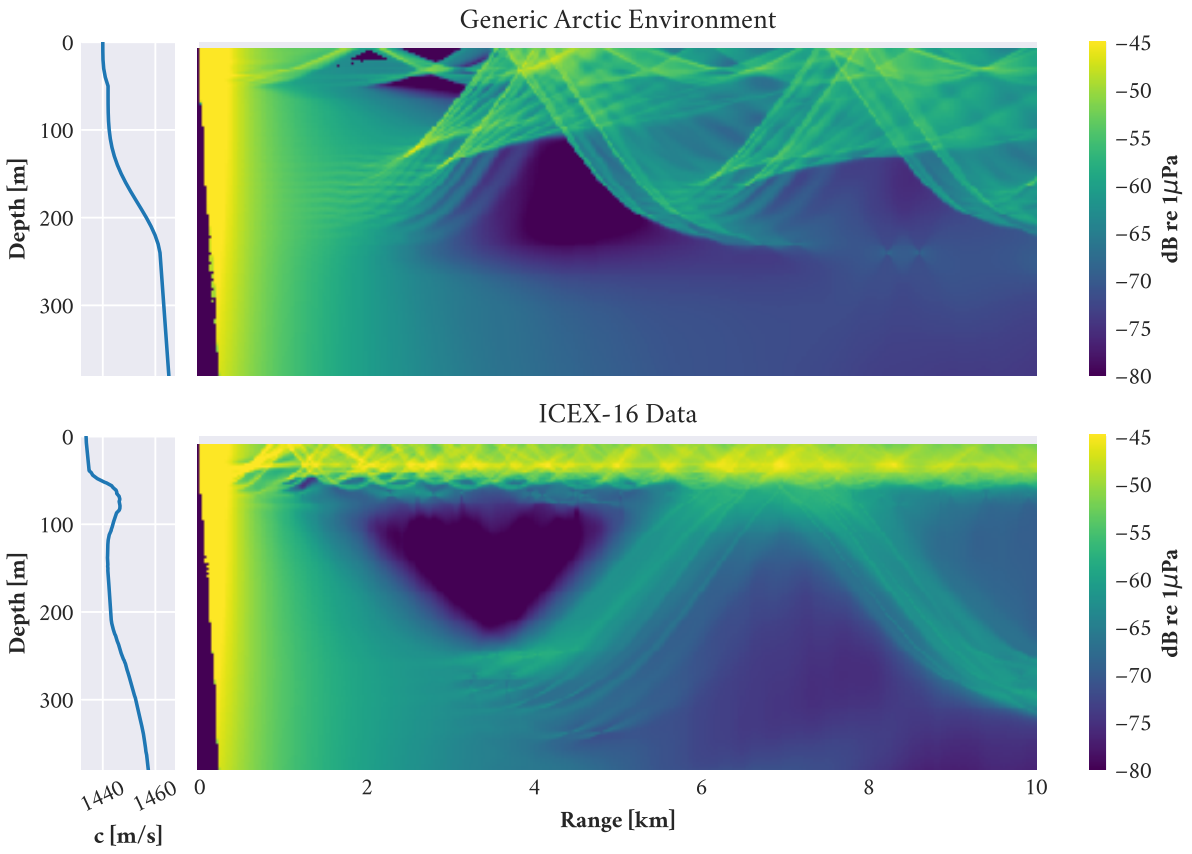


Figure 1.2: Comparison of the generic Arctic environment (top) and a model based on field measurements from ICEX-16 (bottom). Sound speed profiles are shown on the left, and incoherent transmission loss estimates are shown on the right, for a source depth of 33 meters.

would have to go it alone for an important portion of its mission. The numerous surface interactions and subsequent multipath in the surface duct would presumably render that upper region highly unreliable for communications and navigation, while the deeper acoustic paths travel beyond the reach of the vehicle.

There is, of course, a seemingly simple solution to the limitation presented in Figure 1.2: what if we could simply adjust the source depth to address the shadow zone limitation; would this change allow us to insonify the previously unreachable portion of the water column? After all, this introduction has thus far discussed the presence of reliable acoustic paths anywhere in the upper 200m of the water column (which are reachable by the vehicle, as is the case in the generic Arctic environment) as a sufficient condition of viability. This initial treatment comes from the assumption that, in an updated environment such as that of ICEX-16, (i) there exists some alternate source depth which would enable insonification of the entire range-depth domain required for a mission, and that (ii) the vehicle autonomy would then seek to operate within the valid subset of depths for any given range. The reality of vehicle operations, however, is a much more complicated endeavor than that.

It has already been demonstrated, through additional model and simulation work, that changing the source depth alone (to a different static value), or even configuring multiple sources at discrete depths, may not suffice to fully restore communication capabilities in the ICEX-16 environment.[12] Additionally, there is the risk that the chosen environmental model – however well-founded on prior data – is itself an inadequate representation of reality; much like the generic Arctic model was a poor estimate of the true field, going into ICEX-16. Thus, a more promising approach to handle the challenges of the new Arctic, as they were experienced during the 2016 deployments, would involve enhancements in two key areas. First, there is a need for the system to realize real-time environmental adaptation; that is, the vehicle or vehicle operator should be able to detect whether

[12] Schmidt and Schneider, "Acoustic communication and navigation in the new Arctic — A model case for environmental adaptation".

the internal environmental model is sufficient to inform the decision-making process during a mission and, if it is not good enough, then either adjust or replace said model with a more appropriate one. Second, the vehicle or vehicle operator should be able to relay these changes to its own internal environmental model to its remote counterpart in a compact and efficient way, such that the communications link between nodes is not overloaded. By sharing the updated environmental model, both vehicle and operator[13] can then rely on a common understanding of the environment to anticipate the performance of the communication network and adjust their navigation objectives, maintaining contact as needed throughout the mission and thus addressing the key challenge identified during ICEX-16.

1.3 Problem statement and outline

This thesis focuses on the first of those two fundamental needs identified during the ICEX-16 deployments; it explores the application of machine learning techniques towards the ultimate goal of approaching **real-time environmental adaptation for Autonomous Underwater Vehicles (AUVs)**, specifically by pursuing a constrained fit of the true local sound speed profile (SSP) that may be used to inform navigation and communication decisions onboard the vehicle.

In order to tackle this challenge, a multidisciplinary approach is needed. To that end, this introductory chapter first presented the motivation behind this work in the context of a prior AUV deployment conducted in the Arctic ocean. Relevant theory and related works from the various underlying fields are introduced in Chapter 2. As a primer to machine learning techniques used throughout this work, Chapter 3 presents two studies conducted on small-scale AUVs. The experimental system and foundational work for the Arctic deployment at the heart of this project are presented at length in Chapter 4. Chapter 5 dissects the data collected under the ice, out in the

[13] This could also be adapted for any number of nodes in a larger network, such as vehicle swarms; some considerations, such as reserving sharing capabilities for the operator or lead-vehicle only, may be needed to optimize use of the communication channel in this scenario.

Beaufort Sea. In Chapter 6, the contributions from Chapter 4 are related to the data presented in Chapter 5, in the light of implications for navigation performance. From there, Chapter 7 builds on how physics-informed models may be linked to machine learning techniques, to correct internalized models and pursue online environmental adaptation. Finally, Chapter 8 presents conclusions and closing remarks.

2 Background

"No one really starts anything new, Mrs. Nemur. Everyone builds on other men's failures. There is nothing really original in science. What each man contributes to the sum of knowledge is what counts."

– Daniel Keyes, *Flowers for Algernon*

The sentiment captured in Daniel Keyes' award-winning text[1] certainly fits the pursuit of advances in fields as interdisciplinary as marine robotics – be it failure or success, we build on the work of those who came before us in the hopes that our own contributions will move ours fields forward, even if only by one small step.

The following sections seek not to be a holistic exposition of all those domains involved in the deployment of an advanced marine robotic system[2], but rather each is a dive into one or another of those areas which ultimately provide the basic foundation from which the contributions of this thesis may be developed and understood.

[1] *Flowers for Algernon* was first published as a short story, which won the Hugo Award for Best Short Story in 1960; the text was later expanded into a novel which was published in 1966 and was joint winner of that year's Nebula Award for Best Novel.

[2] See footnotes in Chapter 1 for an example of a domain relevant to these deployments, but not covered in detail throughout this thesis.

2.1 Estimating the speed of sound in water

Much work has been done in the pursuit of a general expression for the speed of sound in water since the early experiment by Colladon and Sturm. Most often, these contributions have been largely based on empirical approaches involving either laboratory or field measurements, and some have been challenged or even deemed obsolete after publication. This section presents a brief history of select papers on the pursuit of a mathematical expression for the speed of sound in water, with emphasis on the equations most relevant to this thesis.

> *"In 1923, Stephenson timed the transmission of a bomb explosion between two hydrophones a known distance apart in Long Island Sound, while at about this same time, Heck and Service measured the travel time of pulses reflected from the sea bottom and determined the velocity from depth obtained by wire sounding. These measurements established the velocity of sound in sea water for many years thereafter."*
>
> – Robert Urick[3]

As Urick goes on to explain, the years following World War I saw a focus shift from theory to the development of better equipment with which to explore the physical domain, and it was only around the 1930's that enough sea-going experience had come to show the variability in measurements weren't solely an artifact of the equipment available, but rather a characteristic of the environment. Thus, this brief recounting of history shall skip ahead to the late 1950's, when Wilson first presented his experiment to measure the speed of sound in distilled water, and later presented his results as a function of temperature, pressure and salinity[4] – his results were subsequently updated to account for a wider range of salinity values.[5]

[3] Urick, *Sound propagation in the sea*.
[4] Wilson, "Speed of Sound in Distilled Water as a Function of Temperature and Pressure"; Wilson, "Speed of Sound in Sea Water as a Function of Temperature, Pressure, and Salinity".
[5] Wilson, "Equation for the Speed of Sound in Sea Water".

Wilson's work substantiated prior work conducted at the Naval Research Laboratory (NRL), which established a discrepancy against the then-current standard set by Kuwahara's look-up tables in the late 1930's, and his solution consisted essentially of an empirical equation obtained by fitting 581 measurements of sound speed performed on samples collected by the Navy Hydrographic Office. His paper also included parameterized look-up tables for each of the correction values in his equation. Prior to Wilson's work, Greenspan[6] had provided tables with sound speed as a function of temperature in distilled water. Follow-up work on Wilson's equation, presented in 1971, showed that there were some inconsistencies with the laboratory measurements used to produce the equation, thus rendering it obsolete.[7]

Having provided the aforementioned NRL work on Kuwahara's table, which Wilson substantiated, Del Grosso revisited this challenge in the 1970's with 3 key contributions. The first work, conducted by Del Grosso and Mader, addressed the derivation of an equation from indirect measurements of the speed of sound in sea water samples, collected via interferometry rather than direct pulse measurements. He also noted that storage may have a non-negligible impact on the sample's composition and thus its usefulness for this kind of work.[8] His second work in this set consisted of a series of tables that elaborate on the equation from his first work, emphasizing parameter ranges of particular interest such as those related to the open ocean as well as the Mediterranean and Red Sea. This second work also discussed cautionary arguments which had been raised with regards to the concept of salinity – this included how salinity may be related to chlorinity and conductivity.[9] Del Grosso's third work consolidated prior efforts into a formulation that was valid for sea water and pure water alike, yielding the 1974 equation then referred to as NRL II[10] and given here as Eq. 2.1.

[6] Greenspan and Tschiegg, "Speed of sound in water by a direct method"; Greenspan and Tschiegg, "Tables of the Speed of Sound in Water".
[7] Anderson, *Sound speed in seawater as a function of realistic temperature – salinity – pressure domains*; Dushaw, Worcester, Cornuelle, and Howe, "On equations for the speed of sound in seawater".
[8] Del Grosso and Mader, "Speed of Sound in Sea-Water Samples".
[9] Del Grosso, "Tables of the speed of sound in open ocean water (with Mediterranean Sea and Red Sea applicability)".
[10] Del Grosso, "New equation for the speed of sound in natural waters (with comparisons to other equations)".

The reference value C_{000} and single-parameter values $\Delta C_{\bullet}$ of Del Grosso's equation are expanded in Eq. 2.2, while the multi-parameter term ΔC_{STP} is expanded in Eq. 2.3.

$$C_{STP} = C_{000} + \Delta C_T + \Delta C_S + \Delta C_P + \Delta C_{STP} \tag{2.1}$$

$$C_{000} = 1402.392$$

$$\begin{aligned}
\Delta C_T = {} & 0.501109398873 \times 10^{1} T \\
& - 0.550946843172 \times 10^{-1} T^2 \\
& + 0.221535969240 \times 10^{-3} T^3,
\end{aligned}$$

$$\begin{aligned}
\Delta C_S = {} & 0.132952290781 \times 10^{1} S \\
& + 0.128955756844 \times 10^{-3} S^2,
\end{aligned} \tag{2.2}$$

$$\begin{aligned}
\Delta C_P = {} & 0.156059257041 \times 10^{0} P \\
& + 0.244998688441 \times 10^{-4} P^2 \\
& - 0.883392332513 \times 10^{-8} P^3,
\end{aligned}$$

$$
\begin{aligned}
\Delta C_{STP} = &- 0.127562783426 \times 10^{-1} TS \\
&+ 0.635191613389 \times 10^{-2} TP \\
&+ 0.265484716608 \times 10^{-7} T^2 P^2 \\
&- 0.159349479045 \times 10^{-5} TP^2 \\
&+ 0.522116437235 \times 10^{-9} TP^3 \\
&- 0.438031096213 \times 10^{-6} T^3 P \\
&- 0.161674495909 \times 10^{-8} S^2 P^2 \\
&+ 0.968403156410 \times 10^{-4} T^2 S \\
&+ 0.485639620015 \times 10^{-5} TS^2 P \\
&- 0.340597039004 \times 10^{-3} TSP
\end{aligned}
\tag{2.3}
$$

Following Del Grosso's work, Medwin presented a simplified version of the NRL II equation with the intent of providing an updated textbook-style formulation. Medwin's argument was that while Eq. 2.1 expands to 19 terms with a large number of significant figures (12 for all but one of the terms), his formulation had a total of 9 terms and far fewer significant figures – enough to compress the equation to one or two lines and render it suitable for use in handheld calculators. This kind of convenience, of course, came at the cost of accuracy. Thus, Medwin claimed his equation would be a suitable substitute to NRL II "in cases where deviations of the order of tenths of 1 m/sec are tolerable".[11]

In 1977, only a few years after Del Grosso's equation was published, Chen and Millero argued that work conducted in the intervening period presented a paradox. Per their findings, Del Grosso's measurements at 1-atm were the more reliable ones, acknowledging the issues in Wilson's data for the same pressure; but it was Wilson's work, rather than Del Grosso's, that they found more reli-

[11] Medwin, "Speed of sound in water: A simple equation for realistic parameters".

able at high pressure. Thus, Chen and Millero proceeded to derive a new equation from their own measurements, which were collected by using an ultrasonic velocimeter.[12] The resulting expression was later endorsed by UNESCO,[13] and is given here as Eq. 2.4. The sound speed in pure water $C_W(T, P)$ as well as the remaining terms related to salinity are expanded symbolically in Eq. 2.5, while the numerical values for the various coefficients are given in Table 2.1.

$$C(S, T, P) = C_W(T, P) + A(T, P)S + B(T, P)S^{3/2} + D(T, P)S^2 \tag{2.4}$$

$$\begin{aligned}
C_W(T, P) = \quad & C_{00} + C_{01}T + C_{02}T^2 + C_{03}T^3 + C_{04}T^4 + C_{05}T^5 \\
& + (C_{10} + C_{11}T + C_{12}T^2 + C_{13}T^3 + C_{14}T^4)P \\
& + (C_{20} + C_{21}T + C_{22}T^2 + C_{23}T^3 + C_{24}T^4)P^2 \\
& + (C_{30} + C_{31}T + C_{32}T^2)P^3,
\end{aligned}$$

$$\begin{aligned}
A(T, P) = \quad & A_{00} + A_{01}T + A_{02}T^2 + A_{03}T^3 + A_{04}T^4 \\
& + (A_{10} + A_{11}T + A_{12}T^2 + A_{13}T^3 + A_{14}T^4)P \\
& + (A_{20} + A_{21}T + A_{22}T^2 + A_{23}T^3)P^2 \\
& + (A_{30} + A_{31}T + A_{32}T^2)P^3,
\end{aligned} \tag{2.5}$$

$$B(T, P) = \quad B_{00} + B_{01}T + (B_{10} + B_{11}T)P$$

$$D(T, P) = \quad D_{00} + D_{10}P$$

[12] Chen and Millero, "Speed of sound in seawater at high pressures".
[13] Fofonoff and Millard, *Algorithms for computation of fundamental properties of seawater*.

Table 2.1: Numerical coefficients for the Chen and Millero equation for the speed of sound in seawater

Coefficient	Value		Coefficient	Value	
C_{00}	1402.388		A_{02}	7.164	$\times 10^{-5}$
C_{01}	5.03711		A_{03}	2.006	$\times 10^{-6}$
C_{02}	-5.80852	$\times 10^{-2}$	A_{04}	-3.21	$\times 10^{-8}$
C_{03}	3.3420	$\times 10^{-4}$	A_{10}	9.4742	$\times 10^{-5}$
C_{04}	-1.47800	$\times 10^{-6}$	A_{11}	-1.2580	$\times 10^{-5}$
C_{05}	3.1464	$\times 10^{-9}$	A_{12}	-6.4885	$\times 10^{-8}$
C_{10}	0.153563		A_{13}	1.0507	$\times 10^{-8}$
C_{11}	6.8982	$\times 10^{-4}$	A_{14}	-2.0122	$\times 10^{-10}$
C_{12}	-8.1788	$\times 10^{-6}$	A_{20}	-3.9064	$\times 10^{-7}$
C_{13}	1.3621	$\times 10^{-7}$	A_{21}	9.1041	$\times 10^{-9}$
C_{14}	-6.1185	$\times 10^{-10}$	A_{22}	-1.6002	$\times 10^{-10}$
C_{20}	3.1260	$\times 10^{-5}$	A_{23}	7.988	$\times 10^{-12}$
C_{21}	-1.7107	$\times 10^{-6}$	A_{30}	1.100	$\times 10^{-10}$
C_{22}	2.5974	$\times 10^{-8}$	A_{31}	6.649	$\times 10^{-12}$
C_{23}	-2.5335	$\times 10^{-10}$	A_{32}	-3.389	$\times 10^{-13}$
C_{24}	1.0405	$\times 10^{-12}$	B_{00}	-1.922	$\times 10^{-2}$
C_{30}	-9.7729	$\times 10^{-9}$	B_{01}	-4.42	$\times 10^{-5}$
C_{31}	3.8504	$\times 10^{-10}$	B_{10}	7.3637	$\times 10^{-5}$
C_{32}	-2.3643	$\times 10^{-12}$	B_{11}	1.7945	$\times 10^{-7}$
A_{00}	1.389		D_{00}	1.727	$\times 10^{-3}$
A_{01}	-1.262	$\times 10^{-2}$	D_{10}	-7.9836	$\times 10^{-6}$

One key characteristic of the equations presented thus far is that they relied on three environmental parameters to produce a value for the speed of sound: the temperature T, the salinity S and the pressure P. In 1981, Mackenzie set out to propose an equation that yielded viable results for this same problem by using depth instead of pressure. In his argument, he remarked how temperature and salinity were themselves measured as functions of depth, and how the previous empirical equations were long enough to be effectively incompatible with pocket calculators. While Medwin's simplified version of Del Grosso's equation – one of a small set that actually used depth instead of pressure at the time – was limited to depths of less than 1000 m, Mackenzie's expression was shown to have good agreement with pressure-based expressions down to 8000 m depths, and was likewise brief enough to be "satisfactory for general applications and is particularly convenient when employing small programmable calculators".[14] Mackenzie's formulation is given in Eq. 2.6.

$$\begin{aligned} c = {} & 1448.96 + 4.591T - 5.304 \times 10^{-2}T^2 + 2.374 \times 10^{-4}T^3 \\ & + 1.340(S-35) + 1.630 \times 10^{-2}D + 1.675 \times 10^{-7}D^2 \\ & - 1.025 \times 10^{-2}T(S-35) - 7.139 \times 10^{-13}TD^3 \end{aligned} \tag{2.6}$$

The equations presented in this section are considered relevant even today, in the ocean sensing domain. Indeed, Eq. 2.4 and 2.6 are of particular interest to this work, as they directly informed the sound speed measurements herein reported. The equations given by Wilson, Del Grosso, and Chen & Millero are made readily available to users of the SEASOFT software package provided by Sea-Bird Electronics as part of the interface to the company's Conductivity-Temperature-Depth (CTD) probes. In the case of R/V *Macrura*, this becomes relevant because the onboard CTD is a Sea-Bird Electronics SBE 37-SI. However, the Laboratory for Autonomous Marine Sensing Systems does not rely on SEASOFT, but rather uses a serial interface to communicate directly with the CTD onboard an appropriately equipped autonomous underwater vehicle. Per Sea-Bird's specifications, the sound

[14] Mackenzie, "Nine-term equation for sound speed in the oceans".

speed reported directly from the CTD via serial interface is computed using the Chen and Millero formula.[15] As a counterpoint, samples collected with other CTD probes external to the vehicle were processed with the Mackenzie equation, given the convenience of a depth-based expression.

This section has, by now, presented a brief history of the equations used to compute sound speed values from oceanographic measurements. A key takeaway from the material presented here is that the parametrization of these equations is non-trivial and empirical, rather than deterministic. Indeed the motivation behind providing the equations and related parameter tables is to lay the ground work for a discussion on the need for dimensional reduction of the environmental data in a deployment context, which was alluded to in Chapter 1 and will be presented in depth later in the text.

2.2 Acoustic propagation

As explained in Computational Ocean Acoustics[16] (COA), there are essentially five types of models used for acoustic propagation in the sea. These models are shown in Fig. 2.1, which is a reproduction of an equivalent figure in the aforementioned text (COA, Fig. 1.31). These models differ in the level of fidelity with which they reproduce the acoustic field; a distinction that is also largely coupled with the computational cost of each method – at higher fidelity, higher the computational demand. Additionally, the complexity of the field is also related to the signal frequency. Thus it stands that, for higher frequencies, ray methods are often the most practical, as they produce a physically intuitive and useful solution at relatively low cost, even if said result is colored by some well-known numerical artifacts. In contrast, the higher fidelity of other methods becomes increasingly valuable when computing the acoustic field at lower frequencies, such that the improvements in the model accuracy justify a manageable shift in cost.

[15] Sea-Bird Electronics, *Application Note 6: Determination of Sound Velocity from CTD Data*; Sea-Bird Electronics, *Seasoft V2: SBE Data Processing (Software Manual)*; Sea-Bird Electronics, *SBE 37-SI MicroCAT C-T (P) Recorder (User Manual)*.
[16] Jensen, Kuperman, Porter, and Schmidt, *Computational ocean acoustics*.

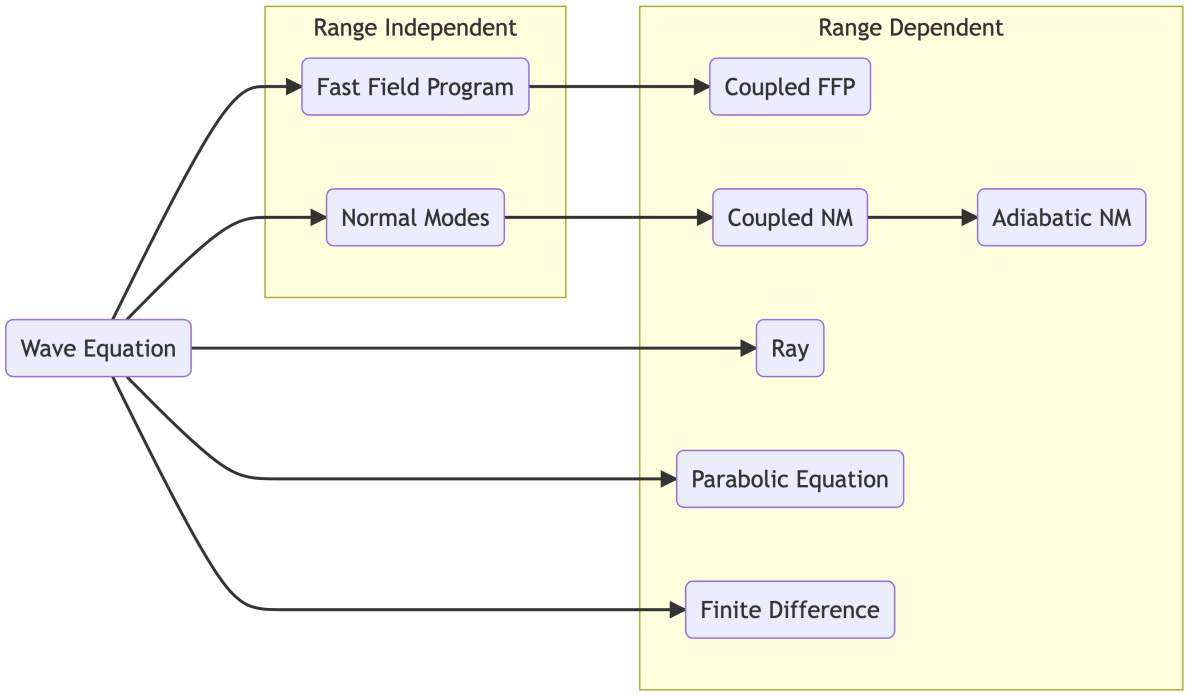


Figure 2.1: Models for acoustic propagation in the ocean.

In ocean acoustics, signals are generally classified as high frequency when they are around a few kilohertz or above, and low frequency signals are typically understood to be around or below one kilohertz. This research is concerned with an acoustic communication system operating at a carrier frequency of 10kHz, which falls within the higher frequency domain for marine environments. Furthermore, this work is interested in real-time AUV operations, where the computational capacity onboard the vehicle is limited and efficient operation of all software tools is of the essence for a successful field deployment. Given the functional requirements of the project, the rest of this work will be based on ray-based acoustic propagation models.

2.2.1 Ray methods

Ray methods are rooted in the Law of Refraction, also known as Snell's Law. Anyone who has spent enough time playing with a magnifying glass, trying to focus the sunlight to a pinpoint, likely has an

intuitive understanding of the concept of refraction. Arguably, this anchoring on physical intuition may be the greatest asset of ray-based methods: the insight we can draw from this sort of intuition can help us interpret and evaluate the results of more complex models. Thus, a brief introduction of the relevant theory is given hereafter; interested readers are encouraged to visit Chapter 8-C of David Blackstock's textbook[17] for a more detailed presentation.

$$\frac{\cos(\theta_1)}{c_1} = \frac{\cos(\theta_2)}{c_2} \tag{2.7}$$

In the context of acoustics, Snell's Law is typically given as in Eq. 2.7, expressed in terms of the grazing angle $\theta_\bullet$ and the corresponding sound speed $c_\bullet$. Using this mathematical formulation as a basis, the aforementioned physical intuition can be summarized in the following way: in a layered environment such as shown in Fig. 2.2, a ray traveling from one layer to the next will bend such that, when it enters a region with a faster sound speed, the grazing angle will be proportionally reduced; conversely, when the ray enters a region with a slower sound speed, the grazing angle will increase proportionally.

Ray bending alone isn't sufficient for a viable propagation model, as boundary conditions still need to be considered. Some numerical experimentation with Snell's Law may reveal to an interested reader that a physical limit exists for the equation's ability to capture a ray's behavior. Given a set of initial conditions θ_1 and c_1, a critical sound speed value $c_{2,crit}$ exists for the limit $\theta_2 = 0$, where $\cos(\theta_2) = 1$. Of course, transmission to layers with sound speed values greater than $c_{2,crit}$ does occur, but this falls into the domain of evanescent waves where the wavenumber k is complex; as this work is concerned with propagating waves (k is real), evanescent waves and related theory will not be addressed in further detail[18]. An alternative framing of this constraint is that, given two layers with sound speed c_1 and c_2 such that $c_1 < c_2$, there is a critical incoming grazing angle $\theta_{1,crit}$

[17] Blackstock, *Fundamentals of physical acoustics*.

[18] Readers interested in learning more about evanescent waves are encouraged to read the related sections in (Jensen, Kuperman, Porter, and Schmidt, *Computational ocean acoustics*) for a detailed presentation.

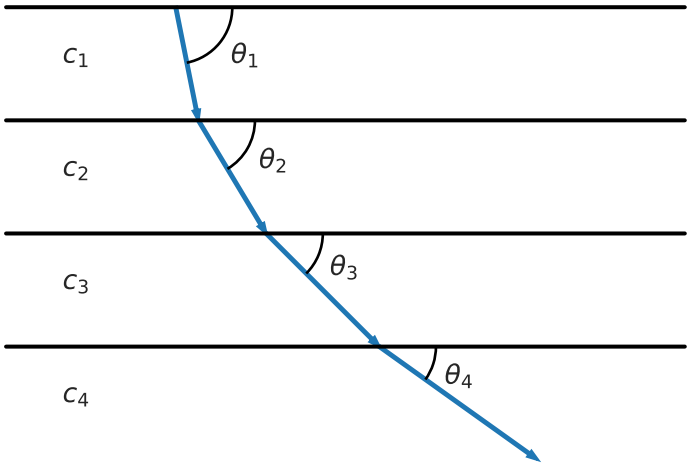


Figure 2.2: A ray bending through a stack of layers, each with an increasing sound speed $c_i < c_{i+1}$.

such that the outgoing ray will have a zero-valued grazing angle θ_2 and the wave will still belong to the propagating domain.

2.2.2 The Eikonal equation

When visualizing rays in an acoustic field, it is important to recall that these rays are, in essence, defined as the normal vectors to a wavefront. Thus, the next step in framing the acoustic propagation model in terms of ray methods is to describe the wavefront itself.

The derivation of a mathematical expression for the wavefront begins with the frequency-domain wave equation, or *Helmholtz equation*, which can be written in Cartesian coordinates as shown in Eq. 2.8. Here, the term $c(\mathbf{x})$ represents the sound speed as a function of position $\mathbf{x}$, and ω is the angular frequency.

$$\nabla^2 p + \frac{\omega^2}{c^2(\mathbf{x})} p = -\delta(\mathbf{x} - \mathbf{x}_0) \tag{2.8}$$

By seeking a solution to the above expression in the special form known as the *ray series* (Eq. 2.9), the Helmholtz equation can be rearranged into three equations, equated by the order with respect to ω. These three equations are known as the *Eikonal* equation, for the terms related to ω^2; and the *transport* equations, related to ω and ω^{1-j}. The particulars of the derivation are covered in great detail in (Jensen, Kuperman, Porter, and Schmidt, *Computational ocean acoustics*), so they will not be reproduced here.

$$p(\mathbf{x}) = e^{i\omega\tau(\mathbf{x})} \sum_{j=0}^{\infty} \frac{A_j(\mathbf{x})}{(i\omega)^j} \tag{2.9}$$

Of the three expressions derived from the Helmholtz equation, this section is mainly concerned with the first one. As shown in Eq. 2.10, the Eikonal equation relates the travel time τ through a medium with the velocity $c(\mathbf{x})$ of the medium. In other words, the Eikonal equation means that the gradient of the arrival time surface is inversely proportional to the speed of the wavefront.

$$\|\nabla\tau\|^2 = \frac{1}{c^2(\mathbf{x})} \tag{2.10}$$

Physical intuition for the effect captured in the Eikonal equation starts with a simple, uniform domain – that is, one where the sound speed is equal all throughout. Given the uniform domain, it stands that $\|\nabla\tau\|^2$ will likewise be a constant. Thus, this simple environment, the time fronts will be circular and centered on the source location, while the gradient will be composed by the radial projections starting at the source. A convenient characteristic of this simplified scenario is that all solutions of the Eikonal equation are unique.

The uniform domain is a convenient starting point to build some intuition, but it is also rather uninteresting on its own. If we then change the sound speed from a constant value to a linear profile, the time field changes to the one shown in Fig. 2.3. Here, the gradient field (represented by radial projections in the uniform environment) is now represented by curved paths shown in blue; the

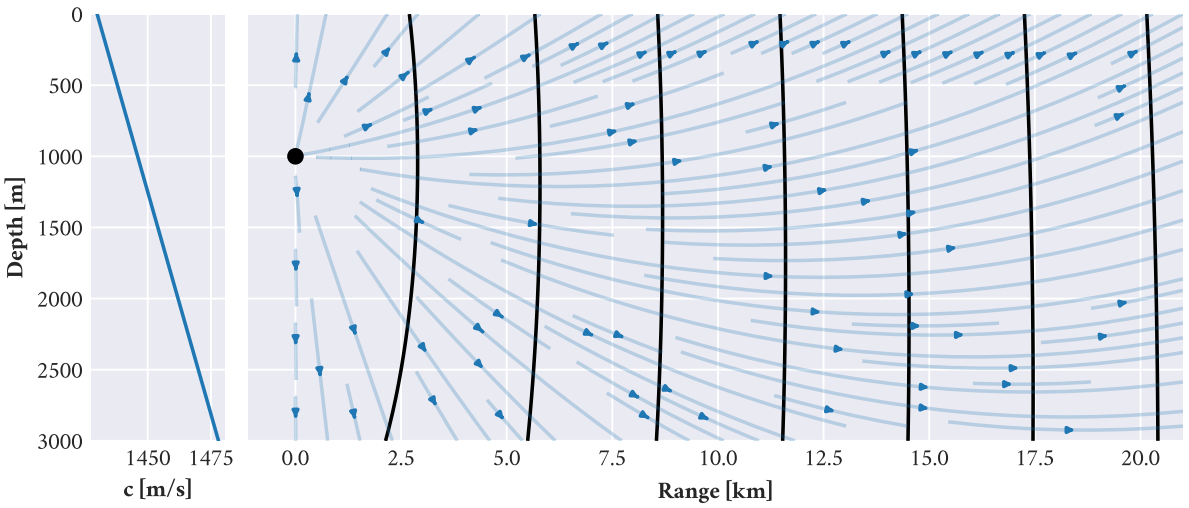


Figure 2.3: Equal time fronts (black) and time field gradient (blue/arrows) for a linear sound speed profile and a source located at 1000m depth.

iso-temporal fronts which used to be circles in the base case are now transformed, as sound travels through the lower depths travel faster than it does in the surface layers.

This simplified scenario does not yet account for boundary interactions, nor does it illustrate how the non-linearity of the Eikonal equation may lead to multiple valid solutions for more complex sound speed profiles. These complicating factors are discussed next.

2.2.3 Multipath propagation

One key issue with the Eikonal equation is that its non-linearity may lead to multiple viable solutions, which can cause a number of complications for a user seeking to exploit the kind of information therein represented. Additionally, boundary interactions can also add to the number of viable solutions. Thus, within the field of ocean acoustics, the Eikonal equation is generally solved by the *method of characteristics*. In simplified terms, this means rewriting Eq. 2.10 with respect to a series of curves that are perpendicular to the time fronts τ_i; these curves (rays) serve as a new coordinate system, such that the otherwise overlapping (non-unique) solutions in the Cartesian representation

become unique relative to *ray coordinates*. Expressed in cylindrical coordinates, this yields Eq. 2.11. These rays, defined as being perpendicular to the time fronts τ_i, therefore match the gradient of the time field which is shown in blue in Figure 2.3.

$$\begin{aligned} \frac{\mathrm{d}r}{\mathrm{d}s} &= c\xi(s), \qquad & \frac{\mathrm{d}\xi}{\mathrm{d}s} &= -\frac{1}{c^2}\frac{\partial c}{\partial r} \\ \frac{\mathrm{d}z}{\mathrm{d}s} &= c\zeta(s), \qquad & \frac{\mathrm{d}\zeta}{\mathrm{d}s} &= -\frac{1}{c^2}\frac{\partial c}{\partial z} \end{aligned} \tag{2.11}$$

To understand the method of characteristics, it helps to once again begin with a simplified environment. Figure 2.4 illustrates a simple example where two viable solutions are shown for a uniform half-space with sound speed c, and a source-receiver pair located at $\mathbf{x}_s$ and $\mathbf{x}_r$ respectively. These two solutions, shown as blue and amber, are the product of boundary interactions and are effectively indistinguishable from one another in the Cartesian space representation. That is, if we were given the instruction to select a value τ for the receiver coordinates $\mathbf{x}_r = [x_r, y_r, z_r]$, we would not have sufficient information to determine which of the two values $\tau(\mathbf{x}_r)_{,1} \neq \tau(\mathbf{x}_r)_{,2}$ to choose (left side of the figure). In ray coordinates, where location $\mathbf{x}$ is of the form $\mathbf{x}_i = [\theta_{0,i}, s_i]$, the solutions become unique with respect to the ray's coordinates. These correspond to $\theta_{0,i}$, the launch angle of the i-th ray; and s_i, some strictly increasing coordinate value (such as distance traveled, or time) along the path of the i-th ray. The right side of Figure 2.4 illustrates the increasing travel time τ along each of the rays as s increases up to the point $s_{i,r}$ at which each ray reaches the receiver coordinates. As this example illustrates a uniform half-space of sound speed c, the values τ_i increase by the simple algebraic rule $\tau_i = s_i/c$.

In switching from the Cartesian representation to ray coordinates, we avoid the issue of having multiple solutions as a product of the nonlinearities in the Eikonal equation – the solutions become unique in ray coordinates. However, the process of collecting results for a given receiver location becomes somewhat more complex because we still have to account for how ray coordinates map

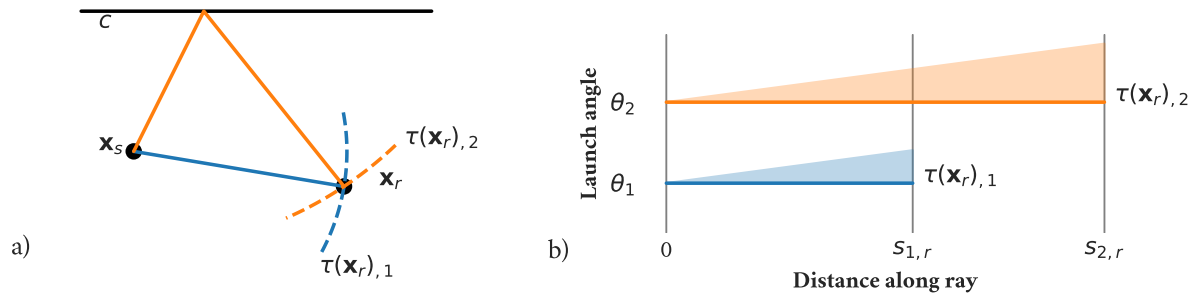


Figure 2.4: Two rays illustrate distinct viable solutions for the Eikonal equation in a simple uniform half-space.

into Cartesian space. The sample rays shown in Figure 2.4 are actually members of a special subset of rays known as *eigenrays*, which connect two points in space. It may be apparent to an attentive reader[19] that a random choice of launch angle does not guarantee the existence of some value $s_{i,r}$ where the ray would reach the receiver. Thus, a methodology is required to (i) identify these special rays that connect a source-receiver pair based on their launch angle $\theta_{0,i}$, and then (ii) report the associated viable solutions τ_i.

The ray coordinates can be described as a manifold representation of Cartesian space; just like we consider the travel time τ in Figure 2.4 (right), we can likewise express a set of functions $r(\theta_{0,i}, s_i)$ and $z(\theta_{0,i}, s_i)$ (in the case of a 2D range-depth domain) that map ray coordinates to their Cartesian equivalent. Assuming a sufficiently fine-grained fan of launch angles and likewise fine enough sampling along the ray distance s, then, one possible numerical method to identify eigenrays that reach a receiver located at $[x_r, y_r, z_r]$ (3D) or $[r_r, z_r]$ (2D) would be to use a capture radius from the receiver position, to detect rays that enter the capture space. This sort of algorithm is well established in literature and is also used in standard software tools for the field of ocean acoustics.

[19] Or a reader who has played enough billiards and has realized the parallels of the game vis-à-vis this uniform half-space scenario

2.3 Acoustic tomography and geophysical inversion

The field of acoustic tomography as we know it today originated around 1975, from a collaboration between Walter Munk, Carl Wunsch and Peter Worcester. At the time, Munk was working at Scripps Institution of Oceanography; Carl Wunsch was at MIT, in what is nowadays the Department of Earth, Atmospheric and Planetary Sciences. Peter Worcester was a student under Munk's guidance, working on long range (25km) reciprocal acoustic transmission experiments. In Wunsch's own words, talking about those days:[20]

> *After a few days, the Jason director, Dick Garwin, wandered in to ask what we were doing; when we told him, he said, "You've just reinvented [medical] tomography." The first written account of the technology was by the three of us in an unclassified Jason technical report. Walter and I went on to make it practical, based upon collaboration with Worcester and numerous colleagues from acoustics, engineering, and oceanography. In a later book (Munk et al. 1995), we attempted a summary of this work.*
>
> – Carl Wunsch

At its heart, Ocean Acoustic Tomography (OAT) aims to create estimates of the sound speed structure in a volume of water – much in the same way that medical tomography seeks to map out different material properties in the human body. To achieve this, both fields rely on a series of measurements collected from source and receiver arrays configured in such a way as to produce overlapping paths in the intermediate space. These measurements, recorded in the receiver projection space[21], are then matched with an initial (*a priori*) model and processed as inputs for the inversion problem to produce an estimate of the volumetric distribution of their respective target parameters.[22] In

[20] Wunsch, "Right Place, Right Time: An Informal Memoir".

[21] A more familiar way to think of this may be to think of the projection space as "camera angles", from which a scene is being recorded.

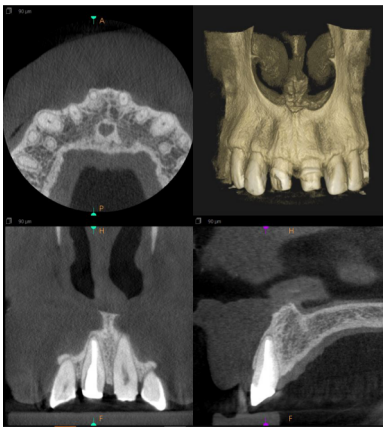
[22] Deffenbaugh, "Optimal Ocean Acoustic Tomography and Navigation with Moving Sources"; Duda, "Modeling and Forecasting Ocean Acoustic Conditions".

terms of the practical value of OAT, it is worth noting that while Satellite Oceanography gets us a surface-layer view of what is happening, we often still need information about depth-dependent variability.

To put things in perspective, Figure 2.5 (a) illustrates a use case of tomography in dental applications, where the volumetric information devised from non-invasive measurements is often used as a reference model to plan surgical procedures. The dental or medical imagery may feel more familiar or intuitive, as it is more closely connected to the visible aspects of our daily lives; but the practical value of volumetric reconstruction carries over just as well to our understanding of the ocean environment and its impact on climatological phenomena. Where medical tomography generally benefits from having a relatively small target volume, however, OAT generally faces a challenge of scale; to produce the comprehensive results sought of tomographic solutions, OAT requires arrays that span tens to thousands of kilometers – to this effect, Figure 2.5 (b) shows two OAT experiments deployed in the North Atlantic between 1988-1992, along with a 2D snapshot of the sound speed at 300m depth.

Another analogy between the medical and ocean acoustics fields may be apparent at this point. Much like the medical field still finds value in resources such as 2D X-ray images, which are generally more accessible than tomograms, the domain of ocean acoustics also draws notable value from less comprehensive techniques. Ocean Climate Thermometry (OCT) is one such example, where the system may be reduced, in the extreme, to recording data for a source-receiver configuration covering a single geodesic path in order to produce a path-averaged estimate of the sound speed. This value of sound speed is, in turn, closely related to the path-averaged temperature. Indeed, we can compare OCT with single-point temperature measurements, which are subject to highly localized variability related to turbulence and internal waves; OCT can provide a relatively cost-effective view into the average temperature of the ocean basins. Although it is much more spatially limited when compared to OAT, even the simplest form of OCT can become highly valuable when looking

a)



b)

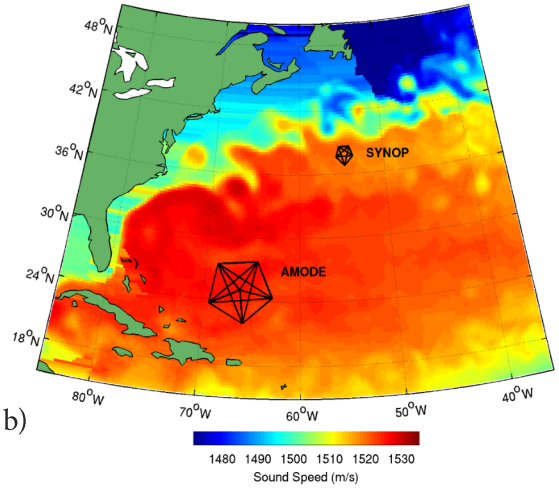


Figure 2.5: A comparison of medical and ocean acoustic tomography. a) A sample volumetric model produced by Cone-beam Computed Tomography (CBCT), which is often used in medical applications to plan surgical procedures. b) Locations of two experiments employing ocean acoustic tomography in the North Atlantic, overlaid on a snapshot of sound speed at 300m depth derived from a high-resolution numerical ocean model. These volumetric reconstructions enable users to explore distinct cross-sections of a target volume, compared to non-separable 2D projections. Image credits: a) Jose D. Viquez, DDS, FRCD(C); b) Brian Dushaw, Ph.D. (released to public domain).

at changes of the average temperature estimate of a large body of water over an extended period of time.[23]

2.3.1 The linearized inverse problem

In order to address the inversion problem, let us first consider the forward problem as stated in Equation 2.12, where x represents perturbations in the sound speed profile and y represents the measured variation in ray travel times. The measurement matrix C captures the modeled transformation from sound speed perturbations x to resulting travel time variability y. In addition to this linear relation, there are two sources of error introduced with the noise vector n: true measurement noise as well as mismatch between the true perturbations and the linear model.

$$y = Cx + n \tag{2.12}$$

The measurement matrix C captures the relation between deviations from a base model. Its entries can be populated by accounting for Fermat's principle, which can be loosely stated as the intuitive idea that neighboring paths will have a very similar travel time, with variations captured in second-order terms. In other words, Fermat's principle states that the time integral will be independent of small perturbations in the integral path – so, the travel time calculation may be performed as if the path was unchanged, and following the derivation given by Deffenbaugh ("Optimal Ocean Acoustic Tomography and Navigation with Moving Sources") yields the relation given in Equation 2.13.

$$C_{in} = \frac{k_i(n\Delta z)}{\beta} \frac{\sqrt{1 - \frac{c^2}{c_t^2}}}{c} \Bigg|_{\min(ct, c((n-1)\Delta z))}^{\min(ct, c(n\delta z))} \tag{2.13}$$

In Eq. 2.13, the term $k_i(z)$ represents the number of times the i-th ray passes through depth z, where $z = n\Delta z$ in the discretized formulation. β is the rate of change of the sound speed profile

[23] Duda, "Modeling and Forecasting Ocean Acoustic Conditions".

with respect to depth, $\beta = \frac{c(n\Delta z) - c((n-1)\Delta z)}{\Delta z}$. The sound speed term c_t is defined as the sound speed for which the ray would become horizontal.

Where Eq. 2.12 aims to convert sound speed perturbations into predicted travel time variation, capturing the forward problem, the objective of the tomography problem is instead to relate measured travel time variations to a best estimate of the sound speed perturbations that caused them. The tomography problem is, therefore, the inversion of Eq. 2.12 to solve for the sound speed perturbations x.

As mentioned earlier, Ocean Acoustic Tomography generally targets large volumetric scales in the order of many kilometers, which can be compared to the relatively small targets in medical applications. This difference in scale translates to an increased operational cost incurred for each unique sampling path – each new eigenray, connecting sources and receivers in the experimental setup. Accounting for this constraint, OAT nonetheless seeks to solve for the volumetric model of the sound speed profile using a limited number of samples, and the inversion problem for OAT is therefore significantly under-determined.

Generally speaking, the most common techniques used to address the under-determined nature of the tomography problem can be described as order-reduction approaches. The idea behind this type of formulation is to exploit model assumptions, or prior data, to rewrite the problem into a form that couples the numerous dimensions of variability into the most typical or most significant modes. This can be achieved with Empirical Orthogonal Functions (EOFs), which are generally comparable to the components produced by Principal Component Analysis (PCA). One notable difference between the two is that some EOF applications may introduce a smoothing step or some other minor alteration to the function set, after extracting the base components equivalent to PCA.

The conventional EOF or PCA approach consists of developing a covariance matrix from a set of samples – in the case of tomography, these samples would be sound speed profiles. The dominant eigenvectors of the resulting covariance matrix are then taken as the basis vectors for the reduced-

order representation. As such, these methods generally provide a means of lossy data compression, and are well suited to fit measurements in the same domain as the model. In the context of tomography, where sampling y occurs in a different domain than the model representation x, the Optimal Orthogonal Function (OOF) method produces the set of basis functions ϕ that represent the greatest possible reduction of error in the inversion problem by accounting for the observability of the variations in the sound speed profile.[24] Other less direct approaches, such as dictionary learning, have also been explored.[25]

2.4 Positioning

The classical tomography approach – be it medical or OAT – generally expects that the position of sources and receivers are known. This is because those positions are typically used as static reference points, such that the inversion problem can focus on seeking the convergence of the volumetric model alone. Doing otherwise means adding complexity to the inversion problem, making it more challenging to solve and making the output much less reliable. Similarly, much of navigation first begins with determining one's position in order to inform path planning from that point forward – positioning precedes navigation.

In very general terms, positioning systems can generally be described by two fundamental models, known as spherical and hyperbolic positioning. The key difference between these two systems lies in how they handle timing information. Thus, it stands that timing precedes positioning much like positioning precedes navigation.

Spherical positioning requires knowledge about timing in a system-wide sense – it needs to know when signals are transmitted and when they are received. This is because spherical positioning relies on a simple ranging model, which can be directly related to the signal's travel time. Equation 2.14

[24] Deffenbaugh, "Optimal Ocean Acoustic Tomography and Navigation with Moving Sources".
[25] Bianco and Gerstoft, "Dictionary learning of sound speed profiles".

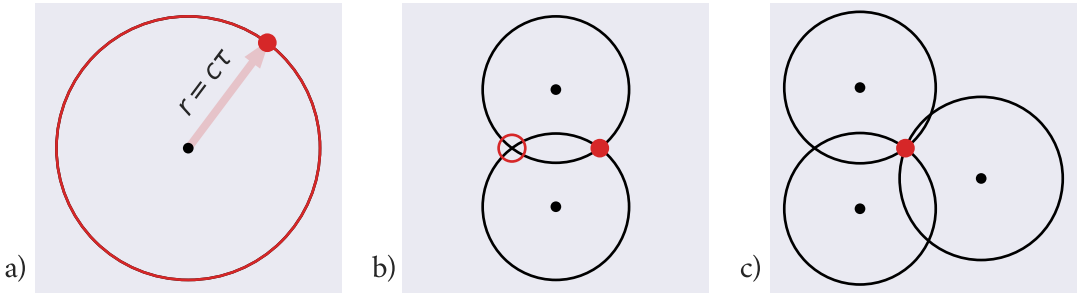


Figure 2.6: Spherical positioning uses travel time information directly, and thus requires knowledge about the system's synchronization. a) Single-beacon range shown as red circle; simple ranging is possible due to available timing information, but distinct positions cannot be isolated. b) Two-beacon position, with red circles marking true position (filled) and ambiguous solution (unfilled). c) 3-beacon solution shows how additional measurements help break the ambiguity and reduce uncertainty about the true solution.

relates the range r between a source and receiver to the travel time τ by way of the signal propagation speed c.

$$r = c\tau \tag{2.14}$$

Using this simple relation, a vehicle's position can be estimated relative to the positions of a set of beacons. The solution, illustrated in Figure 2.6, can be described as minimizing the error between the ranges measured per Eq. 2.14 and the ranges derived from the known beacon positions and the estimated vehicle position. Thus, a single beacon will at best produce a range from itself; two receivers would produce two distinct solutions, but would not be able to break the ambiguity. A system with 3 or more beacons, however, would be able to collect enough information to break the ambiguity and give a unique solution.

Hyperbolic positioning treats time information differently, in that it uses *differences* in travel time, and can therefore remain oblivious to the details about the time of transmission. By rewriting the travel time as the difference between transmission and reception times ($\tau = t_{\mathrm{Rx}} - t_{\mathrm{Tx}}$), we can rearrange Eq. 2.14 to illustrate this point. The relation between the range difference and the travel

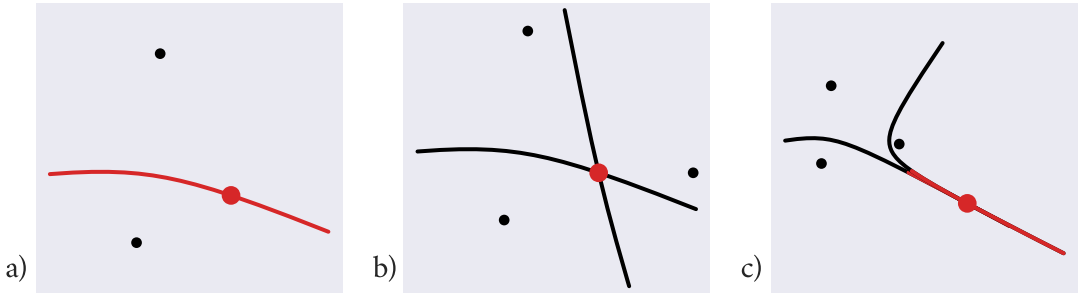


Figure 2.7: Hyperbolic positioning uses *differences* in travel times, and does not require knowledge about the system's synchronization. a) A minimum of two beacons are needed to produce a time difference and a single hyperbola. b) Three beacons can produce two hyperbolas, using one node as a reference point for the time difference. c) In the 3-beacon system, the position estimate becomes underdetermined along the baseline extensions, where the hyperbolas become tangential (red line).

time difference allows us to translate time measurements into hyperbolas – lines that preserve the range differences – which then form the basis upon which this flavor of positioning operates; these concepts are illustrated in Figure 2.7.

$$\Delta r = c\Delta\tau = c\left[(t_{\mathrm{Rx},1} - \cancel{t_{\mathrm{Tx}}}) - (t_{\mathrm{Rx},2} - \cancel{t_{\mathrm{Tx}}})\right] \tag{2.15}$$

These two models, spherical and hyperbolic, can be regarded as two extremes of the timing information domain; the first implies that the system-wide time information is known exactly (zero variance), while the latter represents the scenario where the timing variance approaches infinity. These extremes also have well-defined performance thresholds, in terms of their ability to estimate position from noisy measurements, given by the respective Cramér-Rao bounds. Between these two extremes lie the models for which the timing information is known with finite but non-zero variance; for example, when one node's clock tends to run faster than the rest.[26]

Real world applications of the above models have been well documented in literature over the years; at least one of them is most likely quite familiar to you, the reader. Nonetheless, a brief

[26] Deffenbaugh, Bellingham, and Schmidt, "The relationship between spherical and hyperbolic positioning"; Deffenbaugh, "Optimal Ocean Acoustic Tomography and Navigation with Moving Sources".

presentation of some such examples follows, in recognition of their relevance to this work. A more extensive, if not completely thorough, presentation of these examples was recently published[27] and is recommended to interested readers.

2.4.1 The Global Navigation Satellite System (GNSS)

Although we rarely make the distinction in casual conversation, there is a notable difference between the Global Positioning System (GPS) and a generic Global Navigation Satellite System (GNSS). The former actually refers to a specific GNSS, which is owned by the US government, and is operated and maintained by the US Air Force. The US-based GPS is one of four such satellite networks, also referred to as constellations, in orbit: the Russian GLObal NAvigation Satellite System (GLONASS) was deployed around the same time as GPS; the Chinese BeiDou system reached global coverage with its 3rd generation, whose last satellite was launched in 2020; and Galileo, the European system, has been operational since 2019. Each of these systems generally consists of 20-35 satellites; the United States, for example, is committed to maintaining the availability of at least 24 operational GPS satellites, 95% of the time. As of this writing, the US government reports 31 operational units for GPS, across modern and legacy satellites[28]; the European system reports 22 units available for use on the Galileo constellation[29].

The various GNSS systems provide timing, positioning and navigation services, which are essential to the scientific community; and which are of significant value to the general public as well. To do so, the satellites themselves carry high-precision atomic clocks; but the heart of GNSS is actually land-based. Tracking stations on Earth provide timing updates to GPS satellites twice a day, for example, and also relay satellite position information. The satellites, in turn, are constantly broadcasting their position and timing information. On the user side (client-side), once enough signals

[27] Van Uffelen, "Global Positioning Systems: Over Land and Under Sea".
[28] https://www.gps.gov/systems/gps/space/
[29] https://www.gsc-europa.eu/system-service-status/constellation-information

are detected and tracked by the GPS receiver, the sensor can determine its own position by using Eq. 2.14, with c set to the speed of light.

This is a slight simplification, of course, but one that is good enough in the scope of this brief discussion. From Eq. 2.14, and given the large value of the speed of light (299,792,458 meters per second), it should be apparent that even slight synchronization errors can lead to significant range errors in GNSS applications. Additionally, ionospheric variability and other environmental effects can change the true value of the propagation speed c and introduce additional error to the system. To illustrate this, let's again consider Figure 2.6, which shows the 3 beacon ranges on the 2D scenario perfectly overlapping at the receiver location (the 3D problem handled by GNSS requires a minimum of 4 satellites to break the ambiguity). A mismatched propagation model would offset those range rings, making them larger or smaller than they are in this figure. Say, for example, that the true propagation speed was slightly slower than modeled; then, the travel time measured will increase accordingly.

In this simplified case, the receiver knows nothing about the mismatch and thus it will keep using the model speed to project the expected ranges; the rings produced by the model would be larger than those shown in the figure. The true position of the vehicle would still be expected to be within the area where all circles intersect, but the optimization problem can no longer set the error to zero as in the purely theoretical scenario. In the field, these sources of error will also vary for each satellite-receiver path; they translate to uncertainty in the final output. As was mentioned before, the uncertainty of the model's output is well characterized by the Cramér-Rao bounds.

As may be expected, there are indeed applications where additional information about the model mismatch is accounted for, in order to reduce the uncertainty of the position estimator. One example of such a GNSS enhancement is the case of real-time kinematic positioning (RTK). The RTK paradigm uses a static reference station to determine the necessary corrections with respect to the

conventional GNSS solution, and to relay this additional information to nearby vehicles in order to improve the performance of onboard position tracking.

2.4.2 Acoustic positioning

There is one key difference between GNSS and acoustic positioning, in that the former relies on radio waves (the L1 carrier frequency used by satellites corresponds to 1575.42 MHz), while the latter relies on mechanical pressure waves. Beyond that, the underlying principles are generally the same as those described earlier, for spherical and hyperbolic positioning. In the context of underwater vehicles, we have the advantage that depth-sensing can resolve one of the dimensions, reducing the tracking problem to a 2D scenario. Furthermore, 2-way travel time systems or synchronized clocks may be used to track timing information, thus moving the estimator's performance towards that of the spherical positioning model.

Traditional acoustic positioning tends to break down into 3 categories, depending on the characteristic length of the tracking system. Long baseline (LBL) systems tend to have beacons spread far out, with their positions carefully surveyed or closely tracked with a GPS-enabled surface expression, and typically operate in the order of hundreds of meters to several kilometers. Short baseline (SBL) systems typically have beacons placed along the span of a surface ship, and tend to operate in a much smaller scale – precision tracking is generally only available in the vicinity of the surface ship. Ultra-short baseline (USBL) systems further compress the beacon span to a small transducer array, and generally use phase information detected by the array (which relates to the arrival angle using a technique known as beamforming), along with the arrival time, to produce a vehicle position estimate.

2.5 Autonomy

The concept of autonomous vehicles – along with the plethora of challenges these unmanned systems face – have become a part of the public conversation thanks to the growing presence of advanced driver assistance systems (ADAS) onboard commercially available cars and trucks. Although it may be somewhat less visible to the general public, the autonomy conversation has also permeated the shipping industry at large.

In 2018, Boston-based Sea Machines announced a contract with the Danish company A.P. Moller-Maersk – one of the largest container ship lines in the world. Per this contract, the situational awareness system developed by Sea Machines would be trialed onboard one of Maersk's container ships as a means of providing the ship operators with additional information that may help reduce human error. That same year, Rolls-Royce Commercial Marine[30] and the Finnish ferry operator Finferries demonstrated fully autonomous operations with the car ferry *Falco*, which operated in the Turku archipelago. A third example of autonomy in the shipping industry is given by MV *Yara Birkeland*, developed by chemical company Yara International and technology enterprise Kongsberg Maritime. MV *Yara Birkeland* was designed in 2017, and departed on its maiden voyage the 19th of November, 2021.

2.5.1 Middleware autonomy

At the heart of any autonomous vehicle used in industrial applications (as exemplified previously) or in research, there is some flavor of decision-making infrastructure that justifies the name. This infrastructure generally consist of a set of collaborative processes built on top of middleware architectures; sensor drivers, for example, can interface with mission manager apps to determine what actuator signals are needed to move a robot in the desired direction. Within the AUV domain, three

[30] Rolls-Royce Commercial Marine (RRCM) was acquired by the Norwegian technology group Kongsberg Gruppen. The acquisition was finalized on April of 2019, and RRCM officially became part of Kongsberg Maritime from that point forward.

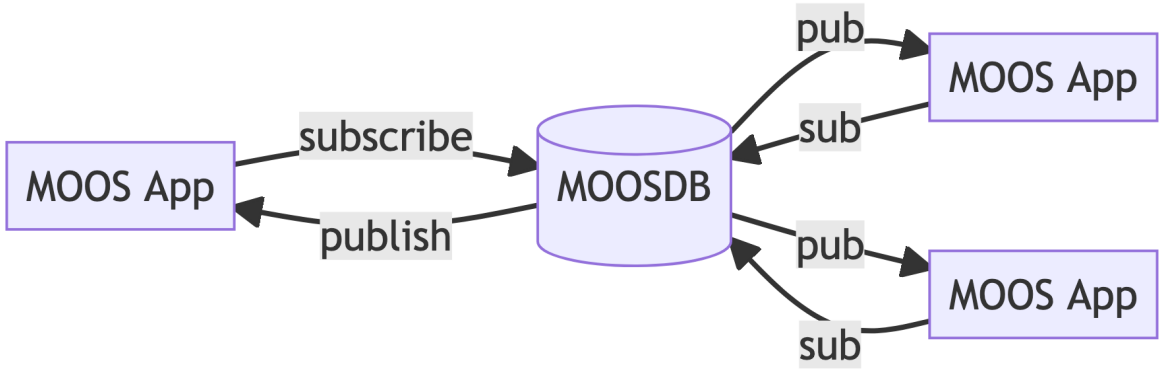


Figure 2.8: The MOOS autonomy middleware uses a centralized publish-subscribe architecture.

such middleware resources tend to stand out: the Mission Oriented Operating Suite (MOOS),[31] the Robot Operating System (ROS),[32] and the Lightweight Communications and Marshalling library (LCM).[33]

The core idea of sharing information between different processes and vehicles is a common one. However, the middleware options above differ in terms of their transport protocol and centralization. LCM, for example, uses the low latency but unreliable user datagram protocol (UDP) in a decentralized architecture. MOOS and the original ROS1 both use the transmission control protocol (TCP) in a centralized structure; the centralized publication-subscription scheme in MOOS is shown in Figure 2.8. ROS2 expanded on its predecessor by adopting the Data Distribution Service (DDS) specification as its base layer. This change enabled ROS to benefit from the decentralized discovery capabilities of DDS, as well as reliable message delivery even on the generally unreliable UDP. DDS is managed by the Object Management Group (OMG), an international technology standards consortium.

High-level middleware such as ROS and MOOS benefit primarily from their developer ecosystems, more so than from the specific details of their plumbing. Given its use for this research project,

[31] Newman, "MOOS-mission orientated operating suite".
[32] Quigley, Gerkey, Conley, Faust, Foote, Leibs, Berger, Wheeler, and Ng, "ROS: an open-source Robot Operating System".
[33] Huang, Olson, and Moore, "LCM: Lightweight Communications and Marshalling".

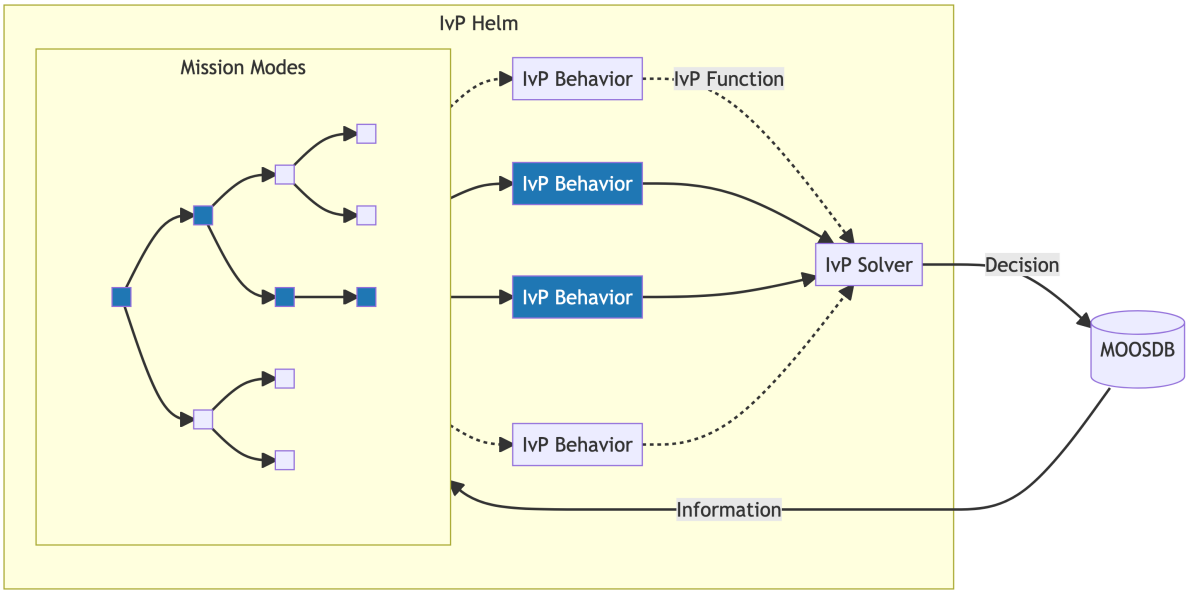


Figure 2.9: The MOOS-IvP autonomy software suite introduces the IvP Helm, which enables complex decision-making through multi-object optimization.

the remainder of this section will focus on MOOS in particular – and on one of the key resources in its ecosystem: the MOOS-IvP suite.

The MOOS-IvP suite builds on top of the MOOS middleware, adding a number of applications and utilities that bring significant value for autonomous vehicle operations. The IvP Helm, which lives at the heart of this expansion suite, stands out for its role in enabling complex decision making – IvP stands for Interval Programming, the multi-objective optimization method used by the helm. As shown in Figure 2.9, the IvP Helm uses a behavior-based architecture; its internal IvP solver interprets the needs of the different (potentially competing) behaviors into a unified decision that can then be relayed to the vehicle's actuator controllers.[34]

[34] Benjamin, Schmidt, Newman, and Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP".

2.5.2 Communications

One of the key limitations of AUV operations is the limited throughput available for underwater communications, which is generally in the order of 100 bits per second, and often less than that. Though the MOOS-IvP suite provides many valuable resources for the operation of Autonomous Underwater Vehicles, one layer that is not readily addressed in the suite is that of this acoustic communications interface.

In response to this need, the Goby Underwater Autonomy Project was initially developed with the intent to create a unified framework for collaboration between multiple marine vehicles, by seamlessly incorporating communication solutions across multiple interfaces: acoustic, ethernet, wifi, and serial. Earlier versions were designed to work with MOOS-IvP and the associated software developed at MIT LAMSS. Now in its third version, the project has become a full middleware in its own right; Goby3 provides a publish-subscribe solution built on the concept of nested communications, where different layers meet the needs for inter-thread, inter-process and inter-vehicle communications.[35] Although Goby3 is a middleware in its own right, the project has preserved compatibility with the MOOS-IvP suite, making it possible for users to continue to benefit from the extensive work captured in the IvP Helm, while also adding advanced capabilities in terms of communication.

In addition to providing access to any choice of transport mechanisms, such as the acoustic communications link, the Goby3 project also offers the flexibility to work with any number of data marshalling schemes such as Google Protocol Buffers (GPB) and JSON. Where throughput is of great concern for the inter-vehicle layer when using an acoustic communications link, the project offers support for the Dynamic Compact Control Language (DCCL), which provides a flexible yet powerful way to marshal data into very small datagrams, or packets. To understand the importance of efficient data packaging for underwater operations, it helps to compare the acoustic communi-

[35] Schneider, "Goby3: A new open-source middleware for nested communication on autonomous marine vehicles".

cations throughput with that of the link used to reach vehicles on Mars. Though latency may be higher, the typical throughput of the Mars to Earth link is generally in the order of 10^4 bits per second. Thus, the fact that DCCL compression offers a 50-80% improvement over pure GPB or a Python struct (packed binary data class) is quite valuable.[36]

2.6 Summary

This chapter has introduced key concepts across various domains: the principles of sound propagation, including how they are used to measure properties of the world's oceans; the core methods for positioning, and how they are embodied in real-world applications, both over land (GNSS) and below the sea (LBL, SBL, USBL); the basics of vehicle autonomy, along with some examples of well-established frameworks.

Additional background material that is relevant to a specific chapter will be presented therein. Such is the case with beamforming and time-to-intercept techniques, used in Chapter 3 for behavior classification; the aforementioned signal processing techniques are thus presented in Section 3.2.1.

[36] Schneider, Petillo, Schmidt, and Murphy, "The Dynamic Compact Control Language version 3".

3 A Machine Learning primer

"How do you know? It's a question we need to ask more often, both of ourselves and of others. The power lies in its frankness. It's nonjudgmental — a straightforward expression of doubt and curiosity that doesn't put people on the defensive."

— Adam M. Grant, *Think Again: The Power of Knowing What You Don't Know*

One of the essential features of the scientist's mindset, Grant goes on to explain in his book, is their innate curiosity. Sometimes, that inquisitiveness may be directed at a problem – "how can we solve this?" – but other times, it is (and should be!) directed at the proposed solution. This line of questioning is, after all, the one that allows us to explore new ways of handling the problem, and to recognize gaps or flaws in our prior views. Given new information, the scientist's approach is then to adjust his thinking accordingly.

This chapter opens with a brief history of Machine Learning (ML). With this historical foundation in place, the chapter then explores Grant's question in two ways, framed within the experiential learning process. The first angle comes from the perspective of an autonomous vehicle, and undergirds the entirety of this thesis: how can the vehicle evaluate information it has collected already, about its environment or about its collaborators, to inform its decision making process? The second angle belongs to you, the reader, as it did to me in conducting this work. For this second perspective, the original question transforms into two threads. First, how can we be confident we understand the information available to the vehicle? And then, do we truly understand the flow of such infor-

mation through the vehicle's decision pipeline? In short, this chapter introduces some of the key concepts and lower-level techniques of ML in the context of two experimental challenges. These concepts and methods constitute the building blocks for more advanced ML techniques which will be discussed later in the text.

From a technical standpoint, the core contributions of this chapter are (1) the implementation of a self-contained vehicle behavior classification algorithm that supplements the well-established bearing-based target tracking signal with a Time-To-Intercept (TTI) estimation algorithm similarly based on passive sensing, to increase the amount of information drawn from the acoustic environment for the classification task;[1] and (2) the implementation of a related but distinct classifier designed for an environment characterization application, this one related to external data sources rather than design choices to inform the construction of reference models.[2]

3.1 A brief history of Machine Learning

The terms *Artificial Intelligence* (AI) and *Machine Learning* (ML) are widely known at this point; without necessarily understanding the intricacies of the words, there is a reasonable expectation that these words may be known and even spoken in the average household. This may be a product of the near ubiquity of smartphones, the far-spread use of social networks or our modern-day dependence on the internet and particularly on search-engines. Whatever the causes that brought us here, concepts like voice and facial recognition, smart news feeds and predictive search suggestions are all a thing of daily life in much of the world.

When considering humankind's desire for what we now call AI, modern day experts in the field have even drawn a line back to Greek mythology. One of the stories highlighted was that in which

[1] Fischell, Viquez, and Schmidt, "Passive acoustic tracking for behavior mode classification between surface and underwater vehicles"; Fischell, Viquez, and Schmidt, "Machine learning for behavior classification of passively tracked vessels".

[2] Viquez, Fischell, and Schmidt, "Estimation of the acoustic environment through machine learning techniques".

the celestial artificer, Hephaestus, crafted the automaton Talos as a gift to Minos; his machine was meant to help the first king of Crete guard his island. Other stories have also been pointed to, in which inventors see their creations brought to life. With this vision of artifacts capable of thought and action of their own, it is no surprise that when computers were first conceived, people wondered if they would ever become intelligent.[3]

The early days of artificial intelligence were centered on using computers to solve difficult, but well-defined problems. Where these problems could be described with clear mathematical rules, they would often prove much easier to solve with a computer than it was to solve them by hand. But problems that are relatively simple for humans to solve intuitively – problems like voice and facial recognition, the examples mentioned earlier – carried too much complexity to be easily captured in a set of hand-crafted mathematical rules. Thought they were intuitive for us, these problems were difficult for a computer to solve. This, then, became the goal of machine learning; and, for the more complex of problems, the goal of deep learning (DL) specifically: in simple terms, the objective was to enable a computer to learn from experience, from data, to identify its own choice of rules which would then allow it to solve those tasks that are intuitive to us but difficult for us to formulate mathematically.

Bringing it back to modern day, the field of deep learning has now gone through three core waves of interest. The first of these spanned the 1940s-1960s, when the field was known as *cybernetics*; this first wave captured developments such as the K-means clustering algorithm and the perceptron algorithm. The second spanned the 1980s-1990s, under the name of *connectionism*, was sparked by the development of the backpropagation algorithm. The third wave of interest in AI and ML principles, which we are living through, is generally considered as starting around 2006.[4]

[3] Goodfellow, Bengio, and Courville, *Deep Learning*.
[4] Goodfellow, Bengio, and Courville, *Deep Learning*; Bianco, Gerstoft, Traer, Ozanich, Roch, Gannot, and Deledalle, "Machine learning in acoustics: Theory and applications".

The literature on artificial intelligence and machine learning is extensive, and has covered many distinct algorithms and applications, as well as general resources and methodologies. Borrowing from the concepts of the Experiential Learning Theory,[5] the next sections in this chapter will introduce concepts of ML in the context of two experimental challenges[6]. However, readers interested in learning more about the field of Machine Learning in general are encouraged to seek some of the aforementioned literature, of which the following select texts are recommended: Bishop (*Pattern recognition and machine learning*); Goodfellow, Bengio, and Courville (*Deep Learning*); Schölkopf and Smola (*Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*). In the context of ML for acoustics, the survey by Bianco, Gerstoft, Traer, Ozanich, Roch, Gannot, and Deledalle ("Machine learning in acoustics: Theory and applications") is also highly recommended.

3.2 Behavior classification : understanding your signals

In seeking to teach a machine to perform a new task, we as developers and operators can benefit from reflecting on our own learning process. Kolb's learning theory can be summarized as a four-stage cycle, consisting of: (i) having a concrete experience as active participants; (ii) reflecting on our observation from that experience, paying close attention to instances such as when what happened was not what we expected; (iii) adjusting our expectations, or developing a new understanding of what happened, a process also known as abstract conceptualization; and then (iv) engaging in active experimentation, testing our new understanding and expectations to new situations. As a first applied example of ML, then, let us consider the task of behavior classification in the light of those first three stages.

The task at hand consists on the application of ML techniques to enable an AUV to perform classification of an external collaborator's behavior; that is, to determine what the collaborator is

[5] Kolb, *The experiential learning theory of career development*.

[6] "I hear and I forget. I see and I remember. I do and I understand." This quote, attributed to Confucius, summarizes Kolb's theory wonderfully; the crux of Experiential Learning Theory is that a person learns through action.

doing without engaging in direct and explicit communication. Thus, the first three stages can be regarded as follows:

1. **Baseline observations**: what have we seen in the field, which may relate to this task? Which signals and tracking techniques can we take advantage of?
2. **Reflecting on the challenges**: what has made, or could make, this task a challenge with respect to the earlier observations? Are there any apparent limitations to the chosen inputs?
3. **Recurring themes as signals**: can we identify recurring themes, and use them to develop a new understanding of the system? How much of this new understanding can we carry with us to new challenges in the future?

3.2.1 First stage : baseline observations

These baseline observations effectively amount to a supplement of the background presented in Chapter 2. They are presented here, in the context of the particular application for which they are used in this thesis.

Swarm operations : a collaborative framework

With the advent of lower-cost AUVs, a small fleet of moderately equipped vehicles may well become a more attractive proposition than that of a single, large vehicle equipped with top of the line sensor systems.[7] Even if the overall cost remained the same across both scenarios, a fleet of smaller vehicles could still benefit from covering a large area in a smaller amount of time. Indeed, the same fleet concept carries over to search and rescue operations over land, where teams of volunteers and trained personnel sweep an area in a coordinated fashion, in hopes of minimizing the odds that any critical clues will be missed. While AUV operations were conducted as part of the Malaysia

[7] Viquez, Fischell, Rypkema, and Schmidt, "Design of a general autonomy payload for low-cost AUV R&D".

Airlines flight 370, the search over an area of 860 square kilometers spanned 70 operational days;[8] how much faster could this have been done with a suitable fleet of smaller vehicles?

In this kind of operations, time is generally considered a critical factor, and when an operation faces the added risk that a target will be displaced (say, by ocean currents, for example), the importance of a timely search becomes ever more significant. Sontag (*Blind man's bluff : the untold story of American submarine espionage*) tells the story of how a similar search operation conducted in the late 1960's nearly ended in failure. The team's efforts had been concentrated in the wrong area for much of their time, and the worsening weather had caused leadership to order an end to the search. It was only the persistence of Chester Buchanan, one stubborn oceanographer and senior NRL scientist; and the wits of John Craven, the chief scientist of the Special Projects Office of the US Navy who helped pioneer the use of Bayesian search techniques to locate objects lost at sea.

From the above historical events and many more experiences like them, along with the newfound availability of lower-cost underwater vehicles, is that research projects in the AUV community steered towards swarm operations. Formation control, for example, can be a challenge for a set of vehicles operating in a current or other drifting conditions,[9] especially if they are tightly packed and have limited communication capabilities. As underwater communications imply hardware and power requirements that increase the cost of these smaller-scale vehicles, there have also been some projects that look at enabling performant single-beacon navigation for these vehicles, such that coordinated fleet operations can be conducted without the need for active acoustic hardware on all units.[10]

[8] LeHardy and Moore, "Deep ocean search for Malaysia airlines flight 370".

[9] Rypkema, "Distributed Autonomy and Formation Control of a Drifting Swarm of Autonomous Underwater Vehicles".

[10] Rypkema, Fischel, and Schmidt, "Closed-Loop Single-Beacon Passive Acoustic Navigation for Low-Cost Autonomous Underwater Vehicles"; Rypkema, "Underwater & Out of Sight: Towards Ubiquity in Underwater Robotics".

Collision avoidance, in a general sense, presents many challenges. We see these difficulties on the road every day, where we have to judge other drivers' behavior and alertness. We also have to account for road conditions, and how the weather may affect our ability to safely operate a vehicle. These challenges carry over to self-driving cars, as their perception of the world differs from ours; in some ways for the better, but in others for the worse. While significant progress has been made in the development of advanced driver-assistance systems (ADAS), failures in self-driving vehicle systems and their operator's behavior have ultimately led to accidents;[11] as similar accidents have occurred with purely human-operated vehicles, these events ultimately serve to emphasize the challenges of collision avoidance – a difficult task for human and machine alike. Though they are relatively uncommon in commercial aviation, mid-air aerial collisions between two planes have also occurred in the past.[12] Marine vehicles are not immune to this kind of tragedy either, with the events involving the US Navy's destroyers USS Fitzgerald and USS John McCain serving only as two recent examples.[13]

The standard sensor suite we have come to expect of top of the line ships – GPS for self-localization; the automatic identification system (AIS) to share position information with nearby vessels; radar and other sensors for additional detection capabilities – has proven insufficient for preventing costly accidents. With the intent of augmenting the information available to marine systems for collision avoidance, US Patent 8,830,793[14] described a system that could estimate time-to-intercept (TTI) from the acoustic signal of an approaching vessel. The patent's approach parted from Equation 3.1, which states the expected acoustic intensity I_{dB} to be detected by a receiver when accounting for

[11] Singhvi and Russell, "Inside the Self-Driving Tesla Fatal Accident"; Griggs and Wakabayashi, "How a Self-Driving Uber Killed a Pedestrian in Arizona".

[12] National Transportation Safety Board, *Midair Collision over George Inlet de Havilland DHC-2, N952DB, and de Havilland DHC-3, N959PA*; National Transportation Safety Board, *Mid-Air Collision: Ongoing Investigation, Accident No. CEN21FA215*.

[13] Rich, "7 Navy Sailors Missing AFter U.S. Destroyer Collides With Merchant Vessel Off Japan"; Beech and Haag, "10 Missing After U.S. Navy Ship and Oil Tanker Collide Off Singapore".

[14] Schmidt and Benjamin, *System and Method for Collision Avoidance in Underwater Vehicles*.

range-decay relative to a constant source. This formulation generally assumes cylindrical spreading as the more conservative basis; cylindrical spreading is also a more likely approximation for shallow water environments, for which this system was originally envisioned. The TTI formulation is thus captured in Equation 3.2, where $\dot{I}_{\text{meas}}$ represents the change in acoustic intensity measured by a sensor in the field; the time-to-intercept value is inversely proportional to the measured change in intensity.

$$I_{dB} = I_0 - 10\log_{10}(r) = I_0 - \frac{10\log(r)}{\log(10)} \tag{3.1}$$

$$\text{TTI} \equiv -\frac{r}{\dot{r}} = \frac{10}{\log(10)\dot{I}_{\text{meas}}} \tag{3.2}$$

Beamforming : tracking relative bearings

Whether we talk about the children's game "Marco Polo" or a bat's use of echolocation, the following statement stands true: the direction from which a signal is detected is very important in many applications. In the domain of signal processing, the spatial filtering process related to determining the direction from which a signal is coming into a receiver array is called beamforming. Furthermore, reciprocity applies here; the same process can be used to transmit signals in a particular direction. Thus, what humans and bats do naturally can be transferred to other applications – we can observe these two processes (directional transmission and reception) used in modern high-end wireless communications, for example. Top-of-the-line wireless routers feature beamforming capabilities as a means of lowering power consumption while attaining better performance – such as reaching targets at longer ranges. The same principles apply, whether we are talking about the electromagnetic waves of wireless radio communications or the mechanical pressure waves of acoustic systems – the latter being the case for that well-known children's game and the bats mentioned be-

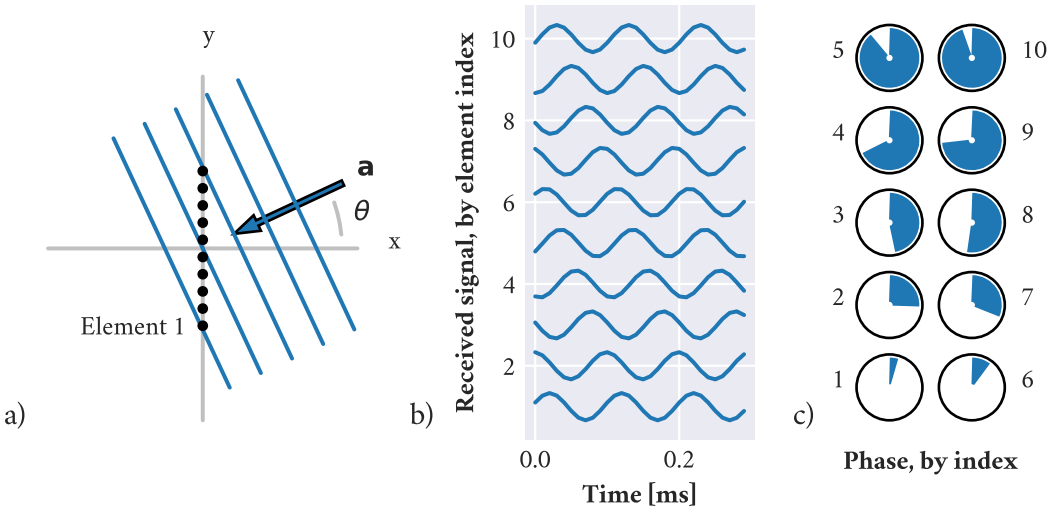


Figure 3.1: An illustration of fundamental concepts in beamforming. a) A 10-element line array in 2D space, with a plane wave input; the plane wave is traveling in the direction **a** (system state shown for $t = 0$). b) The signal recorded by each receiver as a function of time. c) The signal phase recorded for each receiver when processing a snapshot of the data (as shown in b).

fore, and for the acoustic systems employed onboard autonomous underwater vehicles and crewed submarines alike.

Let's begin the presentation of the mathematical formulation of the conventional beamformer by setting a physical point of reference. Figure 3.1(a) shows a line array in 2D space, and an incident plane wave that serves as input to the elements of the acoustic array. For the purposes of this discussion, let's consider the left-most blue line as a global leading edge of the signal (the line that runs closest to element no. 1); all subsequent blue lines represent the start of a new cycle in the signal, and the signal in this case is a sinusoidal function. As drawn, the plane wave is traveling in the direction of vector **a**, at some angle θ from the horizontal axis. For the 2D case, the direction vector **a** is given by:

$$\mathbf{a} = \begin{bmatrix} -\cos(\theta) \\ -\sin(\theta) \end{bmatrix} \tag{3.3}$$

For a plane wave propagating in a locally homogeneous medium, the wave fronts shown in Figure 3.1(a) can be described by the wavenumber $\mathbf{k}$:

$$\mathbf{k} = \frac{\omega}{c}\mathbf{a} \tag{3.4}$$

In a simple harmonic oscillator, we would typically have the system's phase given as the product of time t and the system's frequency ω; since we are considering multiple receivers distributed in space, let us use a value $\tau_n = t + c_n$ for each of the n element, such that the constant c_n effectively corrects for the offset between the leading edge of the signal and the receiver. To illustrate this, consider element no. 1 near the global leading edge, as well as element no. 10 nearly two full cycles into the wave pattern shown; relative to the front that crosses the origin (the second blue line), these two points are nearly one full cycle ahead and nearly one full cycle behind. This effect is also illustrated in Figure 3.1(b), which shows the data recorded by each element as a function of time t; the spatial separation translates to a phase shift of the sine wave.

It should be apparent that the value of c_n would change with the angle θ. More importantly, this correction can also be expressed in terms of the spatial parameters of the system, given by the wavenumber $\mathbf{k}$ which characterizes the wave, and the position vector $\mathbf{p}_n$ for each element:

$$\omega\tau_n = \mathbf{k}^T\mathbf{p}_n \tag{3.5}$$

Using this form for the spatial correction readily accounts for the effect of the incident angle θ, which is captured in the direction of the wavenumber vector $\mathbf{k}$. In other words, the dot-product of $\mathbf{k}$ and $\mathbf{p}_n$ simply represents the number of cycles that separate the zero-phase reference (the front that crosses the origin) and the n-th receiver, along the direction of the plane wave. As the signal is a sinusoidal function, it may help to consider the phase recorded for each element at time $t = 0$, as shown in Figure 3.1(c). As mentioned before, with respect to the spatial distribution shown in

(a), element no. 1 is just past the beginning of a cycle; as we move up the array, we see the elements reporting increasingly large phase offsets. Element no. 6 is just past another cycle start line, and the phase shift shown conveys that clearly; from there, we move up to element no. 10 which is nearly at the end of the second cycle. Indeed, the phase shifts shown in (c) are sufficient to construct the signals in (b); these are equivalent to $\mathbf{v_k}(\mathbf{k})e^{jwt}$, where the replica vector $\mathbf{v_k}(\mathbf{k})$ is given by:

$$\mathbf{v_k}(\mathbf{k}) = \begin{bmatrix} e^{-j\mathbf{k}^T\mathbf{p}_1} \\ e^{-j\mathbf{k}^T\mathbf{p}_2} \\ \vdots \\ e^{-j\mathbf{k}^T\mathbf{p}_n} \end{bmatrix} \tag{3.6}$$

The importance of this representation stems from the following idea, which lives at the heart of the conventional beamformer: if we can represent the expected signals as we do with the replica vector $\mathbf{v_k}(\mathbf{k})$, then we can compare the expected replicas with the measured signals to determine the direction of the signal. To do so, a snapshot of the time-series data collected by the receiver is converted to frequency domain using a Fourier transform; the output will be of the same form as $\mathbf{v_k}$, to a scaling constant that represents the signal amplitude. Then, the agreement between a directional replica $\mathbf{v_k}(\mathbf{k_s})$, which is steered to an angle θ_s, and the measured signal $\mathbf{v_k}(\mathbf{k_m})$ can be expressed by:

$$B(\omega, \theta_s) = \frac{1}{N}\mathbf{v_k}(\mathbf{k_s})^H\mathbf{v_k}(\mathbf{k_m}) \tag{3.7}$$

Equation 3.7 above is known as the *conventional beam pattern*, and it plays a key role in the development of more advanced techniques - namely, optimum and adaptive array processing. The term $\mathbf{v_k}(\mathbf{k_s})^H$ represents the conjugate transpose of $\mathbf{v_k}(\mathbf{k_s})$. Additionally, the $\frac{1}{N}$ term acts as a normalization coefficient, since the replica vectors $\mathbf{v_k}(\mathbf{k_s})$ contain unit-amplitude complex terms in the case of uniform amplitude weighting. Figure 3.2 shows the beam pattern produced by using the replica

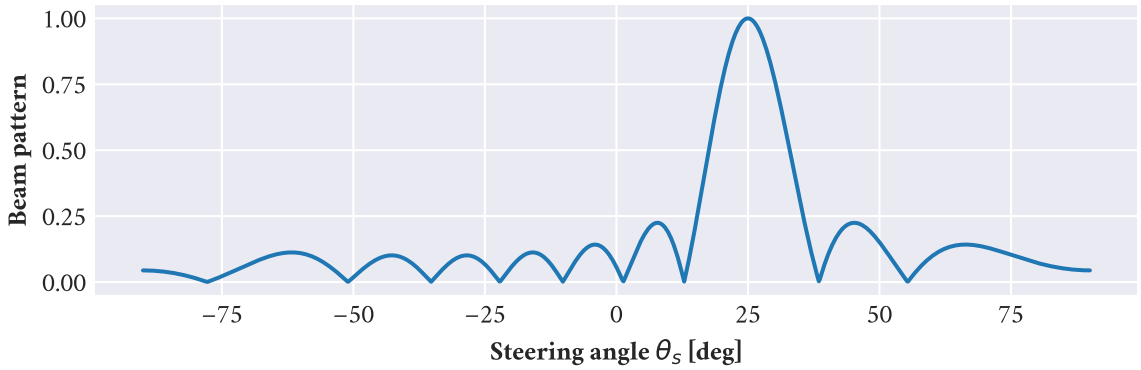


Figure 3.2: Magnitude of the beam pattern obtained by scanning across steering angles θ_s, for the setup shown in Fig. 3.1.

vector to scan across the set of steering angles $\theta_s = [-90, 90]$; the input plane wave shown in Figure 3.1 is arriving at an angle of 25 degrees.

More generally, the conventional beam pattern can be rewritten as in Equation 3.8, where the weight vector $\mathbf{w}$ is given by Equation 3.9. The replica vector $\mathbf{v_k}(\mathbf{k_s})$ is written as $\mathbf{v_s}$ for convenience.

$$B(\omega, \theta_s) = \mathbf{w}^H \mathbf{v_m} \tag{3.8}$$

$$\mathbf{w}^H = \frac{\mathbf{v_s}^H}{\mathbf{v_s}^H \mathbf{v_s}} \tag{3.9}$$

In more advanced implementations of this spatial filtering technique, additional information is captured in the weight vector $\mathbf{w}$. Null-steering, for example, can be used to account for known sources whose effect needs to be removed – such is the case with jamming signals. The spectral covariance matrix of the signal noise, $\mathbf{S_n}(\omega)$, or the spectral covariance matrix of the entire input signal, $\mathbf{S_x}(\omega)$, can be used to augment the resolution of the beamformer. These two matrices are used in the Minimum Variance Distortionless Response (MVDR) beamformer and the Minimum Power Distortionless Response (MPDR) beamformer respectively, where the weight vector takes the form shown in Eq. 3.10 after substituting for the appropriate $\mathbf{S}$ matrix.

$$\mathbf{w}_{\text{MVDR}}^{H} = \frac{\mathbf{v_s}^{H}\mathbf{S_n}^{-1}}{\mathbf{v_s}^{H}\mathbf{S_n}^{-1}\mathbf{v_s}}, \qquad \mathbf{w}_{\text{MPDR}}^{H} = \frac{\mathbf{v_s}^{H}\mathbf{S_x}^{-1}}{\mathbf{v_s}^{H}\mathbf{S_x}^{-1}\mathbf{v_s}} \tag{3.10}$$

This is but a brief presentation of a rather complex part of signal processing. Readers interested in learning more about conventional and optimum beamformers, as well as the adaptive forms of the filter, are encouraged to check out the work of Jensen, Kuperman, Porter, and Schmidt (*Computational ocean acoustics*), and Van Trees (*Optimum Array Processing*). Both of these texts offer detailed derivations and additional considerations for using this type of spatial filtering.

3.2.2 Second stage : reflecting on the challenges

The baseline observations above exhibit the following properties:

- **Swarm operations**

  Multi-vehicle operations are challenging. The use of an active acoustic system, such as in the single-beacon work cited earlier, creates an opportunity for coordinated operating modes based on a small number of predefined signals. A well-defined schedule (when all nodes have a reliable time reference) also facilitates the signal processing step. Under such conditions, the requirements for information shared across all vehicles in the system is non-trivial – signal replicas and timing must be common to all units.

- **Time-to-Intercept**

  The mathematical formulation for TTI is related to the derivative of the intensity measurement, which is a noisy signal. This means that using the algorithm blindly will produce a very noisy output. However, taking steps such as frequency filtering and signal smoothing prior to estimating TTI can improve the reliability of the results. Using additional information about the system, such as the presence of straight shipping lanes, can also be exploited

to improve the interpretation of the noisy input. Generally speaking, a vehicle or mooring tasked with computing TTI in a low to moderate traffic region requires relatively little information about the system-wide characteristics; as long as contacts are sufficiently spaced apart from one another, the broadband intensity can be used to produce useful TTI values even if a target's acoustic signature is unknown.

- **Beamforming**

  The spatial filter requires some information about the expected signal properties. For example, are we tracking a narrow-band signal, such that the replica vectors are only varying with respect to the arrival angle as in Figure 3.2; or are we looking for a broadband signal, such that we'll need to scan across angles and frequency alike? Indeed, the physical layout of the array itself is guided by the expected signal frequency, as the element spacing and array length vis-à-vis the signal's wavelength drive the array's resolution. Beyond the effects of signal frequency on the array's design and resolution, beamforming is by and large a passive technique; as with TTI, beamforming techniques can produce useful information about the environment, even with relatively little external knowledge.

Each of the three categories above have some requirement of system-wide information to perform well. However, the required knowledge threshold between them varies significantly. The work on single-beacon navigation for vehicle swarms presented by Rypkema, Fischel, and Schmidt ("Closed-Loop Single-Beacon Passive Acoustic Navigation for Low-Cost Autonomous Underwater Vehicles") requires a notable level of coordination and a dedicated source; not a source of opportunity. On the other hand, the TTI and beamforming techniques can be used to track sources of opportunity – meaning sources that do not actively collaborate with the tracking unit – as long as the respective processing pipelines can observe the signal produced by the source of opportunity.

From these observations and challenges, then, follows the question: could we use passively acquired data, such as the outputs of the TTI and beamforming pipelines, to do some (if not all) of

the work in an active pipeline like the one used for swarm operations, with respect to coordinating operational modes?

3.2.3 Third stage : recurring themes as signals

The previous question lives, of course, at the heart of this third stage in the experiential learning framework. The requirement for a positive response would be that there exist a recurring theme in the data, which may be exploited as a source of information about the system. Indeed, the concept of pattern recognition is well established as a part of machine learning and is the subject of the book by Bishop (*Pattern recognition and machine learning*). Another project that uses pattern recognition across two different sources - transmission loss and arrival time - to discern information about an acoustic system can be found in simulation-based work presented by Fouquette ("Multipath Arrival Tracking for Marine Vehicles Utilizing Pattern Recognition").

Now, let us go back to the aforementioned question: could we use passively acquired data to coordinate operational modes? To answer it, let us consider the signals presented in Section 3.2.1 as illustrated in Figure 3.3. Here, two sets of signals are shown: one for a ship moving along a transect and another for a ship loitering about a point at some distance from the receiving array. Both paths shown have been configured to share a common closest point of approach (CPA), and the time-series data has been shifted so both vehicles cross the CPA at time $t = 0$.

For a simple enough scenario such as the one shown, and given enough data in time for each signal, it seems as though the cyclical nature of the loiter path would certainly stand out against the linear transect. However, two limitations of this plot ought to be considered: the impact of the signal-to-noise ratio (SNR), and the effect that transect and loiter parameters may have on the uniqueness of the signals.

The first of these limitations is of general importance in most signal processing applications, and ours is no exception. Both the beamformer and TTI algorithms depends on having a sufficiently

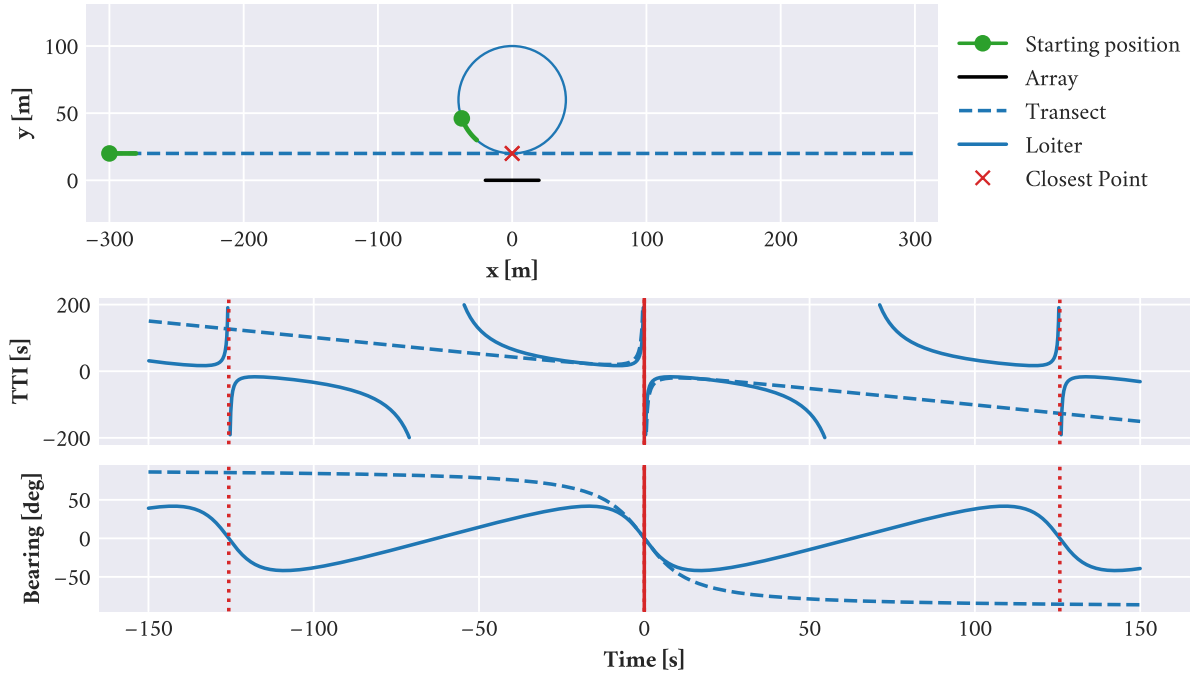


Figure 3.3: Sample bearing and time-to-intercept signals produced by ships moving along a transect and a loiter pattern. The starting positions and initial steps are shown in green, while the closest point along both paths are marked with a red cross. In the time-series data, the intercept time for both transect and loiter are shown as a solid red vertical line, while additional approaches from the loitering vehicle are shown with dashed red lines. Bearing is given relative to the array's broadside, aligned with the vertical axis.

large signal-to-noise ratio in order to produce valuable information from their inputs. For example, the expected signal intensity used for the TTI solution was given by Eq. 3.1 under the assumption of cylindrical spreading, and ignoring noise. As a derivative relation, noise in the input signal would be effectively amplified in the TTI output; in the extreme, a low SNR would cause the TTI output to be of little use. The same goes for the beamformer, where a low SNR may cause the filter to report similar energy values for all bearings and thus make it impossible to track a source.

The second limitation relates the parameters of the ship's path to the scale and nature of the features produced by the two signal processing blocks. For the transect shown in Fig. 3.3, the ship travels parallel to the array; given the relatively small distance from the array to the CPA, the array reports a bearing close to 90deg. As the ship crosses the array's broadside at its closest point, it reverses the sign of the asymptotic bearing. For comparison, a transect that ran perpendicular to the array's axis (but offset from the array's center) would start with some distinct asymptotic bearing; as the ship passed over the array, the bearing would peak at either endfire ($\pm$ 90deg) and then return to the original asymptotic bearing.

Indeed, the parameters that define both scenarios are already enough to create a sizeable search space for a naive, "brute force" approach. For the linear transects, the ship heading and speed are combined with the CPA to fully define the signal shapes. The loiter is defined by the coordinates of the pattern's center, as well as its radius and the ship's speed (the CPA can be drawn from these values). Initial conditions, in this case meaning the vehicle position along the parameter-constrained path at time $t = 0$, can be regarded simply as a shift of the signal's pattern along the timeline. Figure 3.4 shows snapshots from a number of randomized simulations for both transects and loiters, where these shifts can be observed.

A well-known characteristic of time-series data is that it often exhibits time-scaling variability that can be described as slowing down or speeding up. In the case of the beamformer and TTI outputs, it should be apparent that vehicle speed will impact the time scale of the signal; but Figure

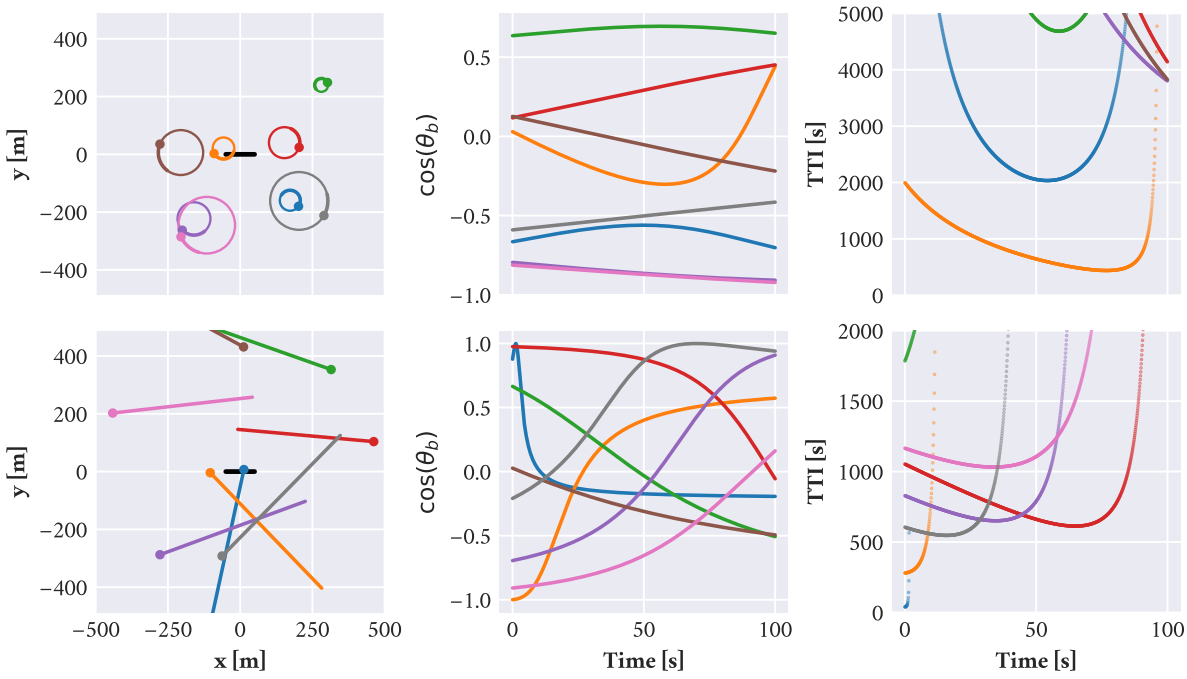


Figure 3.4: Behavior classification from bearing and time-to-intercept (TTI) signals detected at a receiving array; signals are produced by an external vehicle moving relative to the array. This comparison illustrates the challenges that come with conventional "feature design" solutions, as the signals exhibit distinct patterns which are generally visible to the human eye, but require additional processing before they may be classified by a computational algorithm.

3.4 also illustrates how the distance to the closest point of approach (CPA) can also impact the rate of change in the signal. The further away a target vehicle's CPA is, the slower its perceived bearing rate will be as it crosses this critical reference point; we can compare this scenario to us visually tracking a plane in the sky.

Since the various parameters that define the travel patterns in our classification task are unknown, it is necessary for our application to process the signals using a technique that can account for such temporal distortions. This application therefore relies on Dynamic Time Warping (DTW), a technique used in areas where such speeding up and slowing down is expected, such as motion studies and speech classification. DTW is a non-linear transform; at its heart, it allows us to compare two time-series shapes by stretching the sequences in such a way as to minimize time-induced differences between samples, and then reporting the differences between the transformed sequences in a distance-like metric.

The non-linearity of DTW comes from the algorithm's governing rules for stretching each signal. These rules can be conceptualized as follows:

- The transformed signals will have the same length, with the first and last point of each output being anchored to the first and last sample of the corresponding input sequence.
- Every sample in each of the input signals has to be matched to at least one point of the other signal.
- The transforms should be monotonically increasing. When matching one point from signal A to many points of signal B, the points of signal B must be adjacent (no skips).
- In many applications, an additional constraint establishes a maximum number of matches per sample. This is also referred to as enforcing a maximum window width.

The optimal DTW solution would be the one to minimize the distance between the output signals while obeying these rules. The non-linearity that comes from this warping is what allows us to evaluate the information content of the sequences independently of the time scaling.[15]

Dynamic Time Warping exposes a comparative metric between two time-series signals. Thus, using it in a classifier effectively implies that there ought to be known records with which field measurements shall be compared. One method that can be used to build such a classifier, then, is to collect a library of labeled samples; any new measurements can be compared with the entire library to produce a classification score. This score can be based on a single nearest-neighbor (1-NN) distance, or expanded to a k-nearest-neighbors (k-NN) form.

In the context of behavior classification based on the beamforming and TTI signals, we opt for a k-NN approach given its added robustness relative to 1-NN. The final classification is produced by majority vote among the selected neighbors rather than by a single match, thus increasing the confidence that a class choice fits more of the training data and is less likely matching some outlier. The performance of this k-NN classifier was evaluated against simulation data as well as field measurements; these performance results were presented in Fischell, Viquez, and Schmidt ("Passive acoustic tracking for behavior mode classification between surface and underwater vehicles").

3.3 Environment characterization : exploiting *a priori* information

The previous section looked at how a vehicle can evaluate the information it has collected about other vehicles around it to inform its decision making process. That vehicle perspective was framed by the first three stages of Kolb's Experiential Learning Theory. The fourth stage in Kolb's theory

[15] Dau, Keogh, Kamgar, Yeh, Zhu, Gharghabi, Ratanamahatana, Yanping, Hu, Begum, Bagnall, Mueen, Batista, and Hexagon-ML, *The UCR Time Series Classification Archive*; Dau, Silva, Petitjean, Forestier, Bagnall, Mueen, and Keogh, "Optimizing dynamic time warping's window width for time series data mining applications".

consists of engaging in active experimentation, testing our new understanding and expectations to new situations; it is here that we assess our understanding of the information availability and information flow in the vehicle's decision pipeline. To that end, this section considers another classification task: one aimed at environmental characterization.

The core of the vehicle behavior classification algorithm depended on our ability to produce reliable and realistic feature samples through simulation. A number of simulated runs, stored as replica vectors in the training set for the k-NN classifier, were compared with new measurement vectors to produce an output class. The performance of the algorithm was obtained by performing that very comparison using additional simulation runs as well as snapshots collected from field measurements. An important concept in this entire process is that we were evaluating predetermined behaviors – we sought to distinguish a vehicle moving in a loiter pattern from one moving in a linear transect. The simulation data and field tests were both limited to this choice of behaviors – in other words, we were exploiting additional information about the problem that was not inherent to the classification task itself.

The idea of using preexisting, or *a priori*, information is fundamental to many applications. In problems related to navigation, for example, knowledge about a system's initial conditions and its sensing capabilities as well as the expected process errors are used to build Kalman filters of many kinds. This also transfers readily to the evaluation of environmental models; if we can provide the system with a sensible set of characteristic environments, we can once again use a nearest neighbor classifier to determine which model best describes the data.

3.3.1 Assembling candidate models

The idea of using preexisting, or *a priori*, information is fundamental to many applications. In problems related to navigation, for example, knowledge about a system's initial conditions and its sensing capabilities as well as the expected process errors are used to build Kalman filters of many kinds.

This also transfers readily to the evaluation of environmental models; if we can provide the system with a sensible set of characteristic environments, we can once again use a nearest neighbor classifier to determine which model best describes the data.

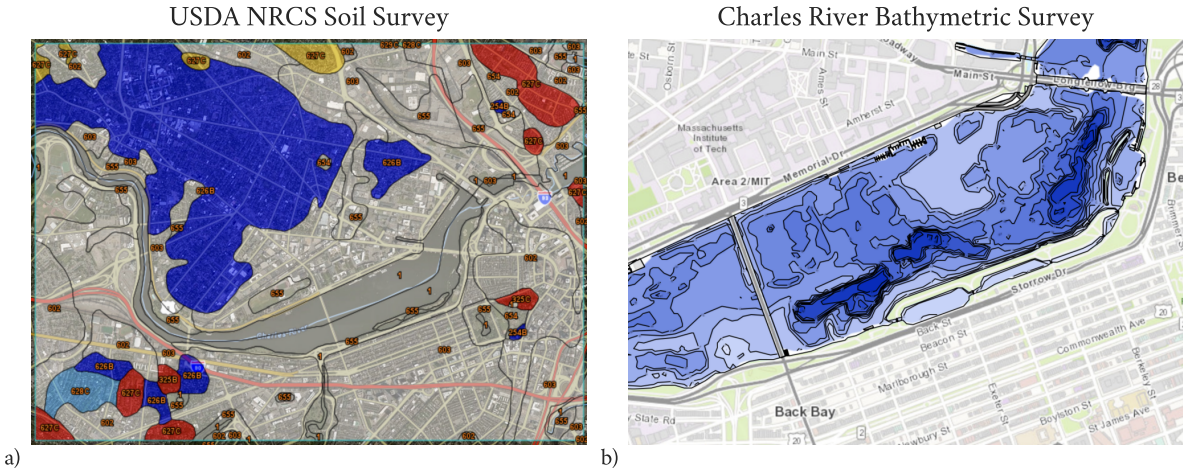
There is an important difference between the models for behavior and environmental classification. The behavior work could benefit from our choice of patterns as the *a priori* information used to produce the sample signals. However, the work on environmental models requires that we explore data over which we have no control – data that is nonetheless highly related to the problem we are trying to solve. The process of assembling the candidate models, thus, requires us to look for external sources of information to serve as a foundation. The acoustic model we intend to use for this task requires information about the water column, such as the sound speed profile; it also requires knowledge of the composition of the bottom layer and its depth.

Figure 3.5 illustrates two surveys that can be used to prepare candidate models for the classification task in the Charles River Basin. A soil survey, such as those facilitated by the Natural Resources Conservation Service of the US Department of Agriculture,[16] can offer some insight to the likely composition of the riverbed. The percentage of sand for a given soil complex, for example, can be collected from the report. Other sources, such as knowledge about the urban drainage system or experiential knowledge about the site can also offer information[17] worth exploring with the candidate models. Likewise, bathymetric surveys like the one released by the MIT Sea Grant College Program and the Charles River Alliance of Boaters[18] provide information that is directly relevant to the propagation models; the depth of the water column can have a notable impact on predicted boundary interactions. These charts can also inform choices made for the modeling stage, such as whether a waveguide model with a representative depth is sufficient, or whether a higher resolution

[16] Natural Resources Conservation Service, *Web Soil Survey*.

[17] It is fairly common for sailboats that turned turtle (boats that are fully inverted, with their mast pointing straight to the bottom) in the Charles River Basin to show some mud on their mast and sail, when they've been righted.

[18] Zimba, Sacarny, Yoder, and Bray, *Chart of the Lower Charles River*.



a) b)

Figure 3.5: Reference surveys for the Charles River Basin. (a) Soil surveys provided by the Natural Resources Conservation Service of the US Department of Agriculture can be used to identify the presence of materials such as sand (shown), clay, and silt in local substrates; the Merrimac-Urban land complex (blue), for example, has a weighted average of 82.8% sand content across all depth layers. (b) Bathymetric surveys such as the one shown, by the MIT Sea Grant College Program and the Charles River Alliance of Boaters, inform decisions about a propagation model's bottom boundary.

approach – one that actually captures the boundary changes with respect to range from the source – is needed for a particular application.

Table 3.1: Parameters for acoustic modeling, by material

Property	Units	Mud	Basalt	Sand
Compressional speed	[m/s]	1500.0	5250.0	1640.0
Shear speed	[m/s]	30.0	2500.0	110.0
Compressional attenuation	[dB/λ]	0.5	0.1	0.8
Shear attenuation	[dB/λ]	1.0	0.2	2.5
Density	[g/cm^3]	1.8	2.7	1.9

Parting from the aforementioned surveys and external sources, then, let us consider two riverbed materials to start with: sand, given the high percentage contained in the neighboring Merrimac-

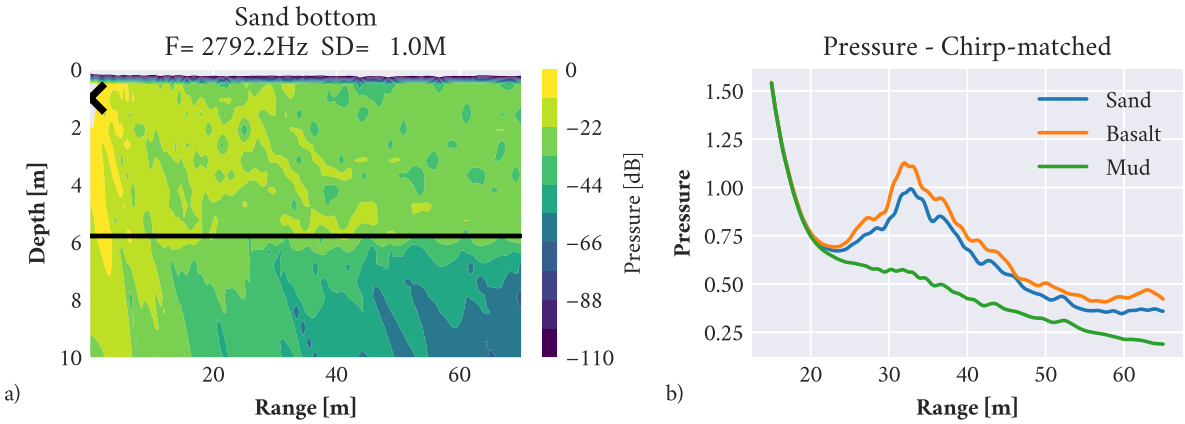


Figure 3.6: Acoustic models for riverbed characterization. (a) Transmission loss chart for the full field, assuming an isovelocity profile with $c = 1451$ and a uniform sand bottom starting at 5.8m (19ft). (b) Power detected by a receiver at a depth of 1 meter, as a function of range, for the three riverbed materials evaluated by the classifier.

Urban land complex; and mud, given the experiential knowledge about the Charles River Basin. A third material, basalt, will also be considered to provide for a comparative discussion later on. Additionally, let us consider a simplified, uniform bathymetry with a typical value of 5.8m (19ft). Figure 3.6(a) illustrates the full field transmission loss for the sandy riverbed, given the uniform depth assumption. Figure 3.6(b) shows a slice of the field, for a receiver at 1m depth; the corresponding pressure values are shown for all three riverbed materials. The properties of each riverbed material used for acoustic modeling are given in Table 3.1.

As with the behavior classification exercise, we need to build a set of replica vectors that will constitute the training set for the k-NN classifier. Where the behavior classifier explored a large domain of path parameters with variable time scaling, the models presented here for environmental characterization are much more tightly defined. The density of mud, for example, can go from around 1.73 g/cm^3 for flowing mud, to 1.84 g/cm^3 for a more steady mud; we will consider only a representative value for each of the material properties as given in Table 3.1. However, the reduced set of parameters will be balanced by the uncertainties in the sensor systems and the experimental

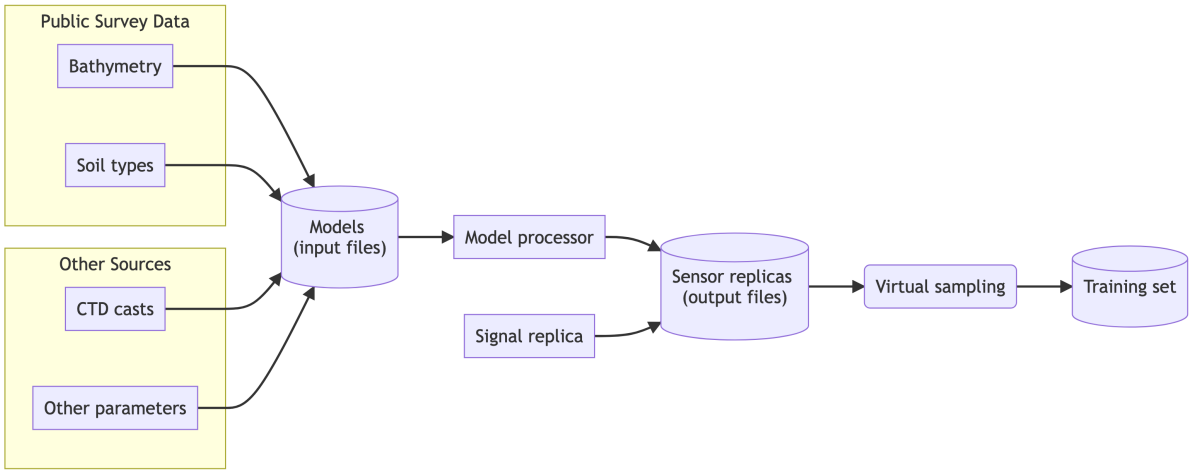


Figure 3.7: Schematic of the model preparation process for virtual sampling.

process overall. These can be introduced to the simulation-based data through virtual sampling, the second-to-last stage in the model preparation process shown in Figure 3.7.

3.3.2 Virtual sampling as a stochastic observer

The objective of virtual sampling is to ingest model-based or simulation-based data and produce an output that is comparable to that expected of real sensors. As such, the process should allow us to simulate field work. A kinematic model, for example, can be used to emulate the motion of the sensor (and the vehicle that carries it) in space, to determine the sampling region from which a measurement shall be collected. The samples can be collected from within the sampling region neighboring the position determined by the kinematic model, rather than directly at the simulated vehicle's position, to reflect potential errors in timing and positioning. Randomized sensor noise can also be added to the data to reflect the sensor's expected performance. Thus, virtual sampling also allows us to replicate the effect that limitations of the experimental process may have on the

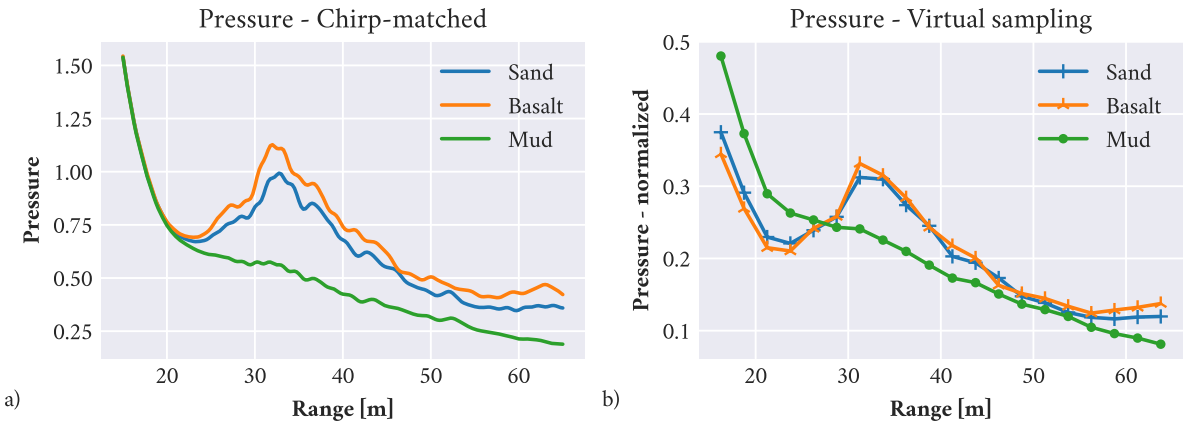


Figure 3.8: The power predicted by the acoustic propagation model (a) is randomly sampled, and noise is added to reflect sensor and process error. Each set of random samples is discretized in range to construct a virtual sampling vector (b), which is added to the training set. This process is repeated multiple times per model to ensure the training set contains enough sample vectors, and sufficient information about the model, to be used by the classifier.

data collected. This is illustrated in Figure 3.8, where the data from the acoustic propagation model is randomly sampled and discretized into range bins. The resulting vectors are also normalized, to focus on signal shapes rather than signal amplitudes which would be affected by a source level that may vary between experiments.

The choice to work with a discrete subset of ranges stems from the fact that there is no guarantee the field measurements would necessarily align with some predetermined choice of ranges. Furthermore, the replica vectors should reflect the fact that our assumptions – for example, to work with a constant depth in this case study – are not a perfect match to reality; we may be confident that large-scale features are robust to mismatch, but smaller-scale structure in the signal may not accurately represent reality. Opting for a discrete subset of ranges, then, allows us to compare the simulation and field data with relatively little cost.

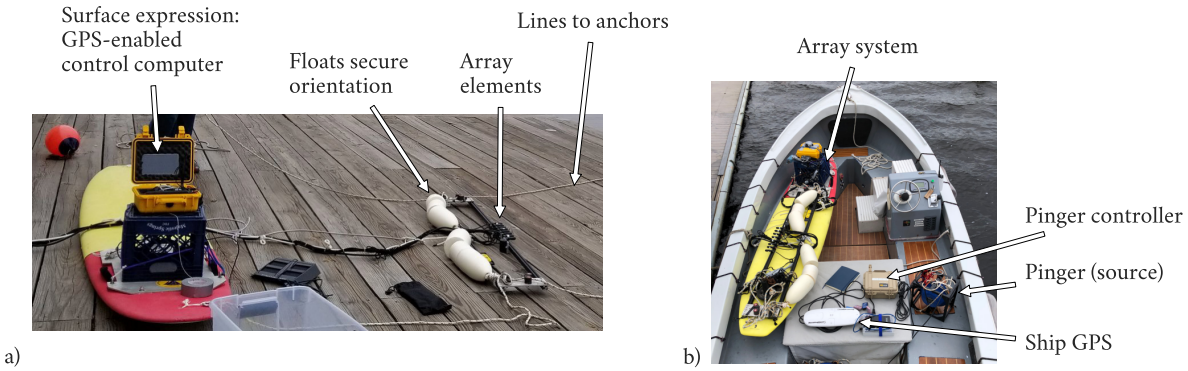


Figure 3.9: Equipment used to collect acoustic data for the environment classifier. (a) Components for a stand-alone acoustic array that generally reproduces the capabilities available onboard an AUV; shows part of the anchoring system used to stabilize the array orientation. (b) Equipment loaded on an electric motorboat prior to deployment; shows the stand-alone array unit and the components that make up a traceable acoustic source node.

3.3.3 Collecting field measurements

The objective of building a classifier for environmental characterization is to eventually apply the classifier to field measurements, assessing how well the virtual-sampling sets and the real data align with each other. To this end, an experimental system was deployed in the Charles River Basin based on the concepts described earlier. Acoustic signals were transmitted from one vehicle, insonifying the environment; these signals were then recorded by a receiving array some distance away. The various components involved are shown in Fig. 3.9.

The receiving system consisted of a vehicle dummy rather than an actual AUV, given the availability of equipment and the overlap with additional research efforts that required similar data[19]. The substitute vehicle had a surface expression (paddle board) which supported the vehicle computer, data acquisition system and GPS receiver. In addition, the vehicle also had an underwater component consisting of the receiving array with its supporting structure, as well as rigging equipment (lines, floats and anchors) to orient the array and secure the system during the experiments.

[19] In addition to facilitating this environmental characterization work, the dataset collected during these field operations was also used as a starting point for research on Synthetic Aperture Sonar, as part of the Strategic Environmental Research and Development Program (SERDP).

The transmitting vehicle was chosen to be an electric motorboat given that one was available – this choice helped reduce ship noise in the data, compared to the other non-electric boats available. Its acoustic equipment included a Lubell Labs underwater speaker, as well as an audio controller to output a selection of waveforms through the speaker. The ship was also tracked with a Hemisphere GPS module to track its position.

3.3.4 Statistical performance

This environment classifier was initially designed by creating a training set from acoustic models and virtual sampling – a purely theoretical foundation. Its implementation, on the other hand, evaluates the training set against field measurements to produce a classification prediction. In real-time operations, this sort of classifier would produce an output from snapshots of data, and depending on the nature of the task, the most recent sequential predictions may be considered as a group to produce an additional confidence metric. Regardless of the design basis, the classifier's performance and reliability need to be evaluated before deploying it in the field; this can be done by taking a bootstrapping approach.

At a high level, bootstrapping refers to the process of sampling a dataset randomly and with replacement; a given sample may appear more than once in the resampled set. This new set is then evaluated against some algorithm or scoring metric, and the resampling and evaluation process is repeated multiple times to estimate the distribution of the scoring metric (the target statistic) across the original population, meaning the source from which the set of known measurements was collected. The conceptual schematic of information flow in the bootstrapping approach is presented in Figure 3.10.

The performance reported by the bootstrapping approach hinges mainly on three parameters: the number of random samples n taken from the original set of measurements (with replacement) to construct a test set; the number of neighbors k used by the k-NN classifier to produce a prediction;

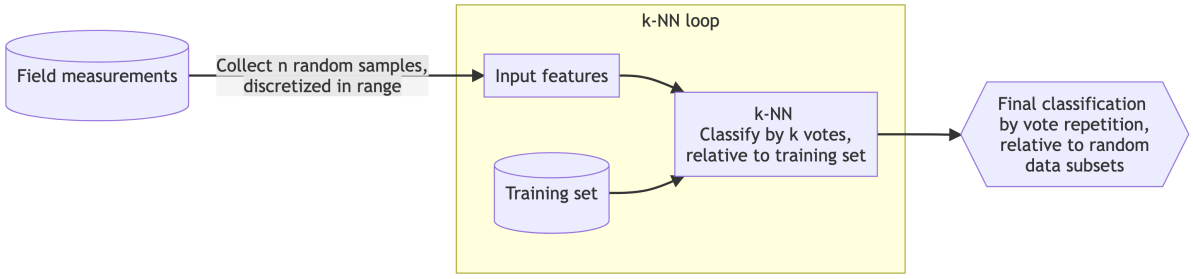


Figure 3.10: Schematic of the bootstrapping approach to determine the classifier performance. The solution consists of a data-driven voting system based on multiple random iterations of the k-NN classifier.

and the number of repetitions used in the bootstrapping process. Figure 3.11(a) illustrates a sample distribution of the classifier output across 1200 repetitions, where each iteration is based on a single neighbor, $k = 1$, and a population size of $n = 200$. A sample set from one of the 1200 repetitions is shown in Fig. 3.11(b).

There are two main observations to be made from Figure 3.11, with regards to this classifier's performance. The first is that the distribution of class outputs is strongly concentrated on what we expect the answer to be – we know from experience that the Charles River Basin features a muddy bottom, and this is the same overall solution identified by the bootstrapping exercise. The second observation is that the observation vector constructed from field measurements are not necessarily as smooth as the training samples; the fact that the acoustic features produced my the mud bottom model are visibly different from those of the sand and basalt models influences the ability of the resampling process to converge on the known solution.

Supplementary analysis

The uniqueness of the feature space directly impacts the performance of the classifier. More generally, machine learning techniques of this type generally aim to create some hyperspace (a trans-

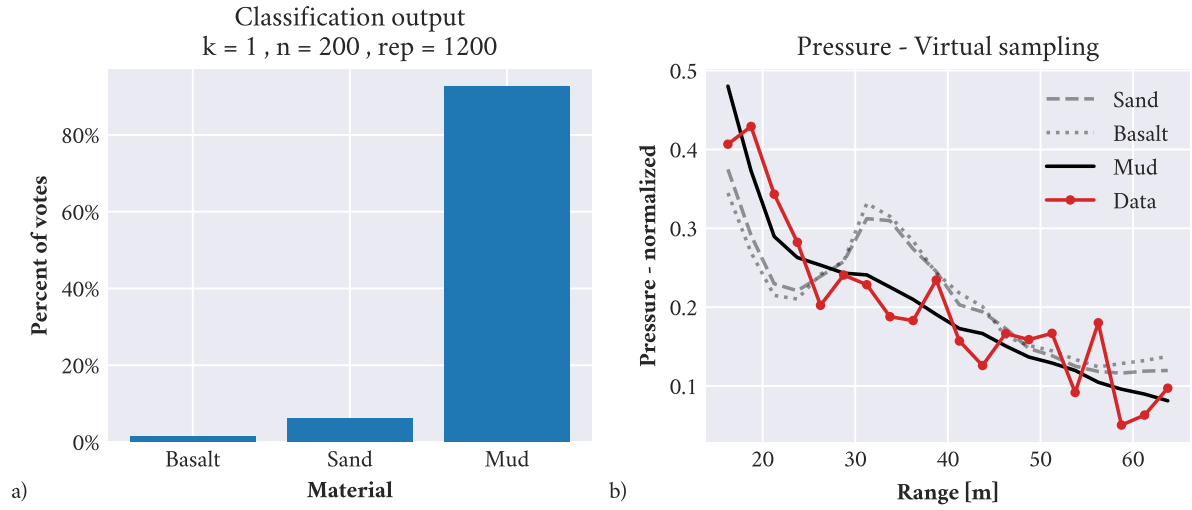


Figure 3.11: Performance of a k-NN classifier for riverbed material characterization. (a) Classification votes from 1200 repetitions with k=1 nearest neighbors and n=200 random data samples per trial. (b) Comparison of a random test vector built with n=200 samples from field measurements, versus one feature vector per class from the training set. The training set is assembled by virtual sampling from the acoustic models.

formation of the measurement or observation domain) that makes it possible to separate samples, such that a well-defined portion of the new space corresponds to each of the assigned labels. In Fig. 3.11(b), this could be associated to distinguishing between the presence or absence of a peak around the 30-40m range, for example.

As the features become less unique (there is more overlap in the data), this becomes an increasingly challenging exercise. This may be better illustrated by exploring the classifier's ability to evaluate two other parameters used in the acoustic models. Fig. 3.12 shows the same classifier, trained on variations of the mud model with different bathymetry. Fig. 3.13 likewise explores variations of the mud model, this time looking at estimating the density of the mud layer as the classification objective.

For both of these supplementary scenarios, a 3-point classifier is implemented; the reference value from the earlier material characterization trial lies in the central position, with positive and negative offsets from the reference values populating the other classes. These are implementations

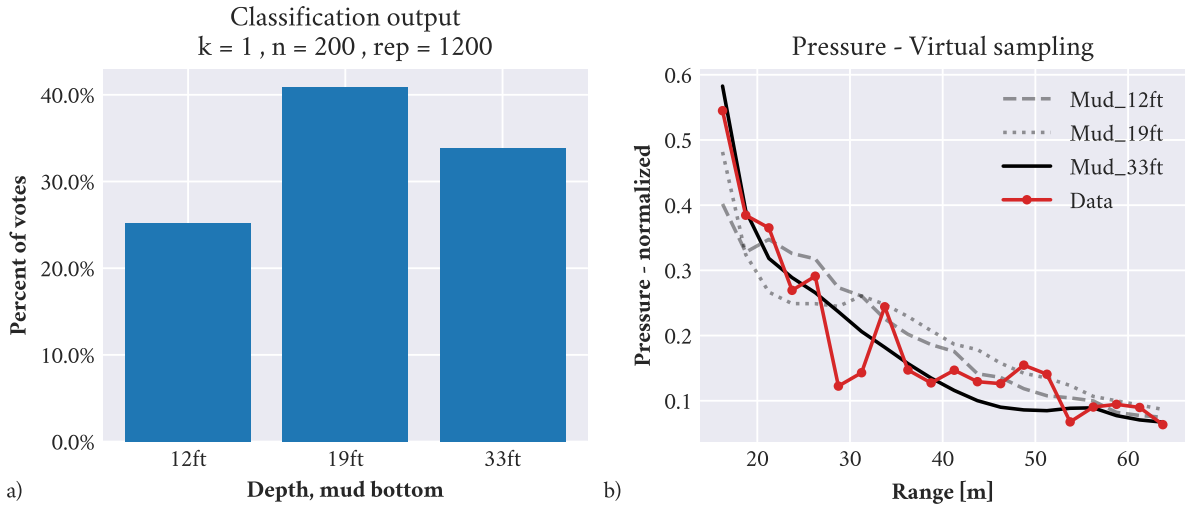


Figure 3.12: Performance of a k-NN classifier for bathymetry estimation. (a) Classification votes from 1200 repetitions with k=1 nearest neighbors and n=200 random data samples per trial. (b) Comparison of a random test vector built with n=200 samples from field measurements, versus one feature vector per class from the training set. The training set is assembled by virtual sampling from the acoustic models.

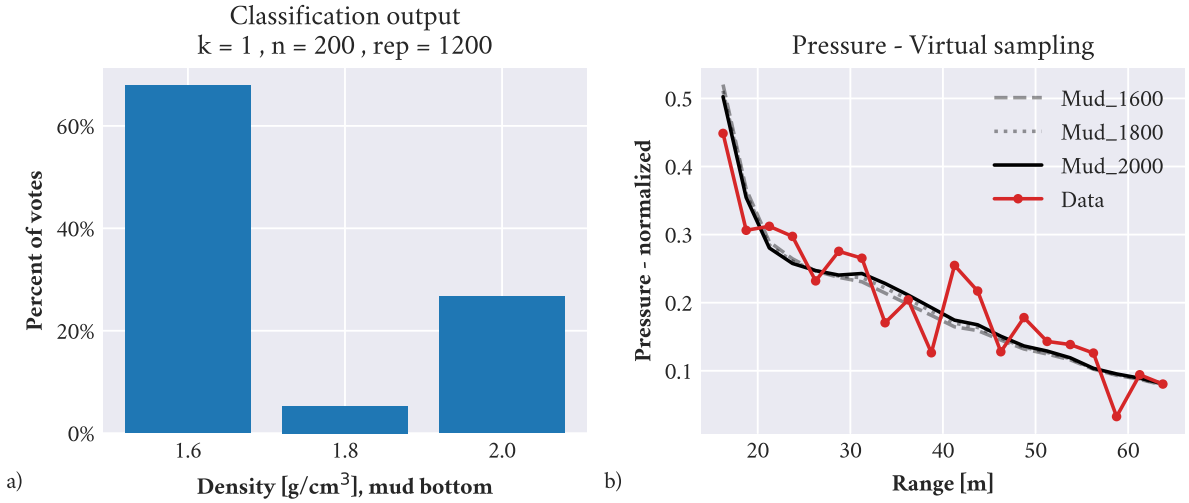


Figure 3.13: Performance of a k-NN classifier for riverbed density estimation. (a) Classification votes from 1200 repetitions with k=1 nearest neighbors and n=200 random data samples per trial. (b) Comparison of a random test vector built with n=200 samples from field measurements, versus one feature vector per class from the training set. The training set is assembled by virtual sampling from the acoustic models.

of a classifier rather a regression approach, even though the new targets are numerical values of a given parameter – an application comparable to that of the data discretization process, in which samples are assigned to the closest bin.

In the order presented, and with the deviations shown, these parameters introduce gradually smaller deviations from the reference model. The bathymetry models, configured to broadly capture the spread of depth measurements reported for the Charles River Basin, exhibit some small differences between the models, and the bootstrap distribution weakly supports the choice of reference value. The density models are configured with small deviations from the mud baseline, and the result is visually overlapping tracks for all models; the classifier begins to take on the extremes due to the variability captured in the resampling (bootstrap) process. Note that the difference in density between mud and sand, as given in Table 3.1, are closer than the chosen offset. This same process could be extended to other parameters given in the aforementioned table and used in the acoustic model, though the benefit of it really hinges on this idea: does the chosen feature space (translated to replica vectors by the acoustic model) reflect any helpful information for the classifier to exploit?

3.4 Summary

This chapter presented two projects that employ fundamental concepts of machine learning to explore the domain of information often available to Autonomous Underwater Vehicles. The first, a vehicle behavior classifier, was presented in the context of the Experiential Learning Theory and captured the benefits of expanded access to information collected from the acoustic environment. The learning framework was then used as a basis to illustrate the development process for the second project, drawing some distinctions between the first task and that of environmental characterization in terms of the information (samples and modeling constraints) available. Where the first project is informed mainly by design choices, the second relies on external information over which no direct

control can be exerted, conveying the fundamental challenge behind autonomous environmental adaptation efforts.

The behavior classification portion of this chapter was published in "Passive acoustic tracking for behavior mode classification between surface and underwater vehicles" and also presented in "Machine learning for behavior classification of passively tracked vessels". The environmental classification section was presented in "Estimation of the acoustic environment through machine learning techniques".

4 ICEX-20 : Experimental system

"You love the sea, Captain, don't you?"

"Yes, I love it! The sea is everything! It covers seven-tenths of the earth. Its breath is pure and wholesome. The sea is an immense desert where man is never alone, for he feels life pulsating all around him. The sea is nothing but the means which permits man to lead an almost supernatural existence; it is all movement and love. It is the living infinite, as one of your poets has said."

– Jules Verne, *20,000 Leagues Under the Sea*

The ICEX-20 experiment builds on many years of field experience and prior work, of which the ICEX-16 mentioned in Chapter 1 is but one example. Discussing the core systems that enabled this experiment – which is itself the first objective of this chapter – is an exercise of rediscovery and wonder Captain Nemo would be thrilled about. Building on this scientific precedent, the following sections then present simulation-based work that explored the feasibility of exploiting data already available onboard the AUV to enhance the vehicle's autonomy via machine learning techniques.

The core contributions of this chapter include (1) the preparation of the vehicle autonomy payload for field trials and the ICEX-20 deployments, including the integration of the most up-to-date LAMSS codebase with necessary kernel module modifications to enable legacy hardware interfaces; and (2) the exploration of environmental adaptation opportunities based on the information content available onboard the vehicle, both from a theoretical perspective and through hardware-in-

the-loop simulations.[1] The insights drawn from this simulation work are exploited later in this thesis, to improve on the performance of fielded algorithms.

4.1 Core systems

The core systems that made the ICEX-20 experiment possible in the first place include the Virtual Ocean Simulator and its hardware-in-the-loop (HITL) implementation, NETSIM. The Integrated Communications and Navigation Network (ICNN), which ultimately enabled the operational capabilities demonstrated throughout the experiment, was originally designed and tested in simulation using both the software-only and HITL implementations of the Virtual Ocean.

4.1.1 The Virtual Ocean Autonomy Testbed

The Virtual Ocean Autonomy Testbed (VOAT), or Virtual Ocean Simulator, is a collection of specialized applications assembled to work in unison in order to reproduce the effect of the physical ocean on acoustic signals. The toolkit includes programs for data assimilation from ocean models, such as the HYbrid Coordinate Ocean Model (HYCOM)[2] and MIT's Multidisciplinary Simulation, Estimation and Assimilation Systems (MSEAS).[3] In addition to ocean models, packages for acoustic propagation modeling are also included. The fundamental characteristic of the virtual ocean architecture is that it was designed to make the vehicle autonomy system agnostic to whether it was deployed in the field or in a simulation.[4]

[1] Viquez, Bhatt, Novitzky, and Schmidt, "Model-aided acoustic environment estimation via data fusion for autonomous underwater vehicles"; Bhatt, Viquez, Novitzky, and Schmidt, "An information theory approach to assess acoustic-environmental significance".

[2] Chassignet, Hurlburt, Smedstad, Halliwell, Hogan, Wallcraft, Baraille, and Bleck, "The HYCOM (HYbrid Coordinate Ocean Model) data assimilative system".

[3] Haley and Lermusiaux, "Multiscale two-way embedding schemes for free-surface primitive equations in the "Multidisciplinary Simulation, Estimation and Assimilation System"".

[4] Schneider and Schmidt, "NETSIM: A Realtime Virtual Ocean Hardware-in-the-loop Acoustic Modem Network Simulator".

At the heart of the system lies the following operational premise: every node in the system that has a need to know something about the environment runs its own, self-contained instance of the virtual ocean. Vehicle nodes, for example, run a copy of the virtual ocean regardless of whether they are launched in simulation or in the field (Figure 4.1). Each of these internal models is used to inform objective functions used by the corresponding vehicle's helm throughout the mission.

The fidelity of the virtual ocean is driven in part by the choice of active modules and configurations. The benefits of having this flexibility built into the system become apparent when looking at real-time operations such as the one illustrated in Figure 4.1. In largely controlled simulation conditions, access to powerful computers is certainly within reach, and using high-resolutions in runtime may appear to be a non-issue. When the environmental simulator is deployed onboard a real vehicle, however, computational limitations most often driven by a limited power budget can become a significant factor that impedes the use of the same high-resolution models.

The vehicle's need to understand its environment, of course, should be apparent: the internal model informs decisions related to the communication and navigation systems, and thus can have a sizeable impact on the autonomous performance of the vehicle. For example, an operational requirement stating that the vehicle ought to maintain a viable communication link available at all times would be all but impossible to meet without supplying the vehicle with a suitable environmental model.

Within LAMSS, the standard implementation of the Virtual Ocean Simulator uses the ray-tracing program *BELLHOP* along with data from the ocean models to produce element-level time series data, capturing both signal and noise. In order to achieve an appropriate balance between performance and computational load, the virtual ocean uses a nested modeling approach which benefits from the use of pre-computation by exploiting the timescales of change for the different features. The ocean models, which serve as the starting point for the framework, generally vary slowly and are thus sampled at relatively long intervals. The data from the ocean models is then used as an

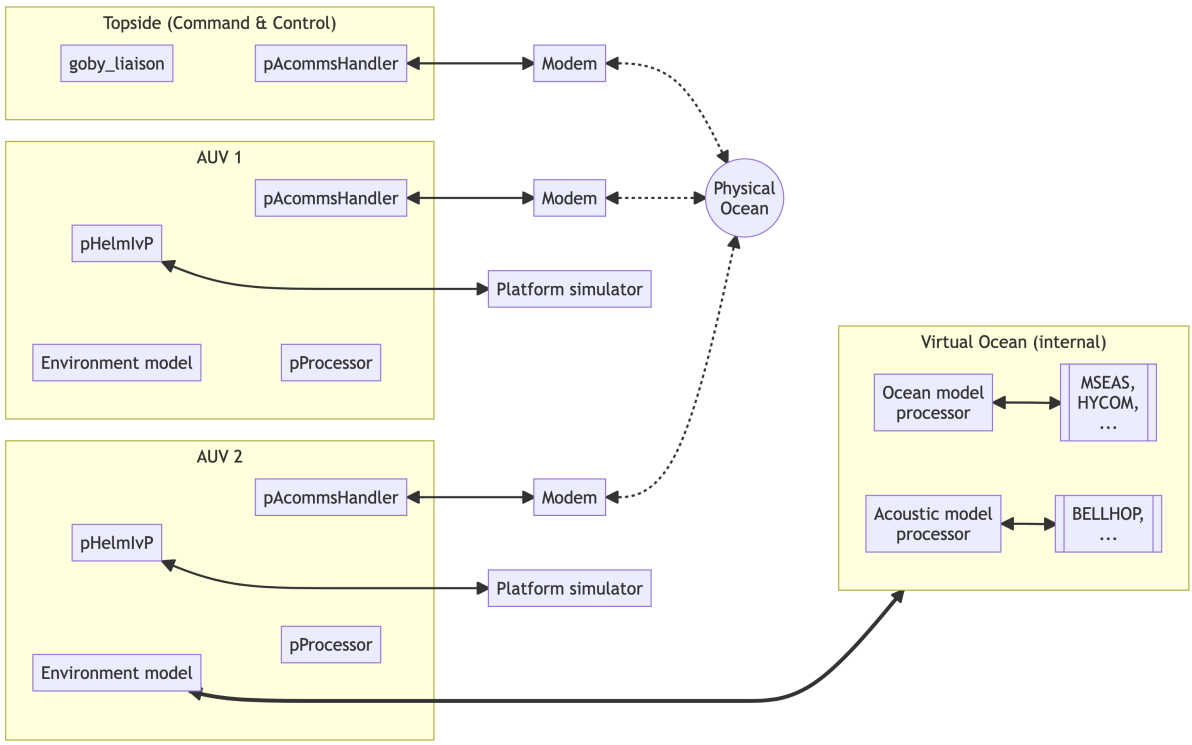


Figure 4.1: The Virtual Ocean Simulator runs in simulation and field deployments alike. In the field, real vehicles rely on a lower-fidelity version of the simulator to inform decisions that may be impacted by the environmental conditions.

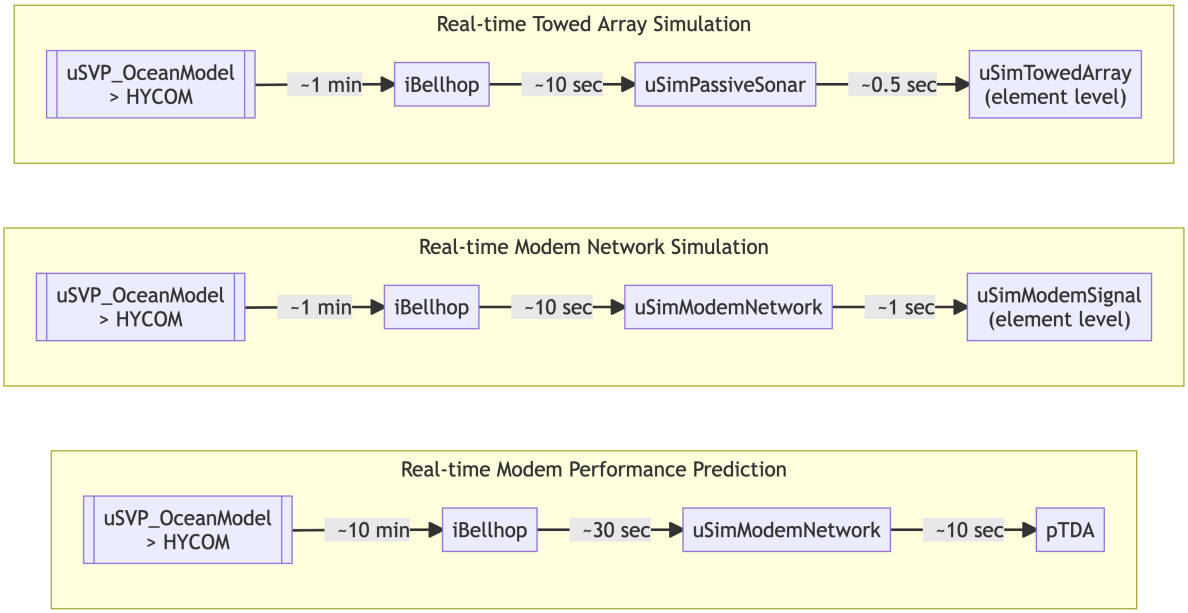


Figure 4.2: The Virtual Ocean Simulator uses a nested modeling approach that exploits the timescales of change of the different features. Slow-changing features such as those reported by the different ocean models are sampled less frequently, while fast-changing features such as the element-level signals (which change as the vehicle travels) are sampled much more frequently.

input for the acoustic modeling code, which produces a local grid of arrival structures in the scale of the vehicle's motion.

This grid of arrival structures serves as an intermediate point from which the element-level signals are computed by local plane wave expansion. Thus, the local grid of arrival structures is recomputed at shorter intervals than the ocean model updates, and the grid is configured to be large enough to provide sufficient information for multiple cycles of the element-level computation, which is the fastest-changing feature in the set. In addition to reflecting the physical rates of change, these timescales are also partly informed by the particular application for which they are being used. Exemplary uses of the virtual ocean are shown in Figure 4.2, along with typical timescale configurations for those applications.

4.1.2 NETSIM : Hardware-In-The-Loop simulation

A notable difference between field deployments (Fig. 4.1) and simulation is that the latter requires separate instances of the virtual ocean for the sensor simulator and the modem interfaces (Fig. 4.3), in addition to the ones used for each vehicle's internal models. This requirement stems from the intent to make the vehicle autonomy code entirely agnostic to the type of run (simulation vs field deployment).

Each instance of the virtual ocean can be configured independently. This makes it possible to evaluate the impact of a mismatched environment model, for example. In single-computer, full system simulation, the number of processes required often means that a lower-fidelity representation is preferred – a need that is covered by a local implementation of the necessary modem interfaces through `netsim_udp`.

NETSIM itself, the hardware-in-the-loop implementation of the Virtual Ocean Simulator, introduces physical modem interfaces and further separates the different instances of the virtual ocean in the simulation framework. The physical modems are assigned to the simulated nodes; given enough capacity in terms of audio ports and modem units, the system can be scaled to any number of nodes. Figure 4.4 illustrates the combined hardware and software components of NETSIM as presented by Schneider and Schmidt ("NETSIM: A Realtime Virtual Ocean Hardware-in-the-loop Acoustic Modem Network Simulator"), with a two-vehicle configuration shown in the diagram.

Upon a transmission command from the autonomy system on any one node, the assigned modem prepares and emits the audio signal on its audio channel, which in field operations would normally be connected to a transducer (via the appropriate power amplifier). The transmitting modem's signal is picked up by a sound card on the audio server. This computer then uses the element-level impulse responses from the virtual ocean to transform the audio input into a high-fidelity approximation of the signal received by each of the remaining nodes. Spatial Doppler effects are captured in the signal transform by convolving each of the multi-path arrivals in the impulse response with

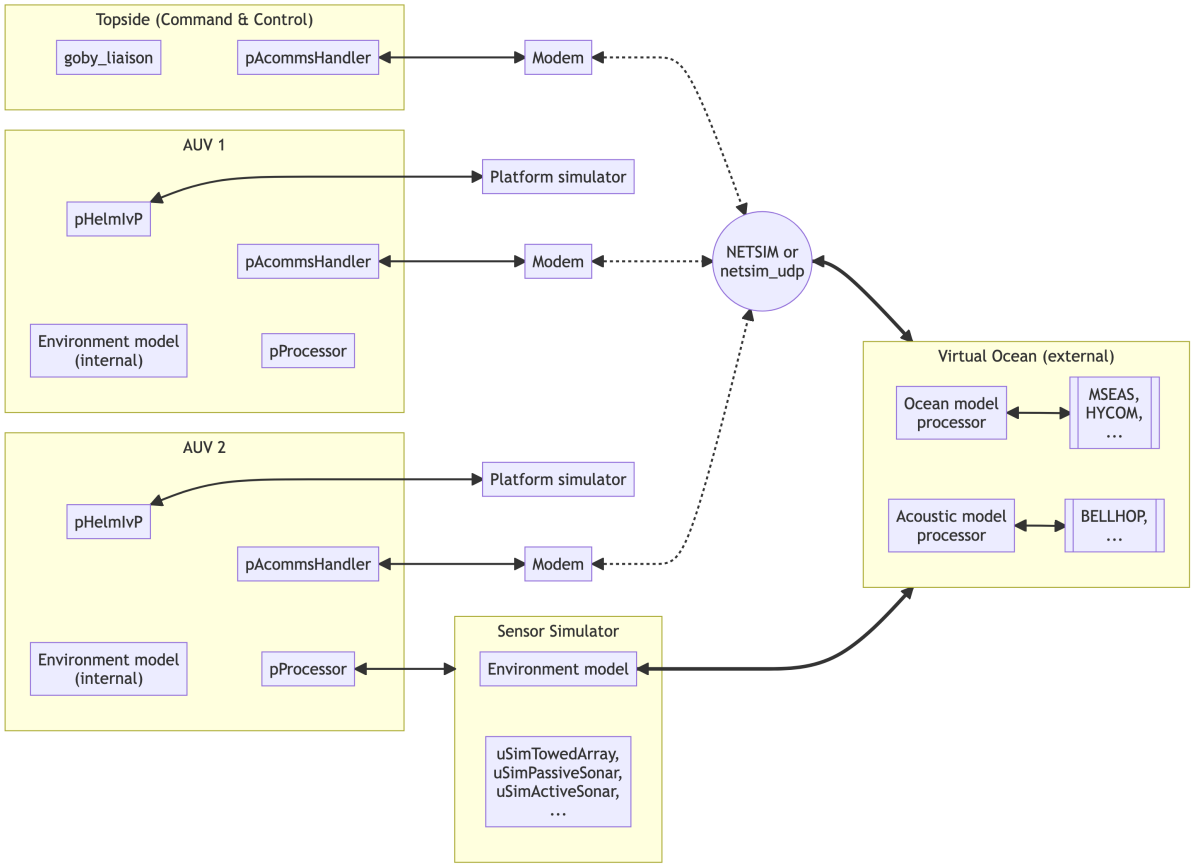


Figure 4.3: NETSIM expands on the Virtual Ocean Simulator by using an instance of the simulator running a high-fidelity ocean model as a substitute for the physical ocean. Software-based and hardware-in-the-loop implementations are available.

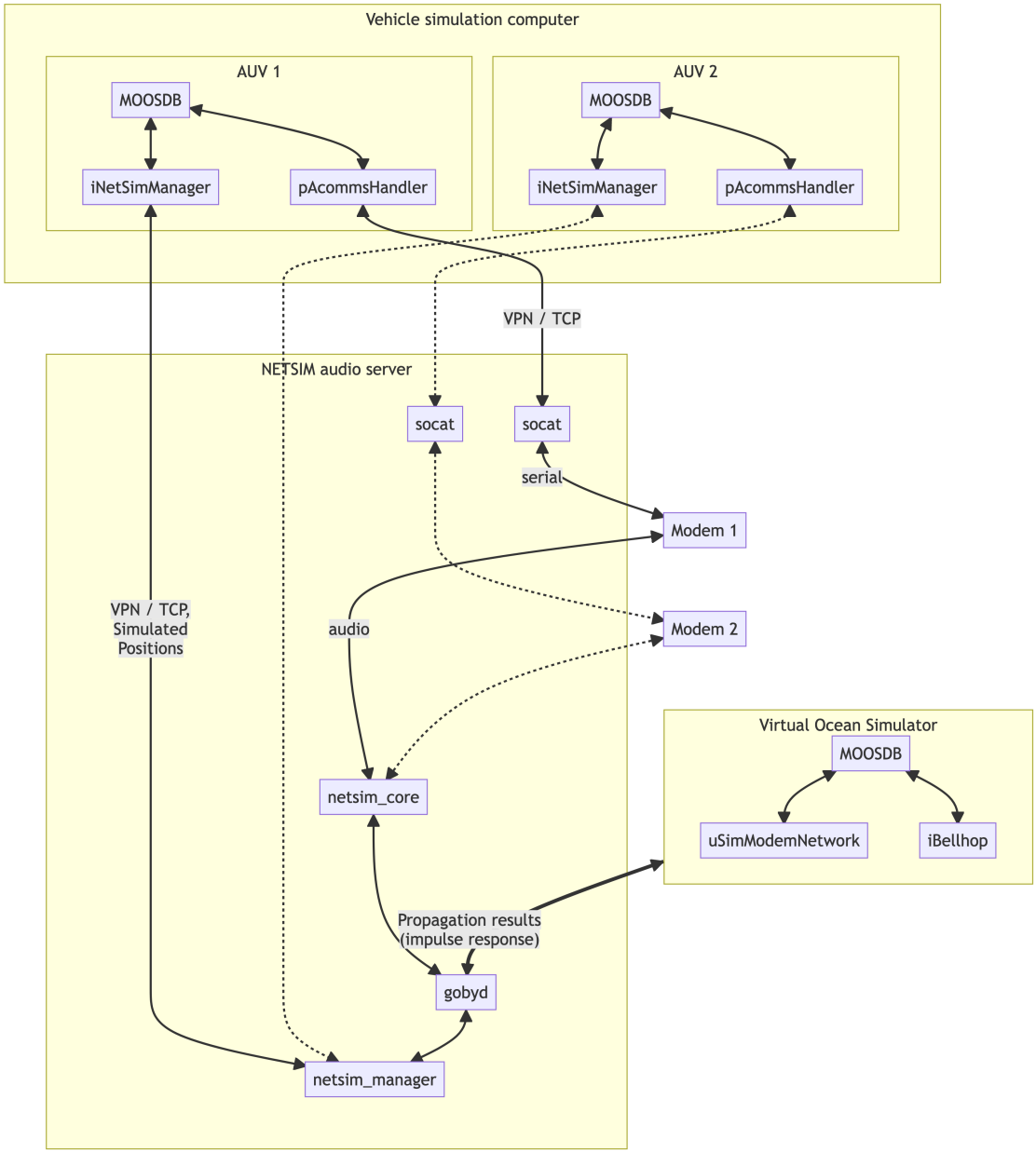


Figure 4.4: NETSIM introduces physical modems as the key hardware-in-the-loop component.

the associated Doppler-shifted replica. Ambient noise can also be injected to the signal at this stage. The modified signals are then relayed to the physical modems according to their node association on their corresponding input channel.

4.1.3 ICNN : The Integrated Communications and Navigation Network

The Virtual Ocean Autonomy Testbed and its hardware-in-the-loop implementation, NETSIM, were instrumental to the success of ICEX-20 deployments. Even from the early design and development stages, the virtual environment enabled continuous simulation-based testing and validation of the many moving pieces involved in the experiment, facilitating risk mitigation efforts along the way. This approach has a close parallel in planetary exploration, where repeat deployments aren't always possible. In space and under the polar ice, a reliable understanding of risks and failure modes is critical to the success of a mission, and is often obtained from iterative simulations with high-fidelity models.

One of the ICEX-20 development pipelines that directly benefited from the VOAT was that of the Integrated Communications and Navigation Network, or ICNN. From a high-level perspective, the ICNN is a specialized implementation of the Long Baseline (LBL) solution discussed in Section 2.4.2. Indeed, there are other systems that expand on conventional LBL, including commercial offerings from companies such as iXblue; but these improvements are often based on more complex tracking of the position solutions given by the same simplified linear model given in Section 2.4. The ICNN stands out from conventional LBL configurations in that the time-of-flight to range conversions are directly aided by the acoustic propagation and ocean dynamics models in the VOAT,[5] rather than relying on the simplified linear model.

[5] Randeni, Schneider, and Schmidt, "Construction of a high-resolution under-ice AUV navigation framework using a multidisciplinary virtual environment"; Schneider, Schmidt, and Randeni, "Self-Adapting Under-Ice Integrated Communications and Navigation Network".

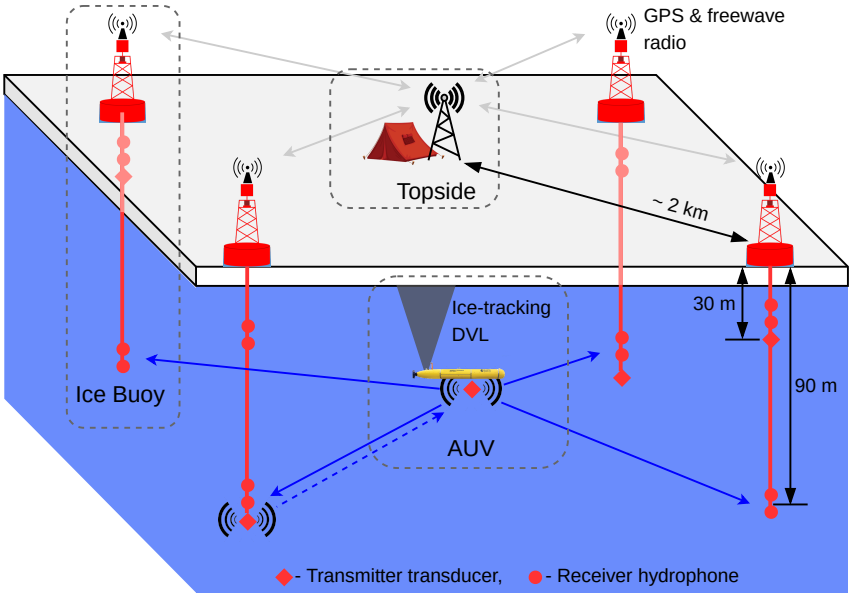


Figure 4.5: Overview of the Integrated Communications and Navigation Network (ICNN).

As deployed for ICEX-20, the hardware portion of the ICNN consisted of 4 ice-tethered buoys deployed in a 1.5-2 km radius from ice camp Seadragon. Each buoy was equipped with its own acoustic modem, as well as two input hydrophone layers, at depths of 30 m and 90 m. They were also equipped with a single output channel each, such that half of the buoys capable of transmitting at each of the aforementioned depths. In addition to the acoustic hardware, each of the buoys also included a GPS receiver and a Freewave radio to communicate with the operations center. This system overview is illustrated in Figure 4.5.[6]

4.1.4 AUV *Macrura* : Autonomy Payload

In discussing the core systems that enabled the ICEX-20 experimental work, it should be apparent that the vehicle platform itself was indeed an essential component of the deployments. Under the payload autonomy scheme,[7] the general architecture of AUV *Macrura* can be represented as in Fig.

[6] Randeni, Schneider, and Schmidt, "Construction of a high-resolution under-ice AUV navigation framework using a multidisciplinary virtual environment".

[7] Benjamin, Schmidt, Newman, and Leonard, "Nested autonomy for unmanned marine vehicles with MOOS-IvP".

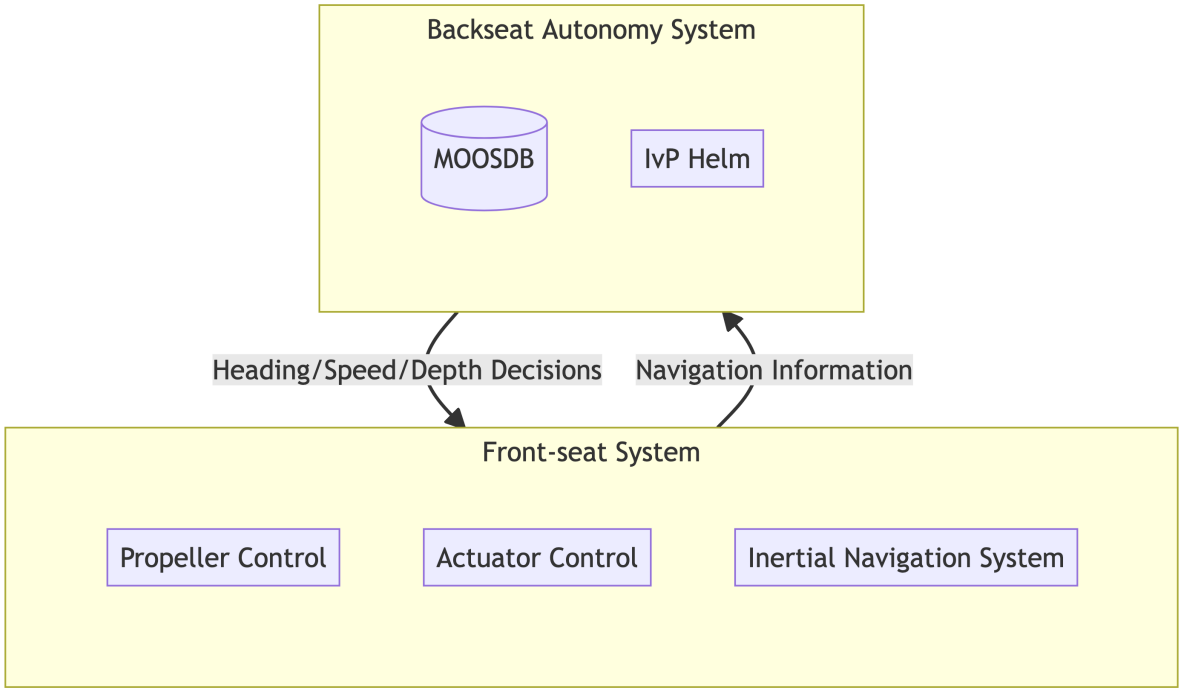


Figure 4.6: Overview of the system architecture for a payload autonomy implementation.

4.6, where the front-seat system is responsible for the low-level controls and any platform-specific sensor processing. The payload system ingests navigation information and any other sensor data relayed by the front-seat, and exploits the available information to make higher-level decisions about how to meet the vehicle's mission objectives. The payload's decisions are then relayed back to the front-seat in the form of targets for its controllers to handle – thus, this scheme is also referred to as a backseat autonomy system, in reference to its role as a backseat driver.

The remainder of this section focuses specifically on the payload autonomy system (backseat), which the author of this thesis was responsible for during the span of ICEX-20 preparations and field operations (Fig. 4.7).

At a high level, preparations involved reconstruction of the hardware and software stacks. In the software layer, this involved a clean installation of the entire system as well as implementing custom configurations and conducting validation of all computers with their fresh, up-to-date op-

a) b)

Figure 4.7: Payload Autonomy stack for AUV *Macrura*. (a) Side view of the distributed computing assembly that enables *Macrura*'s autonomy. (b) The author of this thesis, working on the payload in preparation for field testing ahead of ICEX-20.

erating systems and the lab's latest codebase. Diagnosing issues in the physical interface layer, and modification of custom kernel modules used as part of the payload's data acquisition system were all necessary steps in enabling the vehicle to enter the field once again.

In its latest iteration (as deployed for ICEX-20), *Macrura*'s payload autonomy consists of a distributed computing system with three processing units. The payload also includes networking hardware, data acquisition systems and additional sensor modules. While navigation sensors on the AUV are generally handled by the front-seat system, acoustic and environmental sensors are managed directly by the payload module.

The three processing units are divided into the following functions: a central autonomy unit, a homing unit, and a machine learning module. The homing unit was originally added to the payload in preparation for the ICEX-16 experiments. It was implemented as a specialized module with the sole purpose of ingesting and processing acoustic data from a nose-mounted tetrahedral array, to produce estimated bearing and range to a known beacon.[8] The machine learning module was

[8] Rypkema, "Underwater & Out of Sight: Towards Ubiquity in Underwater Robotics".

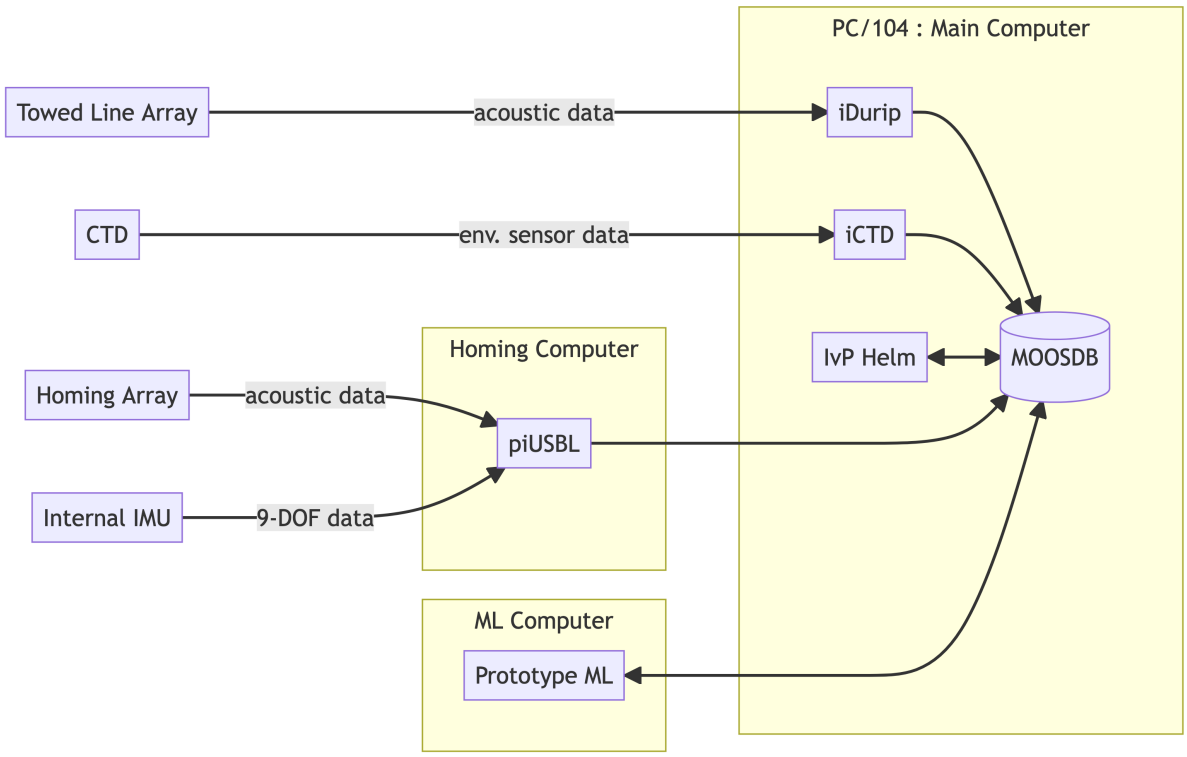


Figure 4.8: Overview of the internal components in the payload autonomy module.

added in preparation for the ICEX-20 experiments, to expand the payload's capabilities for parallel processing tasks used in ongoing research efforts to implement machine learning. This distributed computing setup is illustrated in Fig. 4.8.

4.2 Observable information from the vehicle perspective

The core idea behind model-aided estimation is to use the information provided by some mathematical construct to constrain how a system exploits a set of measurements to inform subsequent predictions about a function's values. For the positioning problem (Sec. 2.4), real-time applications often rely on the assumption of a uniform sound speed in the conversion between travel time and range – that is, they typically use a linear model as a simple but sufficient approximation of the

real world. Tomographic inversion (Sec. 2.3) may use a reduced-order representation to tackle the problem of estimating the environment based on samples between nodes at known locations; the choice of basis functions, then, capture the essence of the model. Although these two applications typically take their counterpart to have a known solution, the environmental modeling and positioning tasks are in fact closely related in the absence of a known ground truth for either of them. Additionally, both tasks have the following in common: the sound speed profile and travel time domains are both of interest to them, either as a basis for subsequent simplifications or as a baseline from which deviations are evaluated.

The following sections revisit the sound speed and acoustic domains, which were first introduced in Chapter 2, to illustrate how various data sources and models were used to inform the development work that preceded the ICEX-20 field operations.

4.2.1 The Sound Speed Domain

The localized sound speed value at any point in the ocean is a complicated function of other physical parameters, such as temperature, salinity and pressure or depth (Sec. 2.1). Moreover, the impact of any such value goes well beyond its particular coordinates; even under the simplified assumption of a range-independent model, the depth-dependent variability of the sound speed profile across the water column directly impacts the acoustic environment – it creates shadow zones and convergence zones that impact navigation and communication, as was first illustrated in Chapter 1, Figure 1.2.

As operated within the LAMSS autonomy paradigm, there are two layers of interest to the vehicle: the real world, or physical layer; and the simulated or modeled world, which constitutes the virtual layer. In the context of local sound speed sampling, these two layers can be described as follows:

- **Physical layer**

  The vehicle has the ability to sample the real ocean directly, using its onboard CTD sensor. The measurements collected by the vehicle can be interpreted using one of the equations

given in Sec. 2.1, and provide a direct – albeit incomplete – observation of the sound speed profile. The key limitation at play here is that the vehicle's specifications may establish a depth limit that would impede sampling the SSP at depths of interest for the acoustic modeling exercise.

- **Virtual layer**

  As was discussed earlier with respect to NETSIM, the vehicle benefits from its own instance of the Virtual Ocean simulator, which is used to inform navigation decisions relative to mission objectives. By assessing performance in the virtual ocean, the vehicle can maintain or establish a viable communication link at any point during the mission, for example. The virtual layer of the sound speed domain is itself built around some combination of (1) historical benchmark profiles, (2) oceanographic models (MSEAS, HYCOM), and (3) field measurements from past and ongoing missions – the latter includes data from the WHOI Ice Tethered Profiler (ITP) project.[9]

In addition to the aforementioned information layers, which center on the vehicle's independent operation, the AUV can also access an abstraction layer through inter-node communication. As the throughput of the acoustic link is limited, a set of basis functions such as the Empirical Orthogonal Functions (EOFs) mentioned in Sec. 2.3.1 can be used to compress the measurements collected by one node or vehicle into a weight-vector representation. In fact, this collaboration framework was at the center of ICEX-20 preparations: as part of the planned field demonstrations, environmental adaptation would be tackled by capturing the SSP through local CTD casts launched from the ice camp and fitting the samples with a reduced-order representation based on EOFs (Fig. 4.9). The resulting weights would then be packaged into an environmental update message, and relayed to

[9] Krishfield, Toole, Proshutinsky, and Timmermans, "Automated Ice-Tethered Profilers for Seawater Observations under Pack Ice in All Seasons"; Toole, Krishfield, Timmermans, and Proshutinsky, "The Ice-Tethered Profiler: Argo of the Arctic".

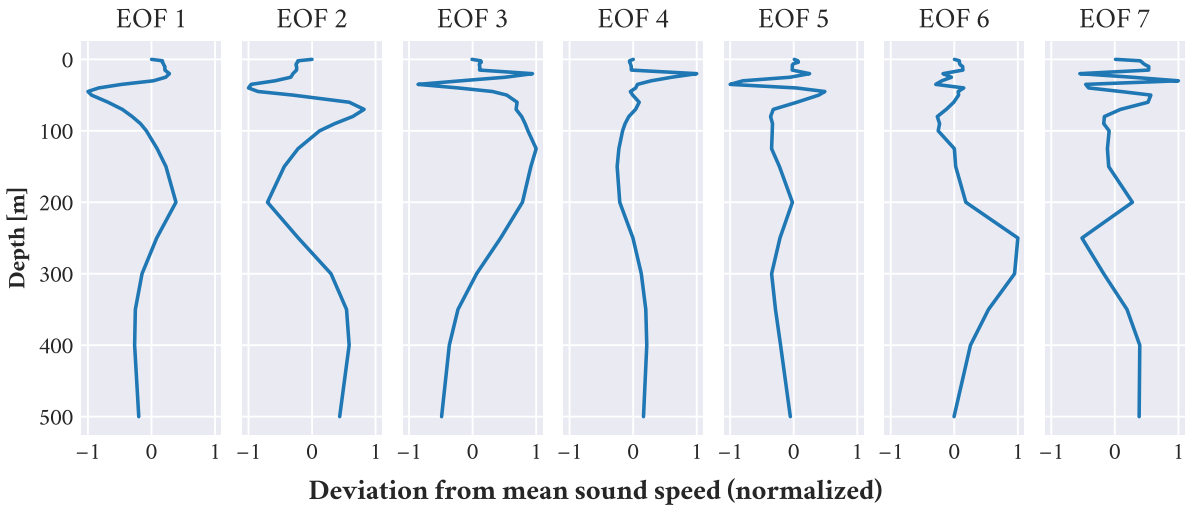


Figure 4.9: The set of Empirical Orthogonal Functions (EOFs) used as the basis for data compression during ICEX-20 field operations. These EOFs were computed from a subset of WHOI ITP data filtered by proximity to the operations region and the time of year.

the vehicle so it could update its internal ocean model while underway, based on the compressed representation of measurements collected at camp.

4.2.2 The Acoustic Domain

As has been discussed throughout this text, the sound speed domain has a direct impact on the acoustic environment, determining propagation modes or paths. In high-fidelity applications, 3D models may capture complex features of a signal's propagation through space. Even in simplified, range-independent applications, the complexity of the depth-dependent sound speed profile can lead to pronounced shadow zones. This dependency of acoustic propagation models on the sound speed profile was introduced in Sec. 2.2. As before, the vehicle's perspective of the acoustic environment comes in two layers. The physical layer corresponds to the data collected by the vehicle in the physical ocean, and the virtual layer is composed of the acoustic propagation models.

In applications such as target detection and tracking, the vehicle's recordings are often considered a key component of the data product. These can be analyzed in real time or in post-processing,

to detect patterns of interest such as marine mammals or other vehicles. In acoustic communications, on the other hand, the raw data is not always available to the vehicle at large. Instead, acoustic modems such as the WHOI Micromodem (the modem used by MIT LAMSS) may report only the decoded data, if any, along with signal reception statistics and other metrics of interest. The WHOI Micromodem sends a known chirp ahead of any data packets, such that the receiver node can identify an incoming message. The leading chirp is also used to produce a short-term impulse response estimate (IRE), which the modem uses to unpack the message contents of later segments of the signal.

As a counterpoint to real-world measurements, one application of the acoustic modeling work that constitutes this domain's virtual layer is to predict how the transmitted signal is transformed into the data recorded by the receiving element. To predict, in other words, what the impulse response should look like and how the modem's short-term IRE, which is based on a limited observation window, may impact its ability to process incoming signals.

4.3 Simulation work

As essential part of the preparations for the ICEX-20 field experiments consisted of performing extensive simulation work to ensure the robustness of the various algorithms deployed in the Arctic. With respect to the challenge of environmental adaptation, these simulations also provided valuable insight into the alignment between the physical and virtual layers described earlier. This section discusses the essential insights from said simulation-based work.

To tackle the relation between these two domains, let us first recall Fig. 2.4, which illustrated multi-path arrivals detected by a receiver. The paths shown were due to boundary interactions in an otherwise simple environment with a uniform sound speed profile, and the relation between distance traveled and time of travel was therefore also linear. Expanding on this basic intuition, Figure

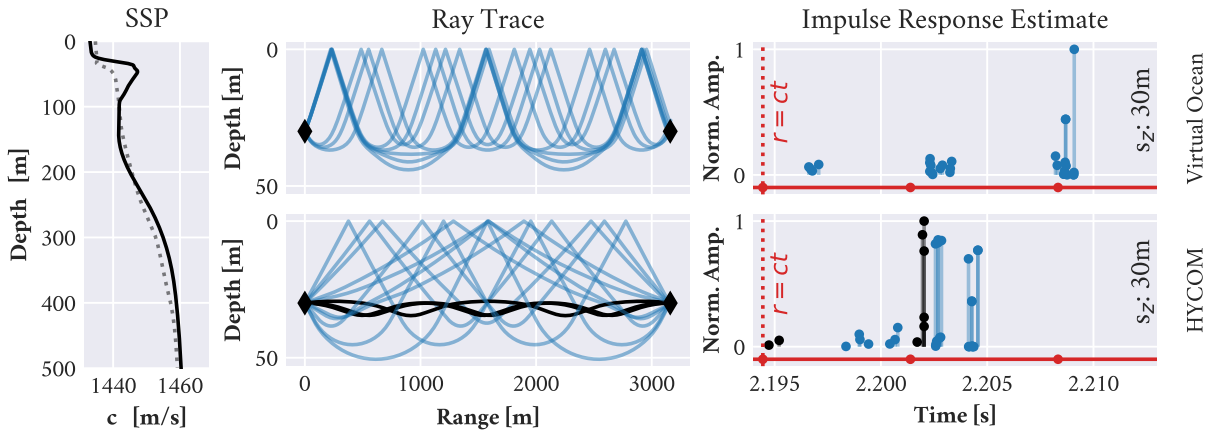


Figure 4.10: Effect of sound speed model on the ray tracing output (middle) and corresponding impulse response estimate (right) used as a basis for ranging. Sound speed profiles are taken from HYCOM (left, dashed) and the Virtual Ocean framework (left, solid).

4.10 evaluates two different sound speed profiles associated with the Arctic Ocean – one from HYCOM[10], and one from the Virtual Ocean framework, based on a sample set of field measurements.

The HYCOM sound speed profile leads the propagation model to predict direct path arrivals – these rays are shown in the middle bottom plot, in black. In contrast, the Virtual Ocean sound speed model, which is informed by field measurements, indicates that any detected signals will involve surface interactions; rays with boundary interactions are shown in the middle plots, in blue. The impulse response estimates (right) are composed from the amplitude and travel time associated with each of the rays (middle) upon arrival at the receiver coordinates. The IREs are compared with a simplified $r = ct$ ranging approach using $c = 1440$ m/s, which corresponds to the value at the source depth. Time steps equivalent to 10 meters each are also shown along the timeline (red dots), relative to the $r = ct$ prediction, to illustrate the error incurred by the linear model when compared to the acoustic propagation models.

As was discussed in Chapter 2, the ocean acoustic tomography problem seeks to estimate the sound speed structure in a volume of water from a limited set of travel time measurements. One of the considerations that should stand out from the discussion on tomographic inversion as presented

[10] Recall that HYCOM stands for the HYbrid Coordinate Ocean Model.

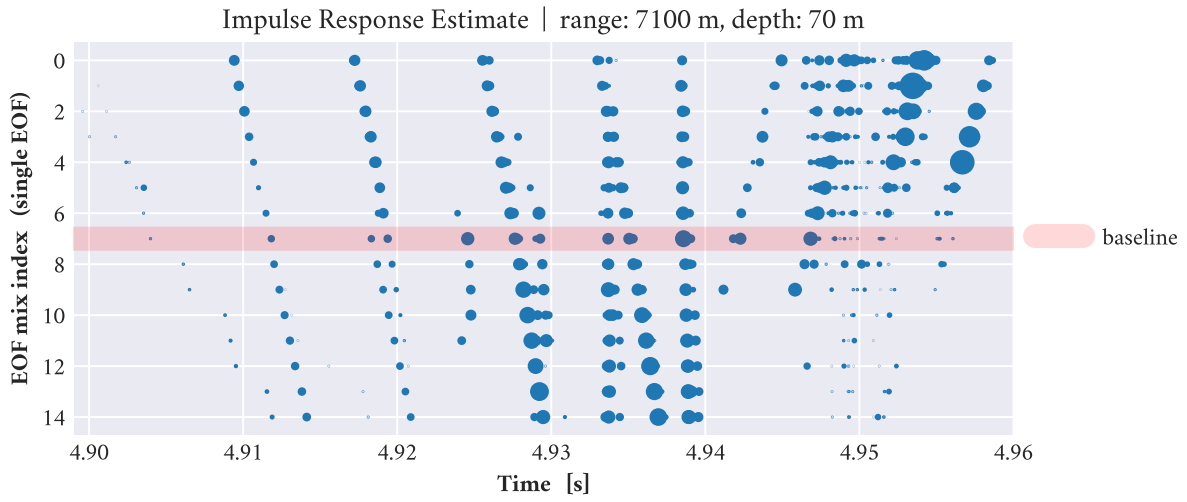


Figure 4.11: Exploration of the impulse response estimate based on the weight of a single EOF. The weight is set to a value between -5 and 5, for a total of 15 evenly spaced entries.

in Section 2.3.1, is that the measurement matrix C may be based on an initial reference model. The *a priori* information about the environment is used to compute baseline eigenrays with their corresponding travel times; the field measurements are then compared with the modeled results, and the deviations are used to estimate the weights for the basis functions. However, the process is sensitive to the correct identification of the eigenrays that should theoretically correspond with each of the arrivals.

Given the challenges of the eigenray identification process, the simulations can be used to explore the effect of different weights in the EOF representation space before actually looking at the inversion problem itself. To this end, Figure 4.11 takes a single basis function from the set previously shown in Fig. 4.9, and shows the impulse response estimate with respect to the chosen weight. The weights are reported by index (first index is zero) rather than value, with values spanning from -5 to 5 for a total of 15 entries. The baseline (index 7 in this case) corresponds to a zero-valued weight, meaning there is no deviation from the reference profile. The IREs are shown with marker size as a proxy for amplitude, rather than using height as in the IREs from Fig. 4.10.

The changes produced by adjusting the mixing parameter of a single EOF are sufficient to illustrate the complexities of eigenray detection at a high level. The set of rays arriving just before 4.94 seconds, for example, appear to be largely unaffected by changes to the EOF weight. The sets arriving ahead of the 4.93 seconds mark appear to exhibit a linear relation between change in time of travel and the change in weight. The paths arriving after 4.94 seconds appear to fade out as the chosen weight becomes positive (index larger than 7).

Building on its predecessor, Figure 4.12 illustrates the IRE changing with respect to an additional shape from the set of basis functions. As before, each weight is set to a value of -5 to 5, for a total of 15 entries each. The 225 combinations are ordered by setting both weights to their first (lowest) value, and then scanning first across all weight values for one of the EOFs – this is illustrated as a fine-step, with IREs of the same color. After reaching the end of the scan along the first shape's weights, the corresponding iterator is reset and the second iterator is stepped forward, creating the coarse step illustrated by the change of color in the IRE markers. The baseline is shown by the midpoint mixture index, where both weights are zero-valued. By scanning across two EOFs now, an additional challenge to the eigenray identification task becomes apparent: the effect of changes in the environment may be such that the order of arrival of the different paths may change. This is illustrated by the arrivals near the 4.93 seconds mark for the higher-valued weight mixture indices.

In addition to considering what can be learned from the models alone, as with the previous 1-EOF and 2-EOF combinations, a "through-the-sensor" scheme for environmental adaptation also calls for a discussion on how the model space and the sensor space align with one another. In this regard, a high-level understanding of the inner workings of the acoustic modem come into play to guide further development. This can be illustrated with the WHOI Micromodems used by LAMSS for ICEX-20: as was stated earlier, the Micromodems begin every transmission with a predetermined chirp, depending on the configured carrier frequency.

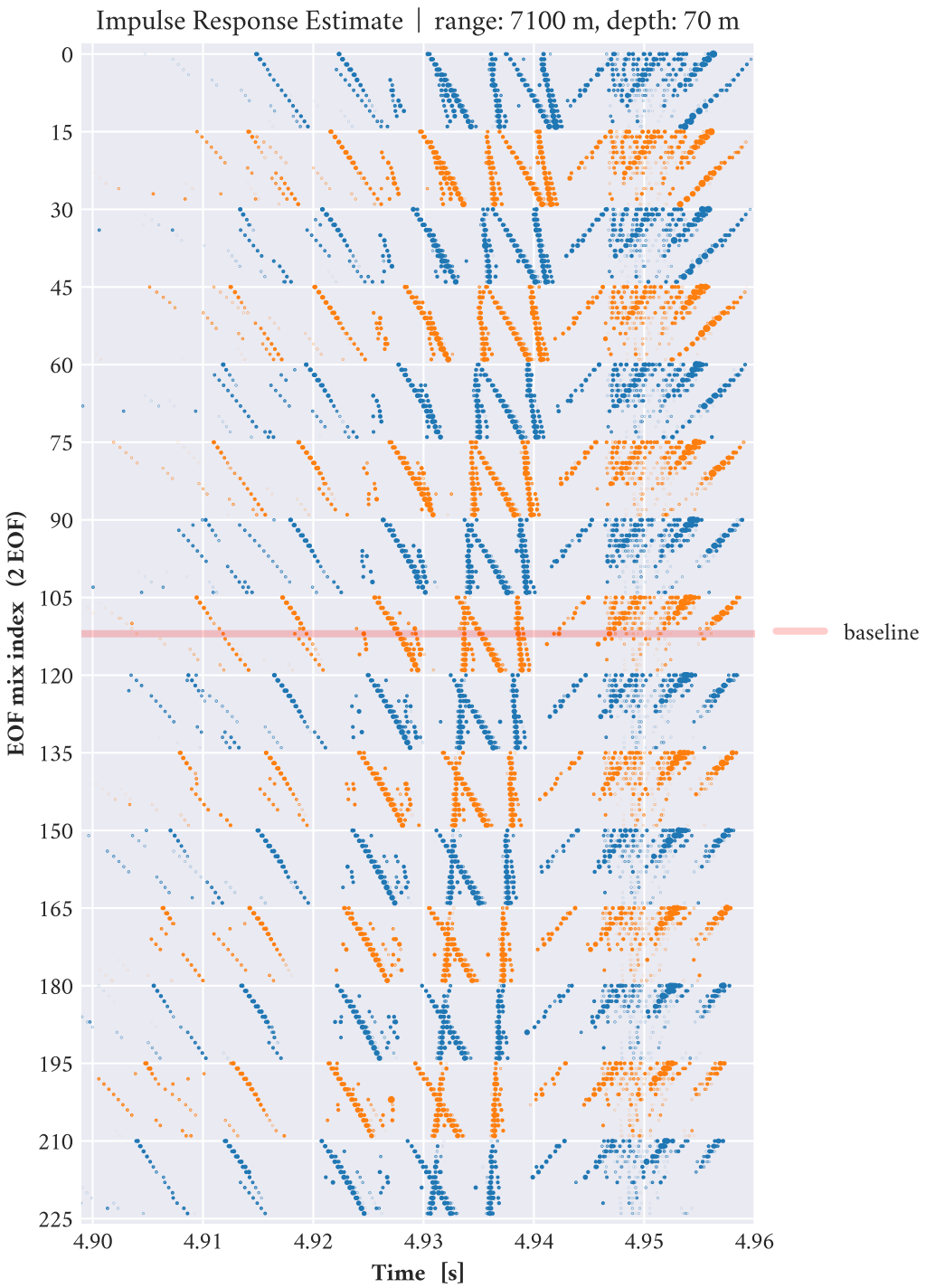


Figure 4.12: Exploration of the impulse response estimate based on the mixture of 2 EOFs. As in the single-EOF example, each weight is set to a value between -5 to 5, for a total of 15 evenly spaced entries each. The 225 combinations are ordered by setting both weights to their first (lowest) value, then scanning across one weight's values as a fine step (same color), and across the other weight's values as a coarse step (change of color).

The leading portion of the signal is used by the receiving modem unit as part of the detection stage. The receiving modem can attempt to deconvolve the recorded signal by using replicas of the expected chirps, in order to produce estimates of the impulse response. The IRE produced by the modem reflects whether the incoming signal contains the expected pattern. If the result of this matched filter exceeds a configurable detection threshold, the modem reports the detection and moves into its reception stage, in which the IRE is used as a key to clean up the signal through deconvolution before the modem tries to decode the data packets contained therein.

While the modem's impulse response estimate is used to make sense of the data packets upon a successful detection, one of the limitations it introduces is that it is based on a rather short observation window. For the ICEX-20 configuration, for example, the modem was set to operate at a carrier frequency of 10kHz, and a bandwidth of 5kHz. The expected IREs for this configuration are limited to 10 milliseconds, with sample points spaced as per Eq. 4.1.

$$\Delta t = \frac{1}{2\,(\mathrm{BW0})} = \frac{1}{2\,(5000\mathrm{Hz})} = 0.1\mathrm{msec} \tag{4.1}$$

In addition to the estimated impulse response, the WHOI Micromodems can report the time of arrival with respect to an internal or external time reference, based on the initial detection. However, in light of the challenges associated with eigenray detection, one potential approach to comparing model and field data is to enforce an amplitude threshold on the simulated set, to better replicate the process undertaken by the real hardware. Figure 4.13 shows IREs for three different environments, shifted from travel time to a relative scale zeroed at the time of detection. Here, the single-point travel time and amplitude values are convolved with a gaussian kernel to approximate the blending of neighboring arrivals.

The three environments illustrate the anticipated operational scheme: a baseline model and an adapted model for the vehicle, as well as a higher-fidelity model deployed on the virtual ocean simulator. The reference model consisted of the ICEX-16 profile, which was deployed on NETSIM.

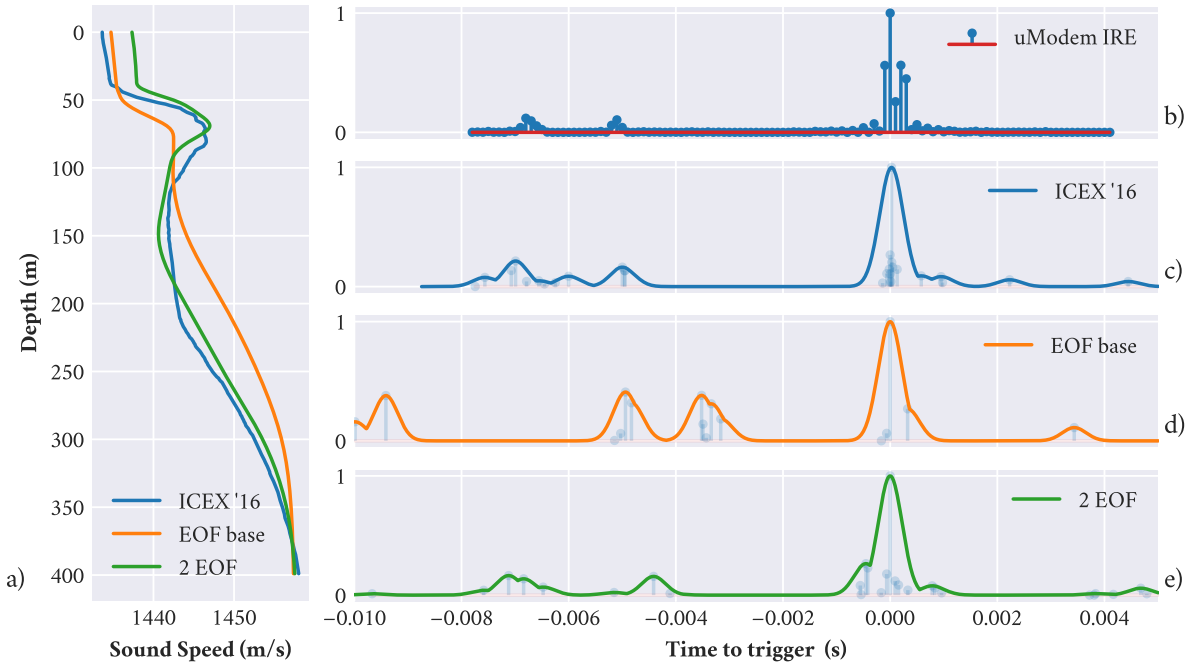


Figure 4.13: Sound speed profiles (a) and normalized model-based impulse response estimates (c-e) for different environment approximations; impulse response from physical modem (b) was obtained from NETSIM, for a simulated vehicle located at 7.1km range and 70m depth.

The hardware-in-the-loop scheme was exploited to obtain IREs from real Micromodems, though the simulated solution is also shown for reference. The baseline model was taken from the EOF framework, with all-zero weights, and the adapted model was limited to using two EOFs only, for illustrative purposes.

A notable takeaway from Figure 4.13 is that the effect of EOF weights can sometimes be clearly observed in the acoustic model's output. With the signals shifted to a trigger (or, detection) basis, the shapes can be compared to one another, much as was done in Sec. 3.3 for the riverbed characterization problem. The risk, on the other hand, is that the limited span of the IRE may at times be too short to capture the complexity of the arrival structure. After all, the IREs at hand capture approximately 10 milliseconds of data, when a typical transmission may be anywhere from 1 to 3 seconds long in total.

4.4 Summary

This chapter has covered the core systems required for the ICEX-20 AUV experiments. Later sections also discussed the theoretical foundation and the simulation work conducted as part of the development of an environmental adaptation framework for ICEX-20. These simulations were used to inform the experimental design, including system configurations, with the aim to maximize the potential for further work in post-processing.

The theoretical work discussed in the later portions of this chapter was presented in "Model-aided acoustic environment estimation via data fusion for autonomous underwater vehicles" and "An information theory approach to assess acoustic-environmental significance". The insights drawn from this simulation work are exploited later in this thesis, to improve on the performance of fielded algorithms.

5 ICEX-20 FIELD REPORT AND DATA REVIEW

And through the drifts the snowy clifts

Did send a dismal sheen :

Nor shapes of men nor beasts we ken–

The ice was all between.

The ice was here, the ice was there

The ice was all around :

It cracked and growled, and roared and howled,

Like noises in a swound!

– Samuel Taylor Coleridge, *The Rime of the Ancient Mariner*

Though Coleridge's text has his subjects traveling South, this powerful description of a frozen domain is just as apt for the Arctic Ocean. The ice was indeed all around – this was what we traveled North for! And the trek out to the territory of the great bear, as the ancient Greeks so aptly put it, was not one without risk. This chapter presents a review of the data collected during ICEX 2020, covering static experiments as well as the tethered and untethered deployments of AUV *Macrura*.

The core contributions of this chapter include (1) a holistic overview of the acoustic data collected during ICEX-20, including a brief discussion on data loss in the field; (2) an in-depth discussion of

the quality of the acoustic data with respect to the operational paradigm and clock synchronization challenges encountered in the field; and (3) a discussion of the impact the acoustic data had on the vehicle's navigation performance and on field operations, including its enabling a complex recovery operation.[1] The data quality assessment in this chapter was also used to support derivative work by the author's peers at the Laboratory for Autonomous Marine Sensing Systems.

5.1 Experiment Timeline

5.1.1 Getting to camp – and coming back

The story of how we traveled to and from ice camp Seadragon is perhaps an unconventional bit of data. This piece of evidence, anecdotal in nature, speaks not of environmental measurements and other data troves collected – that will come later. Instead, the story itself centers on the operational challenges we faced by pursuing successful deployment and recovery operations in such an extreme environment. Certainly, we are not the first to deploy scientific equipment in the Arctic, nor are we the first to experience difficulties out there. But, as a part-time outdoors-person, I am reminded of the American Alpine Club's (AAC) ongoing efforts to record and retell the most significant and teachable moments of recent expeditions in their publication titled *Accidents in North American Climbing*, which has been published annually since 1948. Much in the same spirit as the AAC's annual reports, this particular section seeks to serve as a brief reflection on our experience, in the hopes that it may serve as a reference for future experiments of this kind. More importantly, this very story captures the practical significance of the work we were looking to demonstrate out there – but I will get to that later in this chapter.

The participation of MIT LAMSS at ice camp Seadragon was programmed for March 6 through March 14 of 2020. Travel to the operational headquarters at Prudhoe Bay, Alaska, went largely as

[1] Randeni, Schneider, Bhatt, Viquez R., and Schmidt, "A high-resolution AUV navigation framework with integrated communication and tracking for under-ice deployments".

planned via commercial airlines. Personnel transport from Prudhoe Bay to camp Seadragon was facilitated by the Royal Canadian Air Force, using a Twin Otter (CC-138); equipment was transported using a CASA short takeoff and landing (STOL) aircraft. The first few days were dedicated to setting up our network infrastructure, which included the testing and deployment of the radio-enabled buoys in the field; we also had to prepare the vehicle, both on the software side by updating the autonomy stack to the latest version, and on the hardware side by checking the pressure vessels and assembling the vehicle body in full. *Macrura*'s first taste of the Arctic Ocean as part of ICEX-20 occurred on March 9, 2020, when a tethered run was conducted to validate all systems. Prior to the vehicle entering the water for the planned ICEX-20 missions, the acoustic tracking range was tested using a *virtual vehicle* approach, which relied on a standalone modem and transducer system to interface with the physical ocean.

The day after that first engineering test, on March 10, the LAMSS team faced a couple of challenges worth reflecting on. The first of these consisted of a failure in the power circuitry, which required us to open the front-seat payload. We had encountered this same type of power failure once during prior engineering trials conducted in December of 2019 in Massachusetts Bay. While a repeat occurrence was certainly undesirable, the engineering trials had put the team on alert for this failure mode, and both spare parts and diagnostics procedures were at hand to address the issue. The second challenge stemmed from a disk failure in the back-seat payload – a different pressure vessel from the one housing the front-seat computer and power circuitry. Getting access to the second payload necessitated additional disassembly of the vehicle. In the end, we were able to address this issue as well, with the spare equipment at hand; but not without jumping through some hoops to recover much of the work performed in the days leading to the experiment, and changes made onsite. The *mostly* unavoidable loss in both cases was that of the most precious resource in this kind of experiment: time. I say *mostly*, because during the time when the vehicle was being serviced, we

were conducting additional experiments with the buoys deployed in the field and the *virtual vehicle* system we had first used to test the tracking range.

Overcoming issues that required both payloads opened in a single day out in the field, and with failures occurring one after the other (rather than happening concurrently), may well have set a new record for the LAMSS team's fastest onsite recovery at this scale; but the greatest challenge we encountered out there was one we could do very little to fix. The morning of March 11, we were informed that a weather forecast first released the day before remained unchanged: a storm would soon arrive, and we had but one day left to conduct our experiments before packing up. The MIT LAMSS team would have to leave camp Seadragon on March 12, two full days earlier than planned. Thus, it followed that on March 11, we conducted additional tethered deployments and eventually moved forward with an untethered mission.

It was during that untethered mission that the final challenge took shape. Another fault had caused the vehicle's propulsion system to cease responding. With its propeller at a stall, and given its slightly positive ballast, *Macrura* began to drift upward until it reached the layer of ice at the surface. Despite the fault that disabled the propulsion system, the vehicle's payload computer – responsible for handling acoustic communications – remained active, and the acoustic tracking range was able to locate the vehicle even as it settled under the ice. The AUV was approximately 1 km away from camp, and a storm was coming.

5.1.2 Datasets acquired

As is so often the case when deploying complex systems in the field, the planned timeline was ultimately impacted by logistical issues such as a temporary loss of power at the research facility – meaning, in this case, a tent at the ice camp which was designated specifically for scientific efforts – as well as the aforementioned component failures and the competing use of limited resources such as the hydrohole required to deploy AUV *Macrura*. Nature also had its say, when the approach of

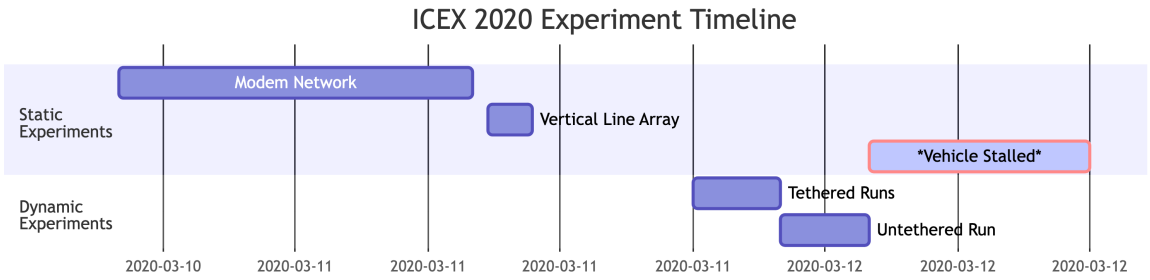


Figure 5.1: Gantt chart showing the timeline (UTC) of ICEX 2020 experiments.

a major winter storm necessitated the evacuation of the ice camp – a move that cut the anticipated experimental window by about two days.

In terms of data collection, the ICEX 2020 experiments formally began during the afternoon of March 10 (in AKDT). The first dataset consisted of a static experiment in which one of the buoys in the ICNN was operated as a proxy for *Macrura*, and the tracking network was supplement with an additional modem deployed from the ice camp to keep the number of operating nodes unchanged. The second dataset was from another static experiment, in which the vehicle's towed array was weighted down and operated as a vertical line array (VLA). The vehicle was lowered to depths of 40, 60, 80 and 120 meters with a tether, and held at each depth for approximately 15 minutes each. On March 11 of 2020, the vehicle was deployed under the ice, first on a tether for short-range loiters, and later on without the tethered, for a longer mission. On its return home from the long mission, the vehicle stalled due to an unexpected hardware failure; as the vehicle is slightly positively buoyant, it made its way to toward the surface and settled under the ice, effectively creating another generally static dataset. Figure 5.1 illustrates the acquisition of these datasets along the timeline, given in UTC rather than Alaska time (AKDT) since universal time is the standard for LAMSS data logs.

5.2 Overview of the acoustic communications logs

At the heart of the ICEX-20 datasets live three components collected from the acoustic environment: the first of them is the acoustic communications data. These logs capture the data used by the ICNN to track the vehicle's position over time, for example. The other two sets consist of recordings collected by the vehicle's array, in both VLA and horizontal line array (when towed) configuration; and of data collected by an upward-facing side scan sonar fitted to *Macrura*, as LAMSS supported the work of the US Navy's unmanned undersea squadron, UUVRON. Due to the sensitive nature of the data in the latter sets, they were both restricted to authorized personnel only. This thesis, and this section in particular, make use of the first set composed of the acoustic communications logs.

As a first overview of the communications logs, Fig. 5.2 shows transmission and reception events associated with each of the modems, with the event type indicated by the direction of the marker and success or failure of the event indicated with different colors. Events are also staggered vertically by type and status, to facilitate reading an otherwise dense plot. It may be noted that all modems report reception failures at some point, while transmission failures are basically non-existent in this dataset. This is expected, since transmission failures would be associated with a modem's inability to queue up a data packet.

5.2.1 Modem Error Codes

The acoustic link used by the WHOI Micromodems can be compared to other data transport protocols in that, at a high level, we can partition a transmission into three segments that represent the information needed for a successful data transfer. The first is the leading chirp, used to establish the link by triggering the detection system on the receiving modem. The second segment reports the parameters used for signal modulation (for ICEX-20, LAMSS used the Micromodem's Phase-Shift Keying modulation, or PSK), which the receiving modem needs to decode the remaining segment. The final portion of the signal contains data headers and the actual body of information transmitted.

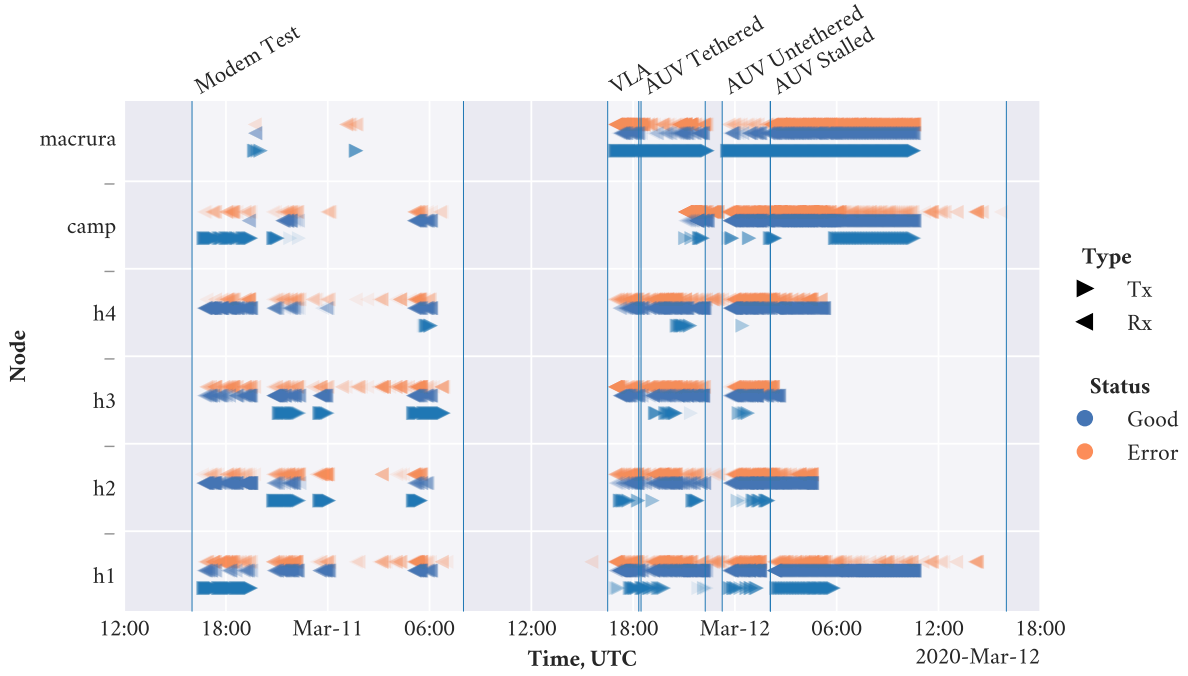


Figure 5.2: Timeline of acoustic transmission and reception events collected during ICEX 2020, identified by experimental subset.

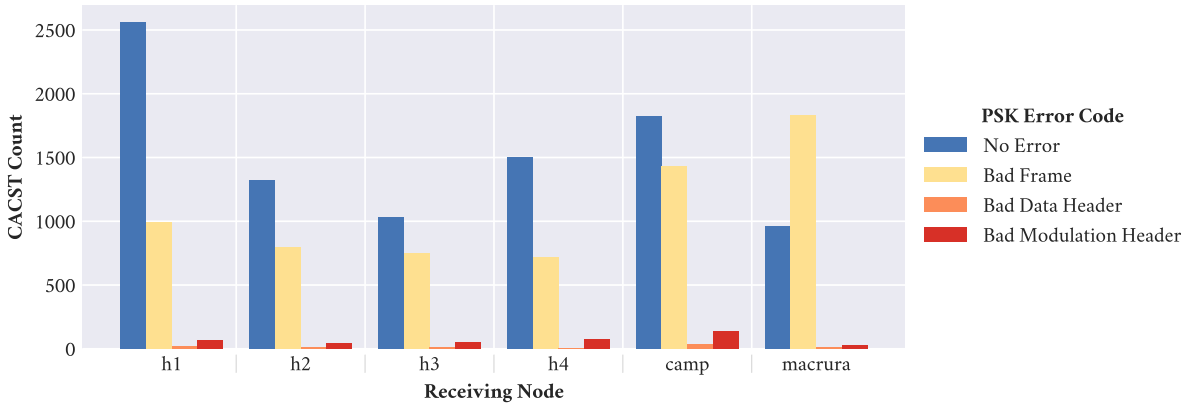


Figure 5.3: Failure mode distribution for acoustic reception events at each node in the network. Failure modes are categorized by each event's PSK error code, as reported by the WHOI Micromodem. Events with bad frames represent a notable portion of the data.

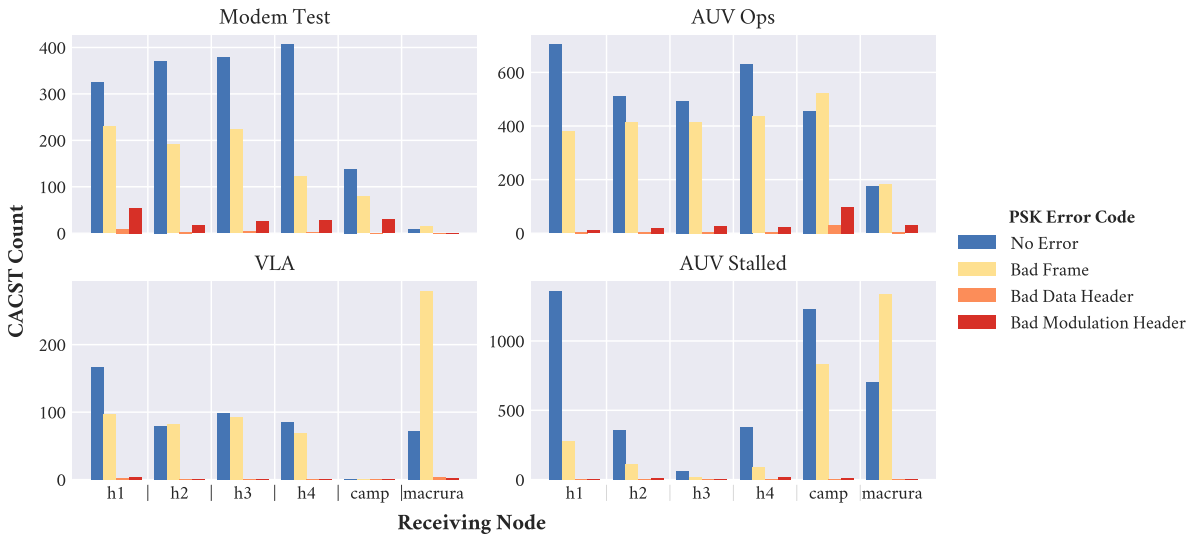


Figure 5.4: Failure mode distribution for acoustic reception events at each node in the network, for each experimental subset. The particular distributions change with the varying spatial arrangements, after breaking down the data set into smaller windows. However, it can be observed that the events with bad frames continue to represent a notable portion of each subset.

The structured nature of the Micromodem's signals makes it possible for a receiving modem to report different error codes depending on what stage of the decoding process actually failed. Taking a closer look at the PSK error codes reported for the reception events from each node, shown in Fig. 5.3, it becomes apparent that the most typical failure mode is related to bad data frames. This pattern generally holds true when breaking down the data across all experiments, as shown in Fig. 5.4. There is a visible relation between likelihood of failure and the positioning of nodes in the network, as suggested by the significant drop in proportional failures for receptions at buoy *h1*, during the window in which the vehicle had stalled; it should be noted that these events do include receptions of topside-controlled transmissions at other buoys in the network.

5.2.2 Impulse Response Estimates

Working on large, multidisciplinary and cooperative operations brings on challenges beyond the scope of the research efforts at the center of the project. This is especially true when the work re-

quires timely coordination across multiple organizations. Despite all efforts to minimize surprises when the team was finally in the Arctic, a series of logistical and disbursement delays encountered during the development stages leading up to the experiment meant that some of the equipment needed for the experiment – namely, the ICNN buoys – were not available for testing during the engineering trials conducted with AUV *Macrura* in the Massachusetts Bay on December of 2019. Instead, the two systems were tested separately and the radio drivers that interfaced both systems were tested on the bench but not in the field.

When the two systems – the ICNN buoys with their radio network, and the vehicle operations stack – were finally brought together atop the Arctic ice, it became apparent that the radio implementation used to connect the buoys to ice camp was more limited than what was predicted by the equipment's theoretical capabilities. When enabling all the expected messages for each of the Micromodems on the ICNN buoys, the radio links became unreliable.

Being already on the ice when this issue came up, with very limited time for making adjustments and a pressing need to proceed with the vehicle deployments required to meet the team's scientific objectives, a decision had to be made quickly. Disabling some of the non-critical reports produced by the modems, including the impulse response estimates produced for each detection event, lessened the pressure on the radio network to the point that it was once again able to support vehicle operations. Thus, the decision was made to operate the buoys with this reduced configuration. The impulse response estimates produced by the WHOI Micromodems, one of the data sources initially anticipated for this work, were among the affected records; Fig. 5.5 illustrates the number of IRE entries relative to the number of detections. The data collected by the modem onboard *Macrura* was unaffected, since the vehicle's interface to the modem was wired rather than based on a radio link.

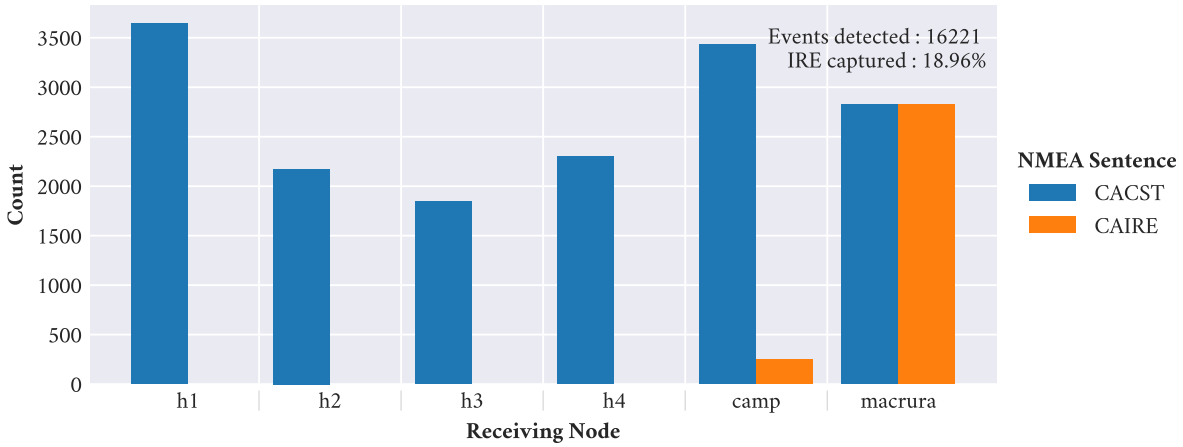


Figure 5.5: Number of impulse response estimates collected, compared to number of events detected per node. Limitations in the radio infrastructure used to communicate with remote nodes required that the collection of impulse response estimates be disabled on those modems, while in-situ mission reconfiguration impacted the collection of samples from the camp's modem.

5.3 Coordinated operations and the TDMA scheme

The acoustic communications paradigm used by LAMSS is based on a Time-Division Multiple Access (TDMA) scheme, where the timeline is divided into slots which are then assigned across the network's nodes. By doing so, multiple transmitters can access the same frequency channel. For vehicle operations with the ICNN, transmissions through the network's buoys are assigned to the topside system despite being unique transmitters. The time division is thus based on two nodes only, and each slot is set to a width of 15 seconds to avoid overlapping transmissions within the operational range. Because the ICNN was deployed in the vicinity of and in coordination with US Navy submarines, the first topside slot of each 4-minute cycle was reserved for pinging the Navy's submarines. Thus, the TMDA cycle used during ICEX-20 is illustrated in Figure 5.6. Under this scheme, transmissions used for AUV operations are expected to occur 1 second into each TDMA window; the start of the window triggers clearing data buffers (for example, the travel time tables used for trilateration on the ICNN), and queuing up the next outgoing message. Later references

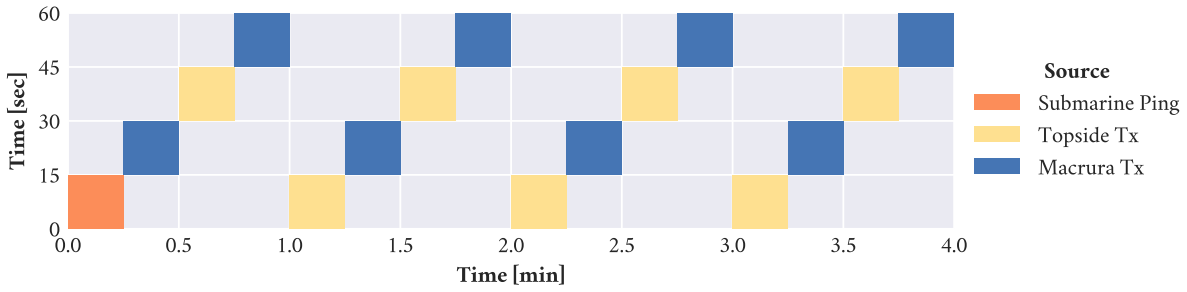


Figure 5.6: Visualization of the TDMA scheme used for ICEX 2020. The first slot in each 4-minute cycle is reserved to ping U.S. Navy submarines in the area as part of coordinated operations. The remaining slots are assigned to the topside and vehicle systems in alternating fashion. The 15-second slot width supports operations in a 10 x 10 km grid, with enough buffer time at the end of each slot to clear late-arrival paths before the next transmission.

to the TDMA cycle will focus on the minimal 2-node pattern, spanning 30 seconds at a time as it alternates between a topside slot and a vehicle slot.

5.3.1 Time synchronization

One of the requirements for the TDMA scheme and the trilateration algorithm is that all nodes share a common time reference. For nodes with a surface expression, as is the case for the mission control computer on topside and the ICNN buoys, this is achieved by exploiting GPS time and the associated pulse-per-second (PPS) signal. On the vehicle, which lacks regular access to the GPS network, a high-precision chip-scale atomic clock[2] (CSAC) is used instead. The vehicle's CSAC is first synchronized to GPS time and PPS via a secondary, external CSAC which is kept in sync with the GPS network, along with the rest of equipment on the surface.

The Micromodem supports multiple options for timekeeping, including an integrated clock. As part of the LAMSS operational framework, the modems are typically configured to accept updates from external sources, such as the GPS receivers, which output time via the NMEA sentence `GPZDA`; upon receiving an update, the modem sets its internal clock accordingly. The PPS output from the

[2]Gardner and Collins, "A second look at Chip Scale Atomic Clocks for long term precision timing".

GPS receiver, or from the CSAC onboard the vehicle, is also connected to the Micromodem as an additional reference to ensure proper time synchronization.

5.3.2 Effects of clock synchronization error: acoustic transmissions timeline

Transmission events are largely unaffected by modem clock drift alone, because the actual transmissions are controlled, first and foremost, by the topside computer's decision that it's time to relay data through the modem; or, for vehicle transmissions, a decision made by the autonomy computer. Upon doing so, the appropriate computer sends said data to the modem for buffering. The second level of control for transmissions is the GPS or CSAC PPS line, which is connected to the modem's appropriate input and acts as the trigger for the modem to begin emitting the data buffered from the topside computer. Thus, there are 3 requirements for healthy transmission events to occur:

- The modem's PPS input signal is stable and accurate, to ensure transmissions are triggered on time.
- The time set on topside computer or vehicle autonomy computer is accurate, to correctly determine when to start loading the modem's data buffer.
- The delays in serial communications, including transit through the FreeWave radio network, do not cause an overflow into the next PPS cycle while the computer is loading data to the modem's buffer.

As long as the conditions stated above hold true, transmissions shall occur as expected within the TDMA cycle constraints. This holds true, even if the recorded time in `CAXST`, the modem's transmission statistics, appears to differ significantly from the computer's log time. This data flow control is illustrated in Fig. 5.7.

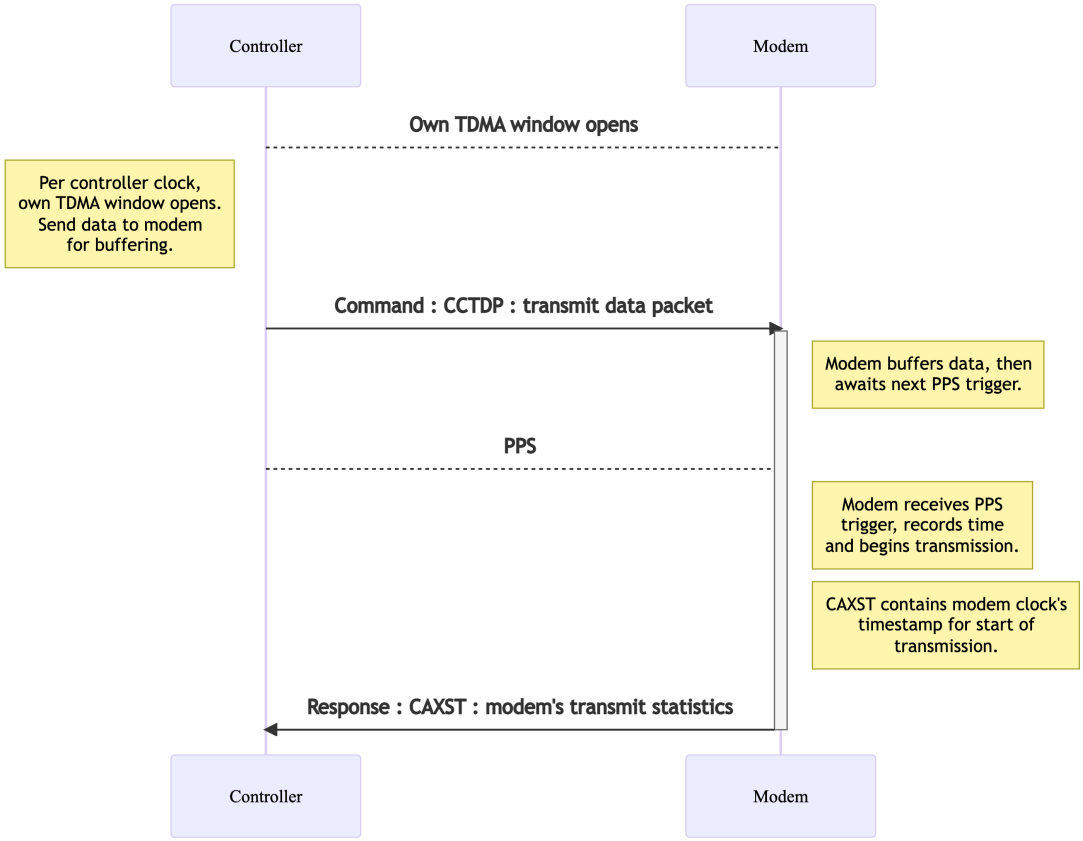


Figure 5.7: Acoustic transmission: timeline of clock-related events. Transmissions are generally robust to modem clock misalignment, since they depend on: (1) the timing of the transmit command coming from the topside computer, and (2) the pulse-per-second signal produced by the node's timekeeping solution (GPS receiver or synchronized CSAC). The modem clock provides timestamps for transmission statistics, but does not control the transmission events.

5.3.3 (More) Effects of clock synchronization error: differences between transmission and reception events

Transmission events are generally quite robust to modem clock misalignment, depending on an external PPS line and the time kept by the computer in control of the modems. Reception events, on the other hand, are not so forgiving. Incoming (reception) events are highly sensitive to modem clock drift, as the only high-precision reference associated with them is the timestamp provided by the modem. This timestamp is collected on the topside or vehicle computers from the `CACST` message produced by the modem, which reports the reception statistics (Fig. 5.8).

Using a PPS-synchronized timestamp captured by the modem allows us to compute acoustic travel time with high precision, as long as the coarse time settings – hours, minutes, seconds – are also set correctly. When all nodes are properly synchronized, the bulk of ranging errors are generally driven by the interpretation and handling of said travel times vis-à-vis the choice of sound speed model used to estimate the acoustic environment.

Impact of the radio link on event timing

Processing and transit times introduce variability such that we cannot use log times as reliable sources for acoustic travel time. An error in the order of tens or hundreds of milliseconds translates to tens or hundreds of meters in the range estimation error; this is the kind of error that could be introduced when the radio link fails and needs to try sending a message again before it is successfully received by the topside computer. Thus, the value of log time is especially limited for modem reports traveling through the FreeWave radio channel.

As part of the LAMSS operations protocol, the topside computer generally runs a Network Time Protocol (NTP) server, which is configured to ingest timing messages (i.e. `GPZDA`) and the PPS signal from a GPS receiver to ensure it is in sync with universal time. Rather than having the modem accept GPS time updates directly, the lab's historical approach to managing modem time consisted of a

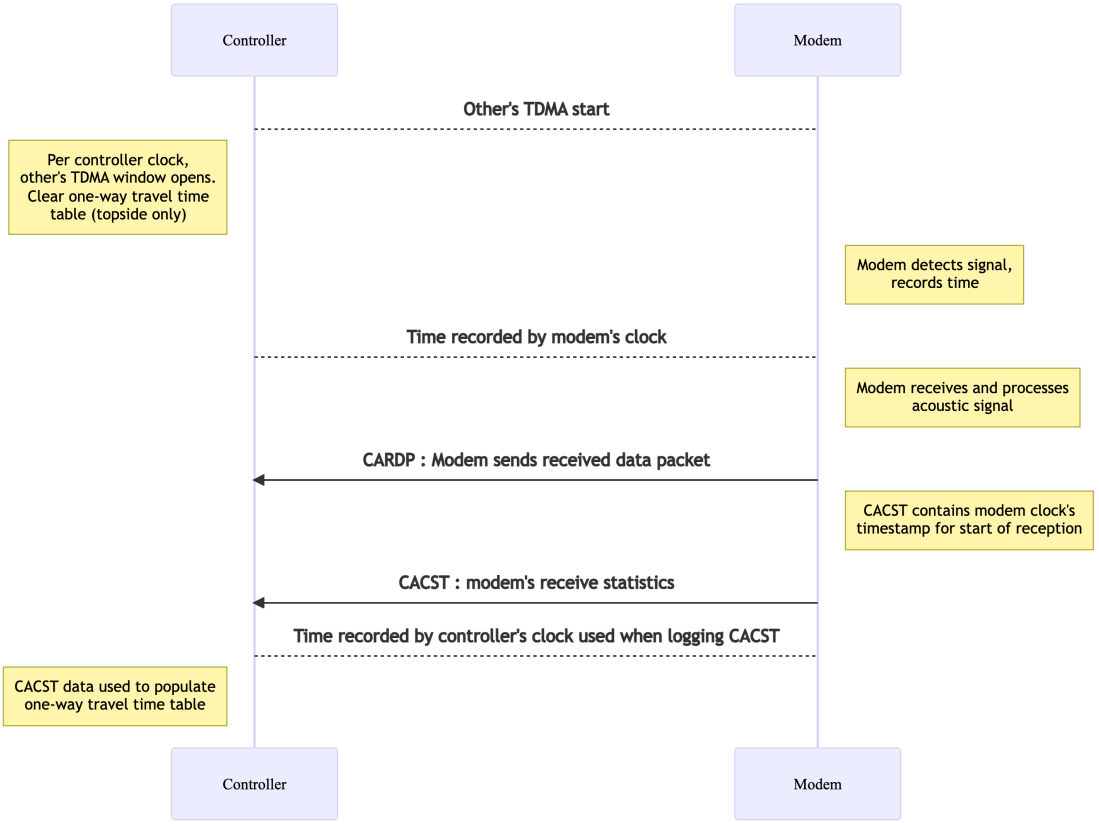


Figure 5.8: Acoustic reception: timeline of clock-related events. The modem uses its internal clock to produce timestamps for reception events. Misalignment of the modem clock relative to GPS time jeopardizes the best opportunity for high-precision timing, as the serial interface to the topside computer may introduce significant additional delays.

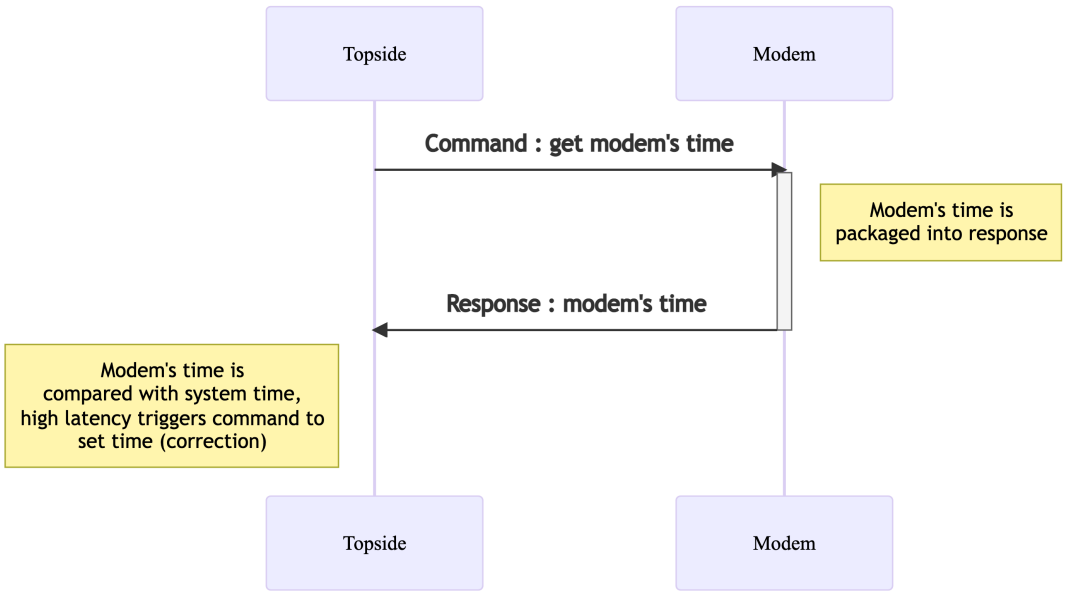


Figure 5.9: Diagram illustrating the conventional serial interface to acoustic modems. When communicating over a wired interface, each modem uses a designated interface and there is less chance of failure due to dropped packets.

probe and update process. This stemmed from early experiences with prototype modem hardware, which had unreliable internal clocks, and was adopted by subsequent generations of students as the accepted approach. Managing the modem time in this way generally meant that mission logs would provide some insight to modem timekeeping issues, and allowed the mission control computer on topside to manually update the modem's time when it ran too fast or too slow relative to the NTP server. This data flow is illustrated in Figure 5.9.

Historically, this approach to modem time management had been employed over hardwired connections. Any delays introduced by the serial interface and the computer's operating system (not a real-time OS) could be accounted for by defining some tolerance threshold to the modem time's latency; any significant offsets could generally be attributed to the modem clock itself. The same clock management scheme was employed for ICEX-20, under the expectation that the FreeWave

radios that connected the topside computer to the ICNN buoys, along with the software required to enable the interface, would behave transparently as a radio-based serial interface.

In practice, the throughput of the radio link was found to be lower than expected – though the theoretical capacity would have supported it, the link could not be used as an equivalent to the hardwired serial interface. Instead, the experience attained during systems testing on the ice made it apparent that the radio interface introduced new and significant delays between the modem and the topside computer. The resulting data flow between these nodes, then, is better described by Fig. 5.10.

5.3.4 Time skew in the ICEX-20 data

During the first few days on the ice, while conducting the various system tests necessary, time skew was one of the key indicators of the issues introduced by the radio link. The buoys had built-in GPS receivers with a PPS line connected directly to their respective Micromodems, but the coarser time setting was managed by topside, per the aforementioned scheme, rather than being updated directly by the buoy's built-in GPS. The limited throughput of the radio interface encouraged the team to reduce the sync probing rate for the remote modems. However, the reduced visibility of the modem clock in the absence of a supplementary reference led to the buoys occasionally drifting far out of sync. The distribution of time skew records available for each modem, along with the corresponding number of events, is shown in Figure 5.11. The skew data shown is computed by subtracting the log time, recorded by the control computer upon receiving a message from the modem, and from the content of modem messages linked with timekeeping: `CATMS`, the response to time-setting commands; `CATMQ`, the response to time queries; and `CATMG`, reported by the modem when the timing sources change – this occurs, for example, when the PPS signal produced modem's real-time clock syncs with the external PPS line from a buoy's GPS receiver.

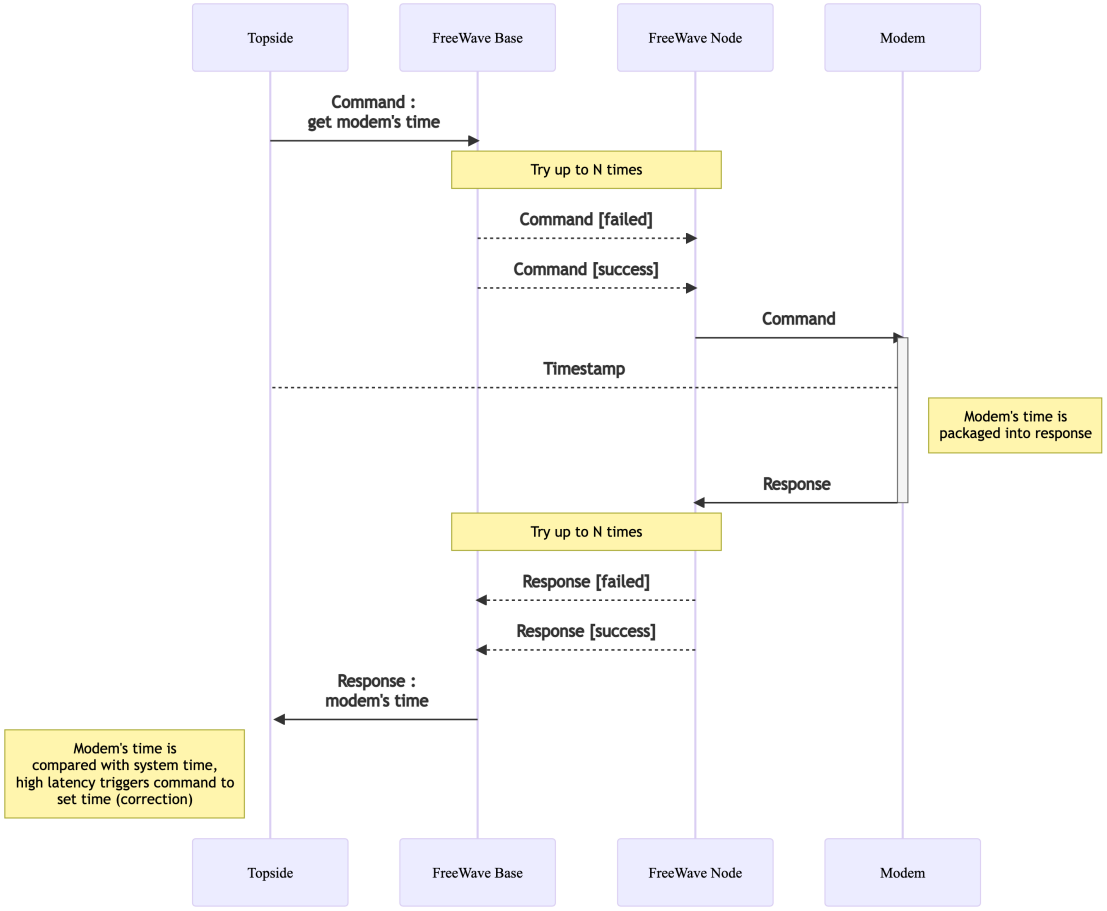


Figure 5.10: Diagram illustrating the radio link as a substitute to the serial interface for remote acoustic modems. The latency of the radio interface increases the chances for delayed or dropped packets, impacting the reliability of the interface.

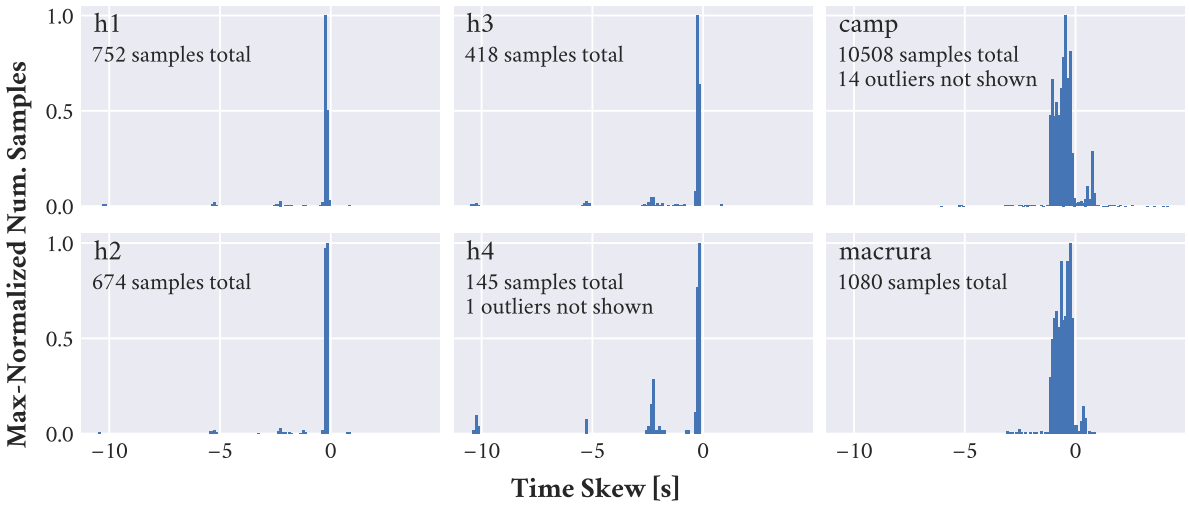


Figure 5.11: Time skew by node, as collected from explicit time queries sent to the node's Micromodem. The skew is given with respect to the control computer's time; operator workstation for surface nodes and payload computer for the vehicle.

The number of timing probe entries for *Macrura* may seem low compared to those for the camp modem, since they both use hardwired serial connections. But, it should be noted that the vehicle was not active outside of the VLA experiment and the vehicle deployments, except for some brief testing conducted on the bench during the modem test experiment. The effect of limited time monitoring is particularly apparent for *h4*, where some of the entries are visibly clustered around 2.5, 5 and 10 seconds behind the computer's own time.

5.4 A CLOSER LOOK AT THE ACOUSTIC COMMS DATA

The previous section discussed the importance of time sync in coordinated operations. In that scope, the effect of replacing wired connections with a radio network can become quite significant when the latter cannot keep up with the expected throughput. Having covered that discussion, this section revisits the acoustic communications logs and looks at the data within the framing of the TDMA scheme.

5.4.1 Transmission events

Because of how they are timestamped, the transmission event logs actually capture the symptoms of clock drift. This is illustrated in Figure 5.12, which illustrates the distribution of events along the 2-node TDMA cycle for the two time records available. When using the date and time recorded within the modem's transmission statistics, collected via the `CAXST` message, then some events that are apparently not aligned with the expected TDMA cycle become apparent. This applies, in particular, to transmissions put out from the camp modem during the modem test experiment. All transmissions made from camp were expected to fall within the topside portion of the cycle (the first 15 seconds). These seemingly tardy events, for the most part, come from a particular subset of the experiment during which the camp modem's clock is apparently drifting, unchecked, for nearly an hour.

When looking at the same data relative to the log times, it becomes clear that the transmissions sent out from the modem at camp were not, in fact, delayed by 5, 10 or nearly 20 seconds. Instead, all camp transmissions occurred within the appropriate TDMA window. Because the log times reflect the moment when the computer receives the `CAXST` message from the modem, they are recorded after the modem has completed the transmission and relayed its performance metrics. However, the clustering of events with respect to the log times suggests that these transmissions did actually occur at the correct moment, triggered by the PPS. It is worth recalling, at this point, that the modem clock and the topside's GPS-synced NTP server are independent; the drift of the modem clock does not in any way influence the control computer's clock. On the other hand, loss of the GPS signal – or a poor connection to the GPS network – could affect the topside computer's synchronization, since it is not consistently attached to a CSAC. The presence of generators, radios, and other electronic equipment, in addition to the tent's structure, could all influence the reliability of the topside computer's clock.

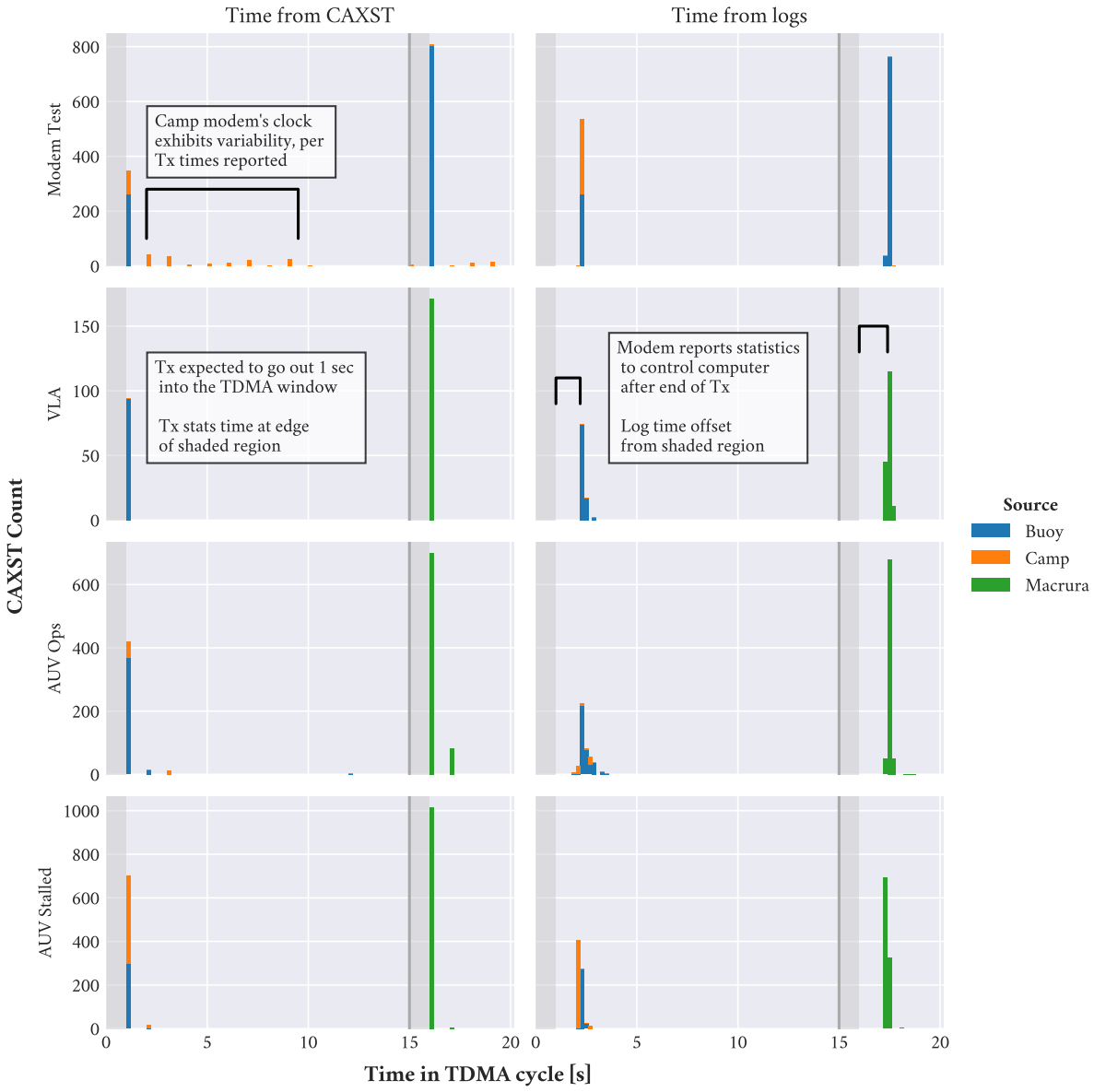


Figure 5.12: Distribution of transmission events along the TDMA cycle, based on the timestamps reported by the modem in the transmission statistics message (left) and on the time of the log entry, given by the control computed (right). When using the date and time recorded within the modem's transmission statistics, some events that are apparently not aligned with the expected TDMA cycle stand out; when looking at the same data relative to the log times, it becomes clear that the transmissions sent out from the modem at camp were not, in fact, delayed. This timing mismatch provides additional insight into the health of the ICNN's synchronization, beyond that attained from explicit time queries.

5.4.2 Reception events

The importance of timing was stressed in Section 2.4, as it is the foundation for Positioning, Navigation and Timing (PNT) systems – ordered by their level of dependency, these should perhaps be named TPN systems instead, but in the interest of clarity, the PNT convention will be used henceforth. Timing enables positioning, and the latter can then inform navigation decisions such as course and speed corrections.[3] Under this purview, acoustic travel times measured through the time-synchronized reception events are what ultimately allow the ICNN to use trilateration to estimate the vehicle's position; that is, after the travel times are converted to range estimates based on the acoustic propagation models.

To that end, Figure 5.13 captures the distribution of arrival times at each node, along the TDMA cycle. The events are coded by their respective PSK error code, to illustrate at a high level how longer exposure to the ocean can effectively make it more difficult for the receiving modem to successfully decode the signal. The signal distortion that causes the increase in difficulty could be attributed to numerous causes, such as the effect of boundary interactions or closely timed multi-path arrivals.

With respect to the distribution of arrival times in Fig. 5.13, the first observation to be made is that successful arrivals are front-loaded within each TDMA slot. This is certainly expected, as it is here that direct paths or minimally distorted surface bounce paths would fall (ray geometries and travel times will be discussed in more detail in Chapter 6). However, the second notable feature is that there are numerous late arrivals – with travel times of 4-6 seconds, for example – that fail due to some number of bad data frames in the signal. These can be seen within the vehicle's transmission window, roughly around 20-22 seconds. Failures related to corrupted data and modulation headers appear in the dataset as well, though much less frequently. Given the local bathymetry and predicted paths, it is likely that at least some of these entries correspond to bottom-bounce paths.

[3] Howe, Miksis-Olds, Rehm, Sagen, Worcester, and Haralabus, "Observing the Oceans Acoustically".

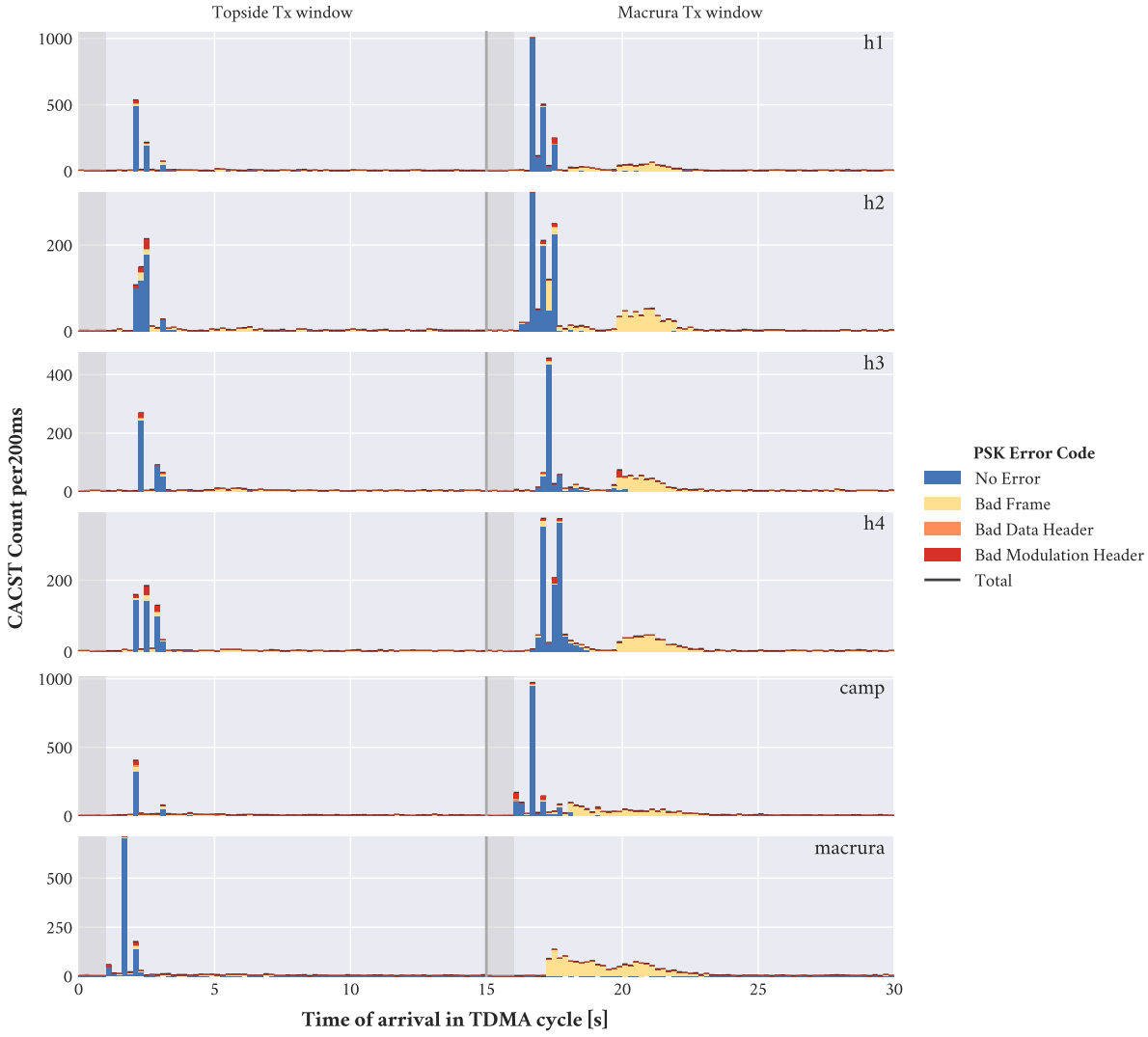


Figure 5.13: Time of arrival of reception events by node, given in the context of the TDMA scheme. Transmissions are expected to occur 1 second after the start of each window (shaded regions). Successful arrivals are front-loaded within each TDMA slot; late arrivals – with travel times of 4-6 seconds, for example – fail primarily due to some number of bad data frames in the signal, but appear to contain valid headers.

It should be remarked that the vehicle (bottom plot) does not report any arrivals during the first second or so after the expected start of transmission, for the vehicle's TDMA slot – that is, around 16-17 seconds. The start of the transmission, as was mentioned earlier, is expected to occur one second into each time slot, and this offset is illustrated by the shaded region at the start of each slot. Micromodem transmissions can span about 1-3.5 seconds depending on the number of frames sent, so the gap in the travel time distribution is consistent with the period during which the vehicle is transmitting. The subsequent series of arrivals would then be consistent with the signal returning to the vehicle through any number of indirect paths.

The likelihood that at least some of those paths returning to the vehicle include surface or bottom interactions is high. Furthermore, it is worth noting that the vehicle modem's transducer is attached at the bottom of the vehicle's mid section. Thus, surface bounce paths – which would be the first to return to the vehicle at its shallow operating depths – would likely be at least partially influenced by the vehicle body; the fact that the reported arrivals contain bad frames is of little surprise. As each of the ICNN buoys may record arrivals from another buoy's transmissions, the presence of successful entries in the earlier portions of the topside TDMA slot is very much expected across the tracking range.

Receptions, by experiment

As with the PSK error codes earlier in the chapter, exploring the bounded subsets for each of the experiments can provide some additional insights. To this end, Figures 5.14-5.17 break down the distribution of arrivals by experiment. This perspective captures some of the effects induced by changes in the vehicle's position and depth.

- **Modem test**

  The modem test was static in nature, as the experiment used only the ICNN buoys and a fifth transducer deployed with a towfish, for the camp modem. The vehicle itself was not

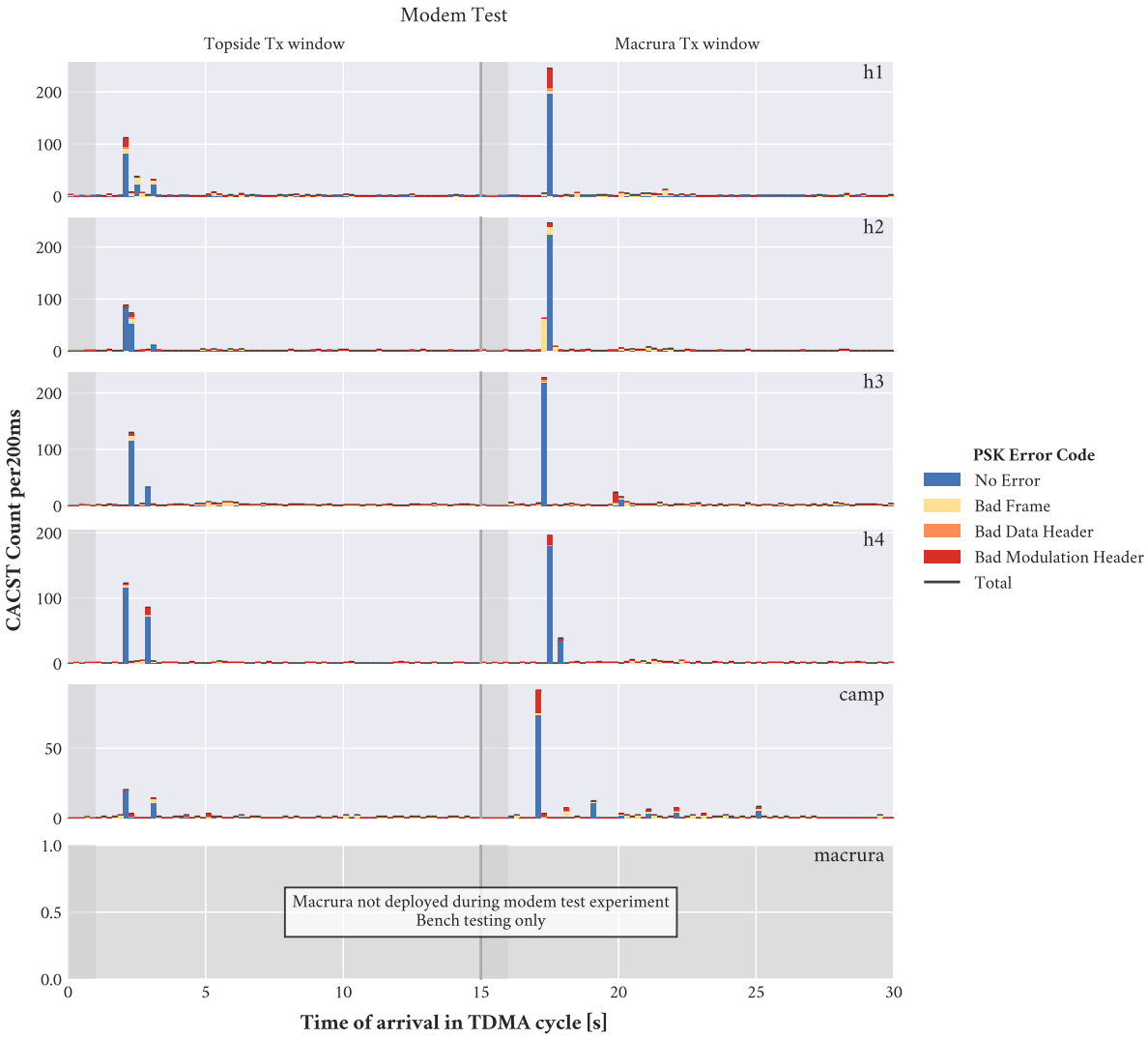


Figure 5.14: Time of arrival of reception events during the modem test experiment. The static nature of the experiment is captured by the narrow and generally consistent peaks. This set reveals few arrivals with travel times greater than 4 seconds.

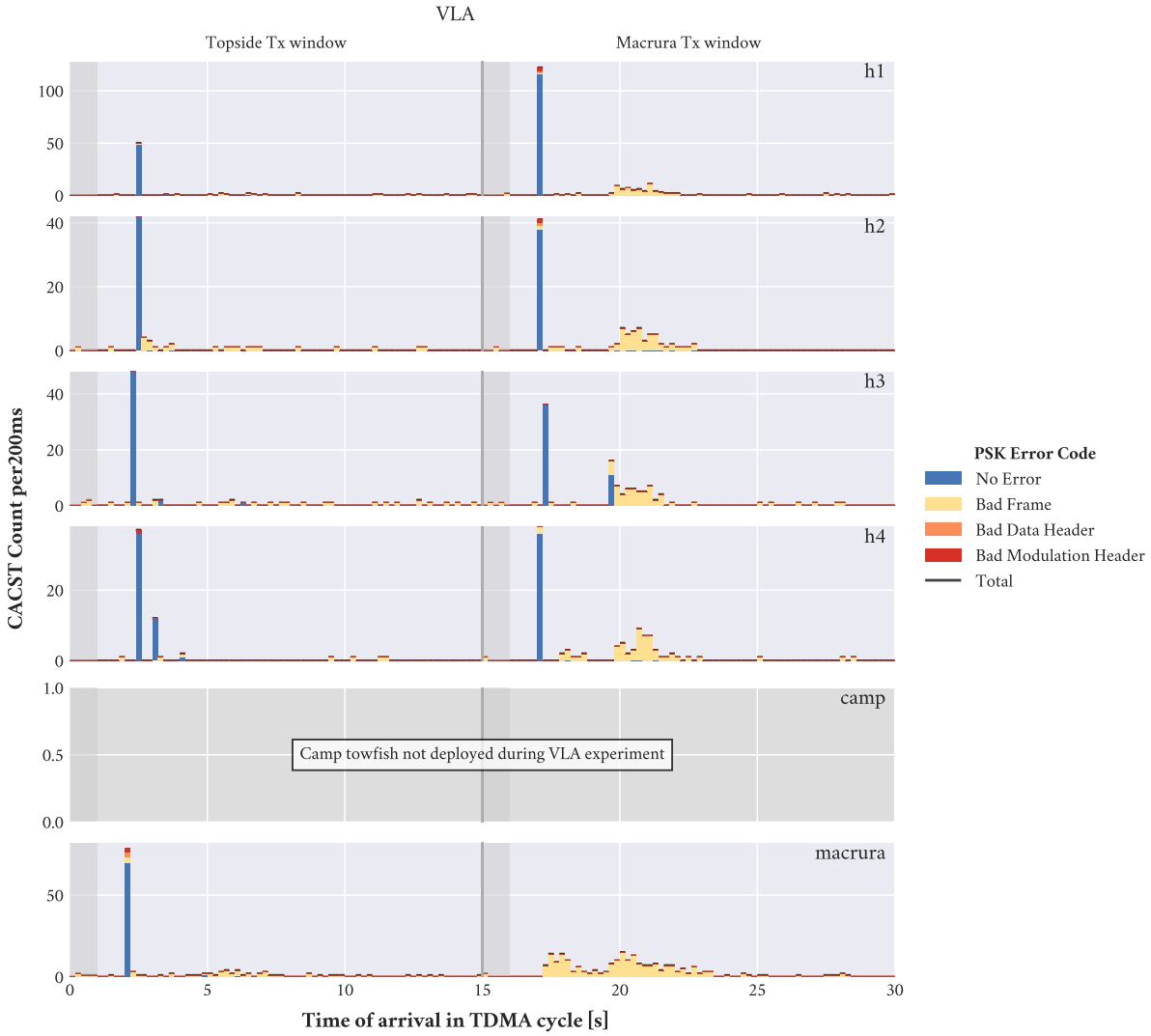


Figure 5.15: Time of arrival of reception events during the vertical line array experiment. The VLA experiment consisted of a series of shorter static experiments, where the vehicle was held at different depths for about 30 minutes at a time. The effect of changing depth is reflected in the appearance of the late arrivals within the vehicle's transmission window, while the quasi-static nature of the experiment is again reflected by the narrow peaks for successful arrivals.

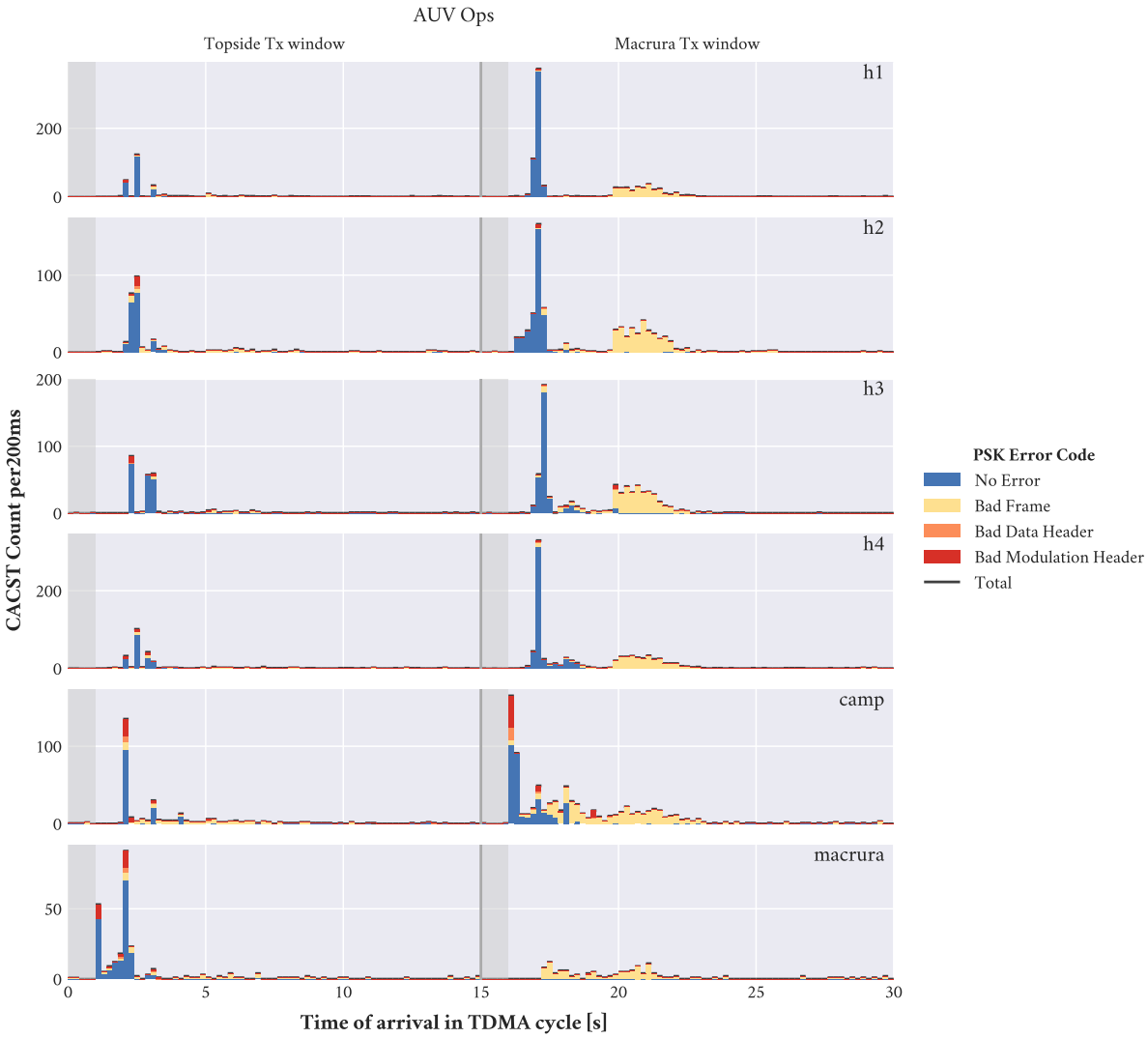


Figure 5.16: Time of arrival of reception events during AUV operations. The dynamic nature of AUV operations is captured by the widening of the peaks for successful arrivals, and the effect of changing vehicle depth remains visible through the distribution of late arrivals.

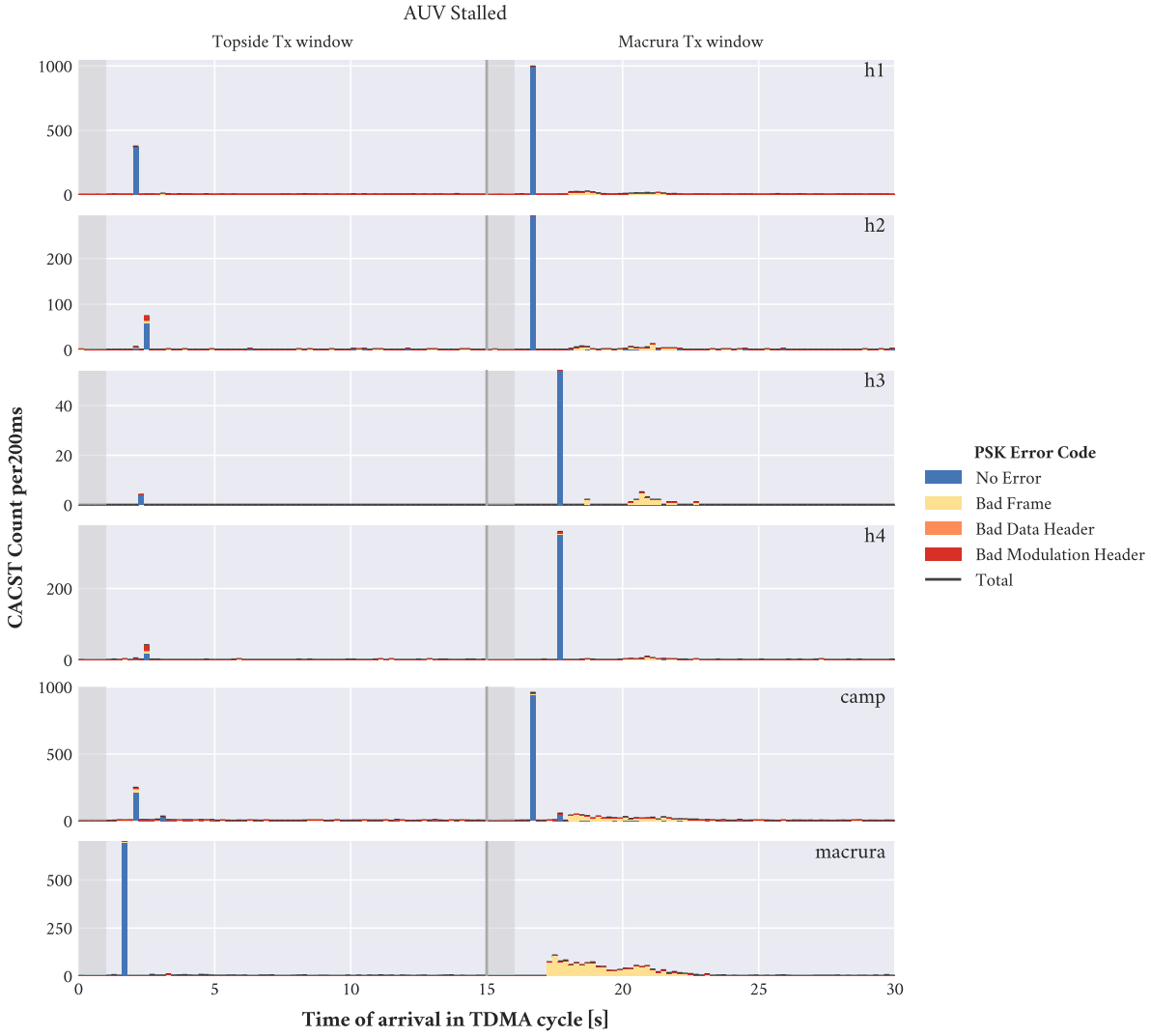


Figure 5.17: Time of arrival of reception events during the final phase of AUV operations, where the vehicle had stalled. This set effectively amounted to another static experiment; the arrival structure once again returns to tall, narrow peaks. With the vehicle settled in place under the surface ice, there are once again few arrivals with travel times greater than 4 seconds.

involved, and all transmission depths were fixed. The only adjustment possible in this set was the choice of reception depth layer for the buoys, which had hydrophones at 30 and 90 meters. The static nature of the experiment is captured by the narrow and generally consistent peaks in Fig. 5.14. The late arrivals observed earlier, during the vehicle TDMA slot, are notable for their absence in this set.

- **Vertical Line Array**

  The VLA experiment consisted of a series of shorter static experiments, where the vehicle was held at different depths for about 30 minutes at a time. With all subsets put together, Fig. 5.15 captures the effect of depth on the distribution of arrivals. Of note here is the reappearance of the late arrivals in the vehicle's transmission window. Though most of these contain bad frames, it is remarkable that ICNN buoy *h3* reports a handful of successful arrivals with travel times just short of 4 seconds (shy of the 20 second mark). Based on the acoustic models, the reported travel times for those events would be consistent with bottom-reflected paths.

- **AUV Operations**

  The dynamic nature of AUV operations is captured by the widening of the peaks for successful arrivals in Fig. 5.16. As with the VLA subset, the presence of those late arrivals at about 20-22 seconds seems to capture the effect of vehicle transmissions at deeper points in the water column. The presence of two distinct peaks in the arrival structure of the camp modem, and to a lesser degree for buoys *h2* and *h3* as well, seems to suggest that the dominant path (eigenray) for these connections is changing as the vehicle travels away from camp.

- **AUV Stalled**

  As with the modem test set, the period of time during which the AUV had stalled and got lodged under the ice effectively amounted to a static experiment. The arrival structure once again returns to tall, narrow peaks in Fig. 5.17.

5.5 ICNN TRACKING DATA

The ICEX-20 experiments showed signs of success during preliminary processing of results in the field. These included, for example, comparing the GPS data collected from the ICNN buoys with the acoustic tracking solution produced by the network for the particular buoy acting as a proxy for *Macrura* at any given point during the modem tests – these results are discussed in detail in Chapter 6. The greatest practical evidence of success, though, may perhaps come from the fact that the vehicle's final failure – the one that caused it to stall and get lodged right under the surface ice – created the need to plan recovery operations by relying solely on the solution collected from the ICNN. The vehicle track during the mission is shown in Figure 5.18, along the acoustic tracking updates produced by the ICNN. Speaking to the tracking data overall, the majority of the navigation error reported by the vehicle's onboard hydrodynamic model-aided navigation[4] (HydroMAN) system was below 15 m and was thus small enough to enable a successful recovery.[5]

The field team traveled to the last location reported by the ICNN and dug a 6-inch hole through the approximately 2 meters of ice. Through this hole, the team introduced a camera in hopes of spotting the vehicle – and indeed, they were immediately able to spot the vehicle, roughly a meter away from the current location. With the storm still coming and not enough time to complete the recovery, the team had to secure the vehicle to a surface expression they might be able to locate after the storm. By this point, the vehicle batteries had run out, so no further location updates would be available – even if the equipment extracted during the evacuation could be brought back afterwards.

Upon returning to the ice after the storm had passed, the recovery team sought out the surface expression to which they had secured the vehicle. This proved a very helpful product of planning, as the entire ice floe complex on which ice camp Seadragon was setup had moved during the storm;

[4] Randeni P., Rypkema, Fischell, Forrest, Benjamin, and Schmidt, "Implementation of a Hydrodynamic Model-Based Navigation System for a Low-Cost AUV Fleet".

[5] Randeni, Schneider, Bhatt, Viquez R., and Schmidt, "A high-resolution AUV navigation framework with integrated communication and tracking for under-ice deployments".

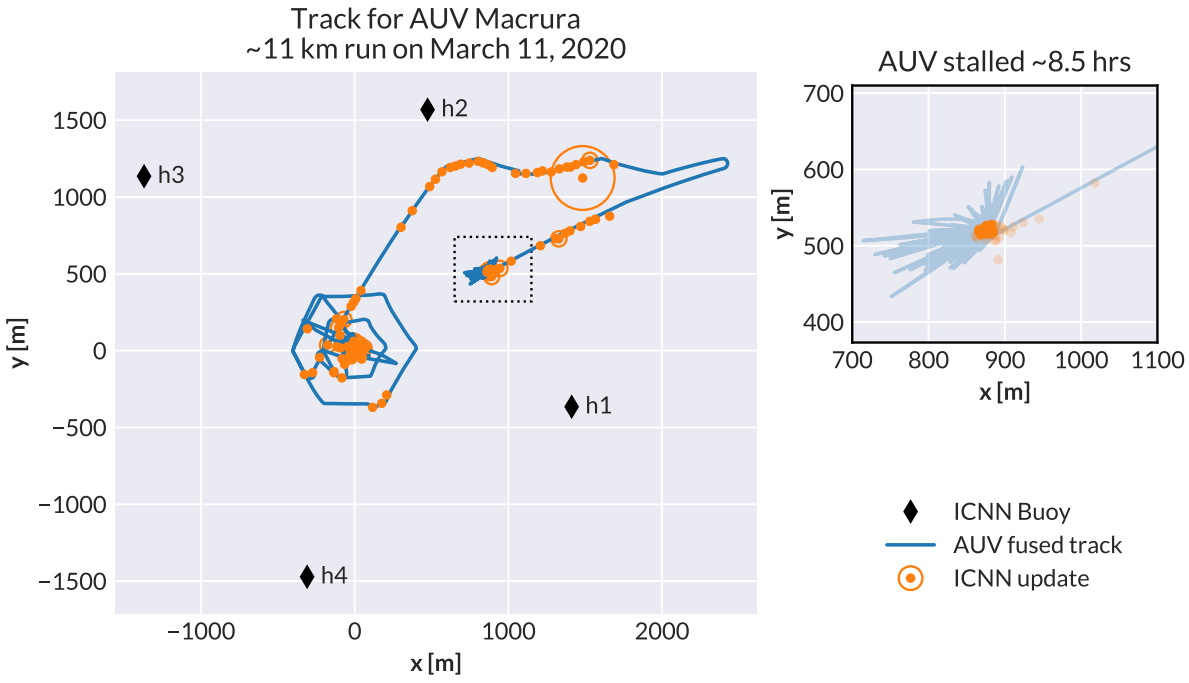
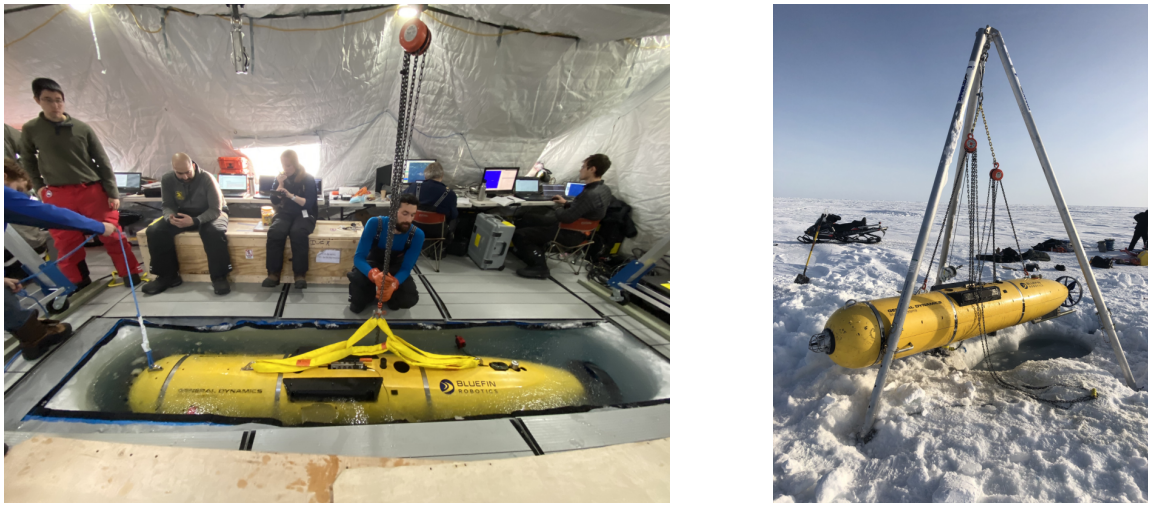


Figure 5.18: Vehicle track during an 11km run, based on the vehicle's onboard sensor fusion system. The vehicle ingests error correction updates from the ICNN, and combines them with the onboard DVL and INS measurements, as well as a self-correcting hydrodynamic model.



a) b)

Figure 5.19: ICEX-20 vehicle deployment (a) and recovery (b) sites.

the vehicle's position relative to camp had also changed significantly, as explained by LDCR Daniel Goodwin.

> *"Consider this: Our backup plan was to use the relative position and range from camp. The location that we went to recover the vehicle was roughly bearing 090° at 1100 m from camp. A significant difference than the original 045° and 1000 m. Again, without securing the vehicle topside prior to leaving the ice, finding the vehicle after the snowstorm was highly improbable."*
>
> – LCDR Daniel Goodwin, US Navy

LCDR Goodwin, who was also a member of the MIT LAMSS team during ICEX-20, represented the lab's interests to other groups involved in the recovery efforts. The planning and execution of the AUV recovery is presented in further detail in his master's thesis, "Environmental Effects of the Beaufort Lens on Underwater Acoustic Communications during Arctic Operations". As part of contextualizing the different working conditions, Fig. 5.19 illustrates the facilities at camp Seadragon (a), from which the vehicle was deployed, as well as the setup used to recover the vehicle (b) after the storm had passed.

5.6 Summary

This chapter has presented a review of field logs and framed the acquired data in the experiment timeline. The chapter also discussed the impact of logistical constraints – by forcing limited field testing prior to the experiment, limitations such as delayed inter-organization disbursements resulted in effective data loss. Building on the need to understand the extent and quality of data available, the analysis herein presented explored the distribution of the data with respect to the coordinated TDMA scheme, as well as the quality of time synchronization. The performance of the tracking range is also briefly discussed, in light of both anecdotal evidence and data logs.

As of this writing, the performance evaluation of the ICNN (summarized in Section 5.5) has been submitted for publication in "A high-resolution AUV navigation framework with integrated communication and tracking for under-ice deployments". Portions of this field report and the associated data analysis were also used to support subsequent research based on the ICEX-20 data, including the works of Bhatt ("A Virtual Ocean framework for environmentally adaptive, embedded acoustic navigation on autonomous underwater vehicles") and Goodwin ("Environmental Effects of the Beaufort Lens on Underwater Acoustic Communications during Arctic Operations").

6 Exploiting the multi-path structure

Two roads diverged in a yellow wood,

And sorry I could not travel both

And be one traveler, long I stood

And looked down one as far as I could

To where it bent in the undergrowth;

– Robert Frost, *The Road Not Taken*

In realistic environmental models, factors such as surface roughness and bathymetry will impact the acoustic field. For one, changes on the angles of these boundaries readily affect the direction of the reflected rays; Fig. 6.1 (a copy of Figure 4.10, provided locally for convenience) illustrates various paths with multiple surface bounces – these would clearly be impacted by large waves on the free ocean surface, or ice keels under the Arctic. Bathymetric features and other obstacles can also block certain rays; paths that might be observable from one receiver position can fade away as the receiver travels to a different location, which can affect a system's ability to identify distinct rays from the model and the receiver's travel time measurements.[1]

Because of the difficulty associated with ascertaining the correct relation between the modeled eigenrays and the arrivals recorded at the sensor, acoustic positioning systems typically opt for

[1] Deffenbaugh, Schmidt, and Bellingham, "Acoustic positioning in a fading multipath environment"; Deffenbaugh, "Optimal Ocean Acoustic Tomography and Navigation with Moving Sources".

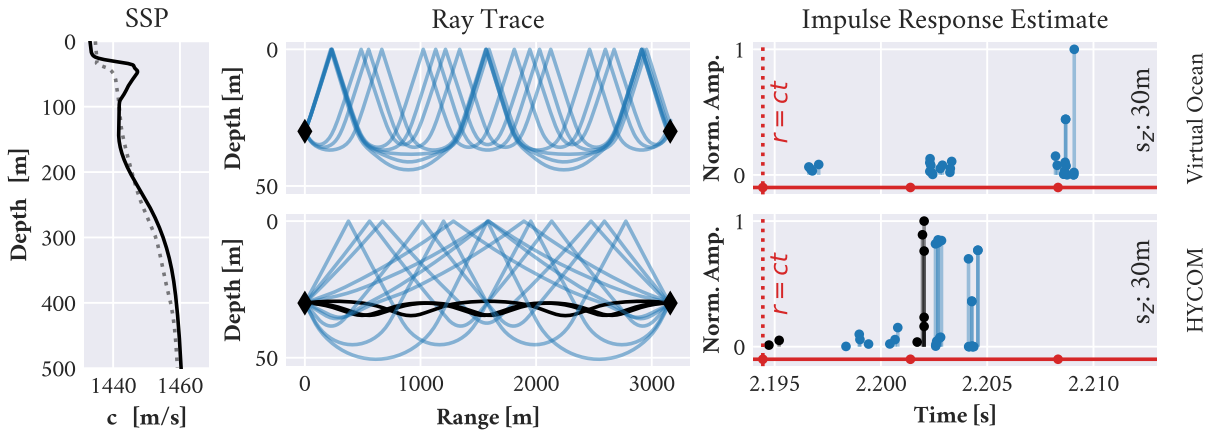


Figure 6.1: Effect of sound speed model on the ray tracing output (middle) and corresponding impulse response estimate (right) used as a basis for ranging. Sound speed profiles are taken from HYCOM (left, dashed) and the Virtual Ocean framework (left, solid). Replicates Fig. 4.10 locally, for reader's convenience.

a deterministic linear transform. Assuming an effective group velocity to relate travel time with range may incur additional error, but can be sufficient for many applications – moreover, doing so sidesteps the complexity of identifying any particular path, and the inherent need for built-in acoustic modeling capabilities that can run throughout the duration of the deployment. However, the use of well-conditioned acoustic models during runtime can improve the performance of the positioning system – particularly in the presence of features such as the Beaufort Lens, which may have a strong influence on the acoustic environment.

The core contributions of this chapter include (1) the performance evaluation of the model-aided positioning solution fielded during ICEX-20 with respect to the static modem test experiment and the buoy-to-buoy communications contained therein; (2) the development and validation of an improved algorithm for the positioning problem, based on the insights from Section 4.3 and the data analysis in Chapter 5;[2] and (3) the evaluation of these two acoustic positioning algorithms with respect to the trilateration framework used for the vehicle operations portion of the experiment.

[2] Bhatt, Viquez, and Schmidt, "Under-ice acoustic navigation using real-time model-aided range estimation".

6.1 Minimum bounce criteria

The acoustic positioning system deployed during ICEX-20 was first validated with a static experiment (modem test), and was later relied upon for the successful recovery of AUV *Macrura* after a critical system failure. This system builds on two fundamental assumptions:

1. The environmental model, which is used to predict acoustic propagation, is sufficiently representative of the real environment.

2. The signal detected by the receiver will be dominated by the most direct set of paths. In this context, the "most direct" set of paths is understood to be determined by the least number of boundary interactions.

The first of these assumptions revolves around recognizing the limitations of a deterministic approach in highly dynamic environments. The Beaufort Lens, whose effect on the acoustic environment was first presented in Chapter 1, can vary significantly within the duration of a mission; changes in its depth and other properties of the lens directly impact the associated acoustic ducts. It would be virtually impossible to anticipate all possible scenarios and identify a reliable one-size-fits-all solution for the relation between travel time and range. Therefore, the need for a representative environmental model during the ICEX-20 deployments was handled by using the EOF framework discussed in Chapter 4, given its ability to update the virtual ocean model from CTD measurements.

The second assumption stems from the fact that producing high-resolution acoustic models during runtime may not always be feasible within the functional requirements of the mission. As part of the positioning exercise, reliable updates to the predicted arrival structure are generally needed within the span of the communications cycle – meaning at the rate of a couple of times per minute, at a minimum; some of the predictive tools deployed by LAMSS may require a faster update cycles. In order to meet these timescale needs, shown earlier in Fig. 4.2, the faster stages of the system are

generally built from plane-wave expansion of local grid solutions produced by the acoustic propagation model at slower rates. This nesting approach is extremely valuable to real-time operations, as these updates must be computed for each of the network's possible contacts of interest, such as the various vehicle-buoy combinations.

Thus, rather than seeking a fully deterministic relation between modeled eigenrays and field measurements, the ICEX-20 approach first selects a subset of the rays in each point-to-point model (for each of the contacts between nodes, such as vehicle to buoy #1 or #2). These are bundled based on an expected commonality, which in this case corresponds to the number of boundary interactions. For each contact scenario, the corresponding set of rays is consolidated into a single metric: a prediction for the effective group velocity u. This metric is obtained by way of a power weighted average, as shown in Eq. 6.1, and is assumed to be smoothly varying with respect to range. The corresponding weighted standard deviation is given in Eq. 6.2.

$$u = \frac{\hat{r} \sum_{n=1}^{N_0} a_n^2}{\sum_{n=1}^{N_0} dt_n a_n^2} \tag{6.1}$$

$$\sigma_u \simeq \sqrt{\frac{\sum_{n=1}^{N_0} (dt_n - \hat{r}/u)^2 a_n^2}{\sum_{n=1}^{N_0} a_n^2}} \frac{u^2}{\hat{r}} \tag{6.2}$$

During the ICEX-20 operations with the ICNN, the vehicle reported its estimated position and depth, along with any other data packets it transmitted during its TMDA slot. Upon receiving these updates, the ICNN evaluated the contents of the message as well as the perceived one-way travel time. The position data transmitted by the vehicle was used approximate the range $\hat{r}$ between the vehicle and each receiving buoy, for each contact. This range was then used to perform the plane-wave expansion of the acoustic model, along with the vehicle's depth. This computation resulted in predicted arrival times dt_n and amplitudes a_n for each of the $n$eigenrays expected to connect the transmitter and receiver nodes. If any of the eigenrays were direct paths ($N_0 > 0$), then only those

rays were accounted for. If no direct paths were predicted by the model ($N_0 = 0$), then the N_1 single-bounce rays were considered next, only moving upward in number of boundary interactions when the lower-count ones are unavailable – therein comes the name chosen for this metric: the Minimum Bounce Criteria (MBC). Finally, the pseudo-range between two nodes i, j was calculated simply by using the effective group velocity $u_{i,j}$ in the linear relation (Eq. 6.3).

$$r_{i,j} = u_{i,j}\Delta t_{i,j} \tag{6.3}$$

6.1.1 Modem-to-modem connections

In order to validate its use for under-ice operations, the MBC was first evaluated during the modem test experiment. Figure 6.2 illustrates the various transmitter-receiver connections recorded during this static experiment. Per the presentation of the ICNN infrastructure in Section 4.1.3, each buoy has a single transmitter at a fixed depth of either 30 or 90 meters; all buoys have two receiver locations at 30 and 90 meters, and have the ability to select the reception depth layer. Thus, Figure 6.3 complements the modem connection counts with information about the depth layers corresponding to each of the transmission (Tx) and reception (Rx) events reported by each node.

In addition to showing the Tx/Rx activity by depth layer, Fig. 6.3 also shows the group velocities estimated by the topside computer during the ICEX-20 deployments. These estimates were based on the MBC (Eq. 6.1), and are reported here by their receiving node. The performance statistics for both sets – using either the baseline or fitted SSP – are shown in Table 6.1. These values are computed from the logged data and group velocity estimates only, and receptions with anomalous values have been filtered out.

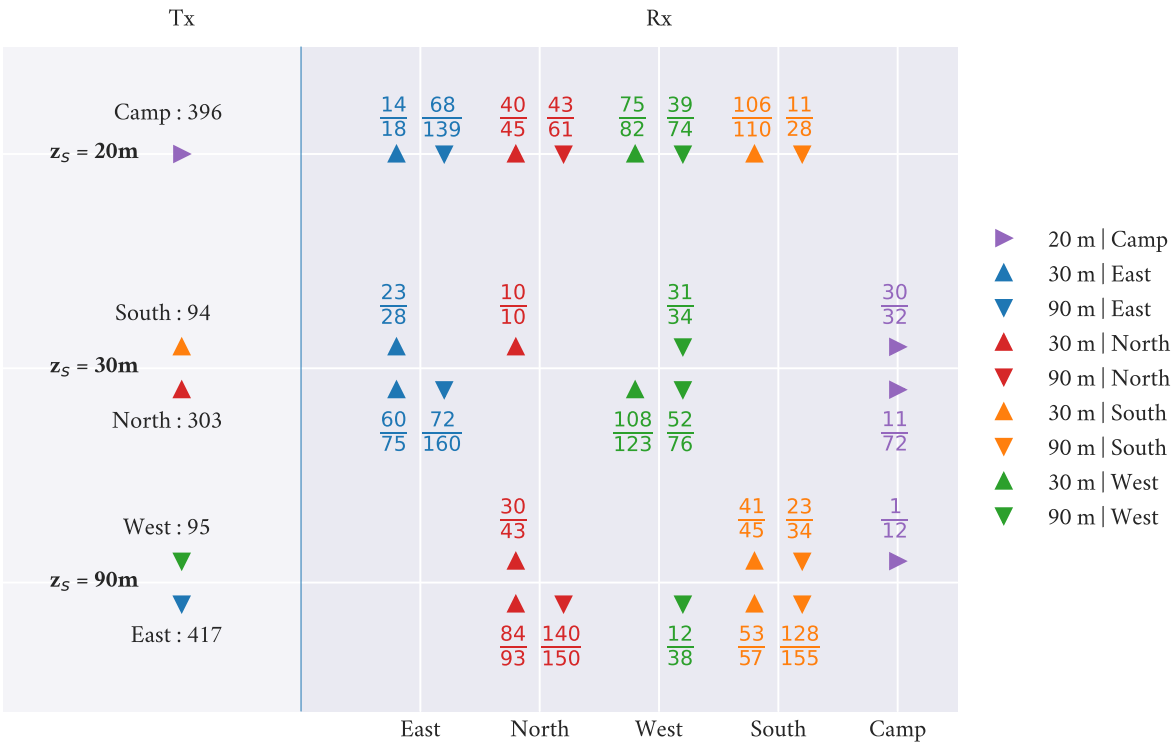


Figure 6.2: Overview of the modem connections recorded during the modem test experiment. Reception event counts are shown as fractions, with the number of successfully decoded entries on top and the total number of detections on the bottom.

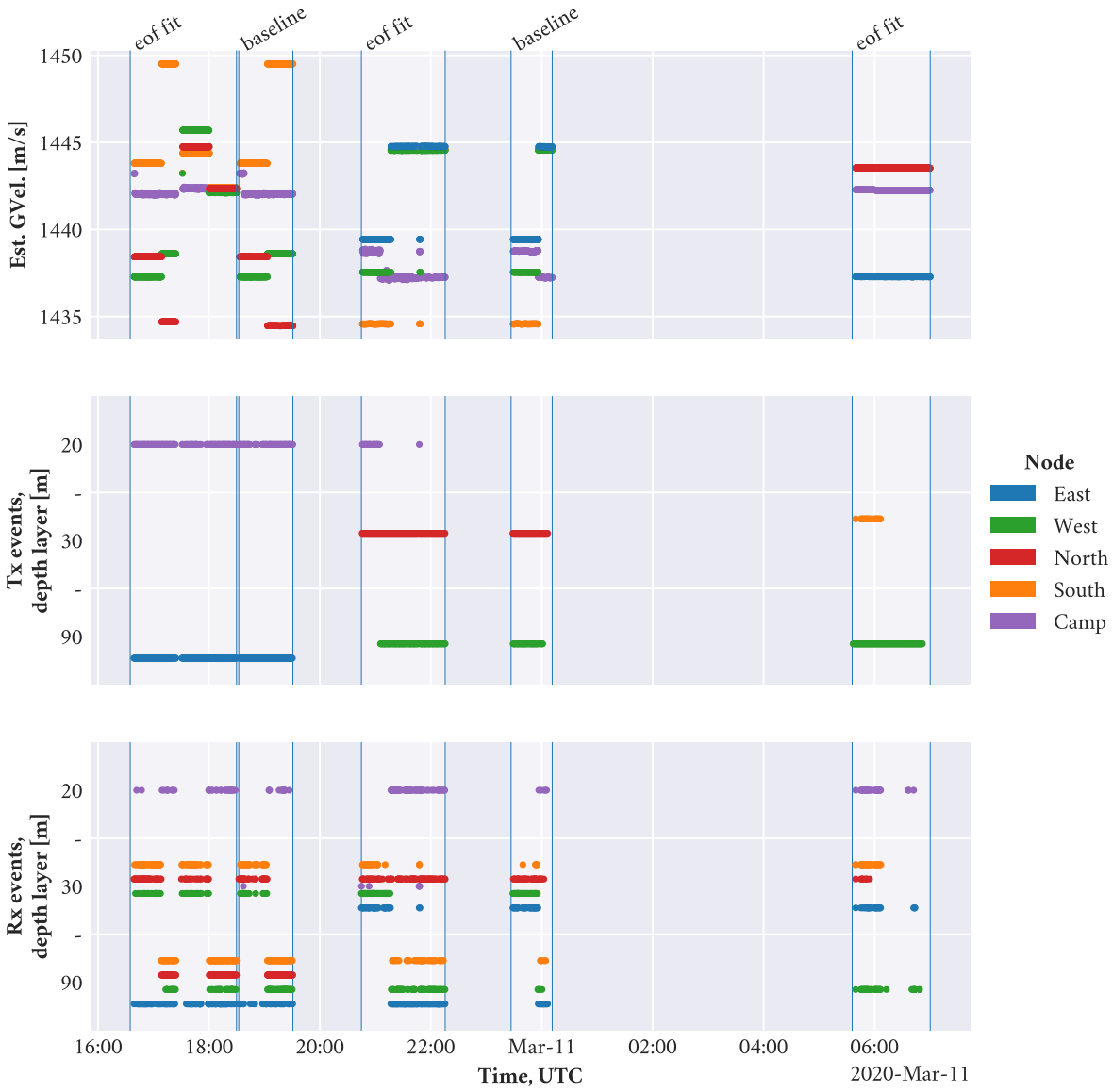


Figure 6.3: Timeline of events and configuration during the modem test experiment. The group velocity estimates are based on either the baseline sound speed profile, or one obtained by fitting the model to a CTD cast with the EOF framework. Transmitter and receiver activity is categorized by depth layer, with the camp modem lowered to approx 20 meters depth, and the buoys operating at either 30 or 90 meters.

Table 6.1: Range estimation error for events recorded during the modem test experiment, based on data and predictions produced *in situ*. Metrics are reported with respect to the absolute error.

SSP	Count	Mean [m]	Median [m]	STD [m]	Max [m]
Baseline	247	11.35	11.96	4.23	23.95
EOF fit	575	11.31	11.20	8.09	25.99

6.1.2 On recorded data vs bottom-bounce predictions

One of the patterns regarded as anomalous consisted of events where the predicted group velocity and recorded travel time produced an unusually large error with respect to the GPS-derived range. This occurred with events that are considered consistent with bottom-bounce paths, and were only recorded during portions of the experiment where the topside computer was configured to predict group velocities with respect to the baseline profile. These events appeared only rarely in the dataset, reporting travel times of 4.19 seconds for transmissions originating at the eastern-most buoy and recorded by the western-most buoy. For consideration, the error metrics for these events alone, with respect to the associated GPS-derived ranges, are shown in Table 6.2.

Table 6.2: Range estimation error for events recorded during the modem test experiment, which may be considered consistent with bottom-bounce paths. Metrics are based on data and predictions produced *in situ*, and are reported with respect to the directional error.

SSP	Count	Mean [m]	Median [m]	STD [m]	Max [m]
Baseline	11	2865.54	2865.61	0.24	2865.88

The two nodes involved in these events are located approximately 3162 meters apart from one another. Using a group velocity estimate related to a direct path in the models would nearly double the estimated range, as the predicted travel time (not used by the MBC) would be smaller than the

recorded measurement, thus producing the error reported in the previous table. As a first-order approximation for these likely bottom-bounce arrivals, accounting for both the GPS-tracked range between the buoys and the local bathymetry of approximately 2700 meters would produce a much more reasonable estimate of the signal's effective speed through the water column.

There is one final remark worth making, on the insight attainable from the mission logs alone. In balancing competing computational requirements during the missions, the experimental system was configured to produce ongoing estimates of the group velocity only for acoustic links that included the vehicle. Per the principle of reciprocity, this meant the estimates logged corresponded exclusively to models of the vehicle transmissions – or the virtual vehicle proxied by one of the physical buoys, for the modem test experiment. This decision stemmed from the expectation that group velocity estimates would be used exclusively as part of the ICNN's positioning step. Since the buoys reported their GPS-tracked position to the operations center via radio, the buoy-to-buoy group velocity estimates were not used for positioning and were therefore not computed or logged.

6.2 Nearest bounce criteria

Though rarely present in the modem test data logs, the handful of arrivals that appeared consistent with bottom-bounce paths stood out for the significant amount of error they produced with respect to the initial range estimate $\hat{r}$ used in the Minimum Bounce Criteria framework. Drawing from this insight, it follows that if bottom-bounce paths in the models were preserved for comparison under appropriate circumstances, then the model's predictions of the group velocity could be used as part of the ranging framework. Indeed, the same argument could be extended to the various combinations of surface and bottom interactions that may occur in environments such as the Arctic Ocean, particularly when recognizing that an otherwise viable path could be blocked by some external obstacle of which the acoustic model has no knowledge. Parting from this observation comes the Nearest Bounce Criteria, which aims to strike a balance between the need for more gran-

ular interpretations of the acoustic models and the reluctance to pursue a more detailed eigenray identification approach in real time, in the field.

Like the Minimum Bounce Criteria (MBC), the Nearest Bounce Criteria (NBC) uses a power-weighted average of the various arrivals predicted by the acoustic propagation model. The main difference between the two is that the NBC categorizes its various predictions by some choice parameter; in this case, the number of boundary interactions tracked by the acoustic model. Thus, the predicted travel time t_k corresponds to the power weighted average of the ray travel times for the N_k rays with k bounces (Eq. 6.4). A given arrival reported by the physical modem is associated with a number of bounces by minimizing the travel time error (Eq. 6.5).

$$t_k = \frac{\sum_{n=1}^{N_k} dt_n a_n^2}{\sum_{n=1}^{N_k} a_n^2} \tag{6.4}$$

$$t_{i,j,k} = \min_{k=0,1,2,\ldots} \left| t_k - \Delta t_{i,j} \right| \tag{6.5}$$

The effective group velocity $u_{i,j}$ associated with the closest travel time match $t_{i,j,k}$ is then used to produce the pseudo-range between two nodes i, j by using the linear model, as before (Eqs. 6.6, 6.7).

$$u_{i,j} = \frac{\hat{r}}{t_{i,j,k}} \tag{6.6}$$

$$r_{i,j} = u_{i,j} \Delta t_{i,j} \tag{6.7}$$

From an operations perspective, one of the advantages of the NBC over the MBC deployed during ICEX-20 is that it attains more granular control over the interpretation of the acoustic models, without requiring any changes at all on the implementation of the models on the AUV or the oper-

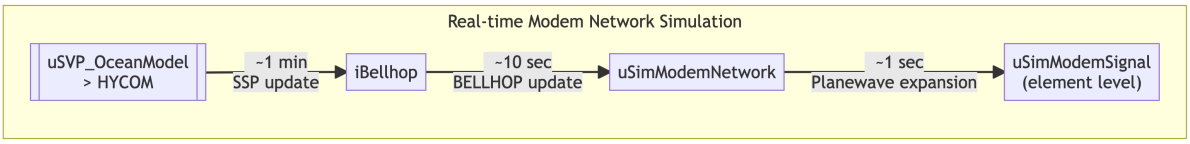


Figure 6.4: Runtime simulations are based on the Virtual Ocean Simulator, which uses a nested modeling approach that exploits the timescales of change of the different features. The acoustic model is evaluated at an intermediate timescale, producing data for a local grid near the target node. The element-level impulse response estimate is computed by plane wave expansion of this pre-computed grid.

ator station. Instead, it seeks to exploit information already available within the vehicle's autonomy system, updating only how the model output is processed for comparison with field measurements.

6.2.1 Validating the nearest bounce criteria

The NBC was not implemented during the ICEX-20 experiments, but was instead a product of insight attained during the data analytics efforts that followed. Consequently, its validation necessitated a post-processing approach that allowed the augmentation of field logs, while remaining within the bounds of vehicle operations. In other words, the post-processing simulations sought to approximate the simulations performed in real-time throughout the mission.

The main driver for pursuing an approximation, rather than a full recreation of runtime simulations, is that the runtime system relies on a nested approach (Fig. 6.4), with intermediary data products from slower layers feeding into the faster ones. Rather than replicating all layers of this nested approach, then, the validation framework captured only the salient features of runtime processes. As with the runtime acoustic model, a local grid was used to predict the arrival structure; but absent the constraints of real-time operation to inform the construction of this mesh, a set of 11 x 11 points centered at the target node and spanning 10 meters in range and 20 meters in depth was used instead, for each data point evaluated. The plane wave interpolation to the target coordinates was also disregarded; in its stead, the arrival predictions across the grid were processed in bulk.

The use of a local grid in this scenario was not only intended to recognize the operational framework. It was also aimed at minimizing the impact of numerical resolution in the propagation model. This limitation becomes apparent when using a single receiver point rather than a grid, as the conventional approach to ray tracing starts from a set of launch angles; the resulting traces may represent paths that would reach a vehicle in a real application, but might not be recognized by the numerical solver as a viable detection. Building on this consideration, the 2:1 ratio between the vertical and horizontal spans covered by the local grid was driven by the expected spread of the rays in both directions within the range of operations. This point is discussed further in Sec. 6.2.3, where details about the ray-tracing models for the different environmental models are shown.

6.2.2 Performance of the nearest bounce and minimum bounce criteria

The NBC validation process was limited to evaluating paths with up to 4 bounces. This decision was informed by the ranges and depths of operations involved in the ICEX-20 deployments, as well as the expected attenuation related to multiple boundary interactions. Furthermore, surface and bottom bounces were not accounted for separately; accounting for the fan of angles evaluated by the acoustic propagation model and the generally upward-refracting nature of the surface duct in the environments evaluated during this assessment, it was expected that there would be little overlap between single surface bounce and single bottom bounce paths. On the point of environmental models, three were used as part of this assessment: the baseline sound speed profile in the EOF framework, and the EOF fit of experimental CTD data, were supplemented by a third sound speed profile taken from HYCOM.

From all the events recorded during the modem test experiment, shown previously in Fig. 6.2, only those that were successfully decoded by the receiving node were used in this performance evaluation of the two proposed performance metrics, MBC and NBC. After all, the ICNN was likewise configured to distrust failed receptions as part of the collaborative navigation solution. For the

purposes of producing a balanced standard for comparing the two metrics, entries whose measured travel times was consistent with bottom-bounce paths were disregarded. The same was done for events where the simulated MBC prediction was likewise locked onto a bottom-bounce path – this point is discussed further Section 6.2.3.

Table 6.3: Range estimation error for 1232 acoustic communication events recorded during the modem test experiment, using supplementary simulations. Performance improvements of the Nearest Bounce Criteria over the Minimum Bounce Criteria approach an order of magnitude. Metrics are reported with respect to the absolute error.

Criteria	Metric	Baseline SSP	EOF fit	HYCOM
Minimum Bounce				
	Min [m]	0.01	0.00	0.11
	Median [m]	10.24	13.23	6.34
	Mean [m]	10.35	13.34	7.70
	Max [m]	22.52	31.49	19.55
	STD [m]	6.71	8.44	5.14
Nearest Bounce				
	Min [m]	0.00	0.00	0.01
	Median [m]	2.27	2.13	4.49
	Mean [m]	4.00	4.16	5.17
	Max [m]	14.96	20.21	15.81
	STD [m]	4.02	4.99	3.53

In all, 1232 events were isolated from the experimental data and evaluated in simulation as previously described, with a 10m by 20m grid centered at the target node's range and depth. The range estimation error statistics are shown in Table 6.3, where the Nearest Bounce approach is shown to perform better than the Minimum Bounce approach across all three environmental models. Going

by the median error on the more realistic models in the EOF framework[3], these simulations indicate that the scale of improvement attainable by tracking the model's boundary interactions may approach an order of magnitude – with the baseline SSP, for example, the median error goes from 10.24 meters to 2.27 meters. Although the gains are smaller with the generally smoother HYCOM profile, the nearest bounce criteria nonetheless outperforms the MBC in this third environmental model.

6.2.3 A case study of the performance improvement

At the beginning of Sec. 6.2 it was stated that the presence of receptions consistent with bottom bounces in the data logs played an important part in motivating the improvements to the minimum bounce criteria deployed during ICEX-20. Yet, these same bottom-bounce arrivals were disregarded in the preceding performance analysis, in order to provide an equitative comparison between the two algorithms. This section dives deeper into the decision to disregard bottom-bounces in the comparative metrics, expanding on the measurement-driven example from Section 6.1.2 by presenting a choice case study from the simulation data to illustrate the improvements afforded by the Nearest Bounce Criteria.

The values provided in Table 6.2 were based on *in-situ* simulation data collected from the ICEX-20 logs, compared with the GPS-based range. These acoustic events, and the associated model-aided predictions, are herein revisited under the post-processing simulation framework previously presented. This is done in order to enable the comparison with respect to the different environmental models; additionally, the post-processed simulations are needed to supplement field measurements, since no real-time simulation data is available in the logs for the events connecting the North and South buoys (see Table 6.4).

[3]The EOF framework used for ICEX-20 was based on WHOI Ice-Tethered Profiler data in the region.

Table 6.4: Event data collected from the ICEX-20 modem test data logs for two buoy-to-buoy connections: between (1) the East and West buoys, and (2) the South and North buoys. Note that *in-situ* simulation data is only available for the 11 events between the East and West buoys; the predicted one-way travel time (OWTT) for these events is roughly half of the measured value.

	East to West	South to North
Field measurements		
Number of events	11	10
Mean OWTT, measured [s]	4.19	2.20
Mean GPS range [m]	3161.90	3139.58
Logged simulations		
Mean sim. range [m]	3161.82	N/A
Mean sim. OWTT [s]	2.20	N/A
Mean sim. group vel. [m/s]	1438.60	N/A

The number of bounces predicted for each environment, along with the corresponding mean and standard deviation of the range estimation error, are shown in Table 6.5 for the East-West pair, and in Table 6.6 for the North-South pair. For the East-West pair, the Minimum Bounce Criteria overshoots the true range by nearly 3 kilometers – this error is consistent with the difference between modeled and measured travel times in Table 6.4, adjusted by the predicted group velocity. For the North-South pair, the opposite effect is observed: for the CTD-driven environment, based on the EOF fit, the MBC predicts a much shorter range than the one produced from GPS data.

Table 6.5: Range estimation error for the 11 events recorded traveling from the easternmost buoy to the westernmost buoy during the modem test experiment. Metrics are reported with respect to the directional error.

Criteria	Metric	Baseline SSP	EOF fit	HYCOM
Minimum Bounce				
	Bounces	0	0	0
	Mean [m]	2882.93	2892.62	2878.60
	STD [m]	0.24	0.24	0.24
Nearest Bounce				
	Bounces	2	2	2
	Mean [m]	-28.01	-26.35	-30.37
	STD [m]	0.08	0.08	0.08

Table 6.6: Range estimation error for the 10 events recorded traveling from the southernmost buoy to the northernmost buoy during the modem test experiment. Metrics are reported with respect to the directional error.

Criteria	Metric	Baseline SSP	EOF fit	HYCOM
Minimum Bounce				
	Bounces	1	1	0
	Mean [m]	15.39	-1490.66	11.90
	STD [m]	0.57	0.40	0.58
Nearest Bounce				
	Bounces	3	4	2
	Mean [m]	2.38	0.76	8.29
	STD [m]	0.59	0.57	0.58

The source of these errors becomes apparent when tracking the number of boundary interactions, as proposed by the NBC, when assimilating the results of the acoustic propagation model. Table 6.7 shows the travel time predictions produced by the acoustic model, with respect to the number of bounces. The shortcoming of the central assumption behind the MBC should become apparent here; for the link between the East and West buoys, the MBC assumes the signal will be dominated by a direct path with a travel time of 2.188 seconds, although the field records capture 11 events that are better approximated by a 2-bounce path with a travel time of 4.225 seconds. Similarly, the 2.196-second link between North and South buoys appears to be better described by the model's 4-bounce path than by a 4.181 second single-bounce path.

Table 6.7: Predicted travel times by number of boundary interactions (bounces), for the EOF fit environment.

	East to West	South to North
Field measurements		
Number of events	11	10
Mean GPS range [m]	3161.90	3139.58
Mean OWTT, measured [s]	4.190	2.196
Predicted travel time [s]		
Direct path (0 bounces)	2.188	N/A
1 bounce	2.202	4.181
2 bounces	4.225	2.183
3 bounces	N/A	2.188
4 bounces	N/A	2.195

6.2.4 Comparison with an eigenray-search approach

The ultimate goal of the MBC and NBC algorithms is to provide a reliable, physically intuitive interpretation of the acoustic propagation models, without taking on the added burden of regularly identifying specific paths that may connect any given source-receiver pair in the network. However, as part of validating the algorithm's fundamentals in this chapter, it is nonetheless helpful to perform the eigenray search to verify the results presented in Sections 6.2.2 and 6.2.3. Thus, Figures 6.5-6.7 illustrate the eigenrays with the closest-matching travel times, for each of the buoy-to-buoy connections captured during the static modem test.

As shown in Figure 6.6, the 4-bounce prediction produced by the power-weighted NBC for the North-South link, under the second environment (the EOF fit; Table 6.7), is consistent with an eigenray that connects these two nodes and exhibits a quadruple surface bounce. The figures do not show bottom-bounce paths due to the steeper angles at which these travel; however, these paths do appear in the simulations as well. For the North-South link, with the source at 30 meters depth, the simulations include rays with a single bottom bounce which would match the predicted travel time of 4.181 seconds. With the source at 90 meters, as is the case for the East-West link, rays with surface-bottom and bottom-surface bounces (for 2 bounces total) are also present.

6.3 Effect of the NBC on the navigation paradigm

The operational paradigm of the ICNN consists of using acoustic positioning to compute correction values relative to the position estimates transmitted by the vehicle, rather than transmitting the trilaterated positions themselves – this stems from the need for a compact representation of the most useful information, as it must fit within the limited throughput of acoustic communications along with any other pieces of information required for the mission. The particular approach to acoustic positioning used in the ICNN benefits from the model-aided range estimation capabilities of the

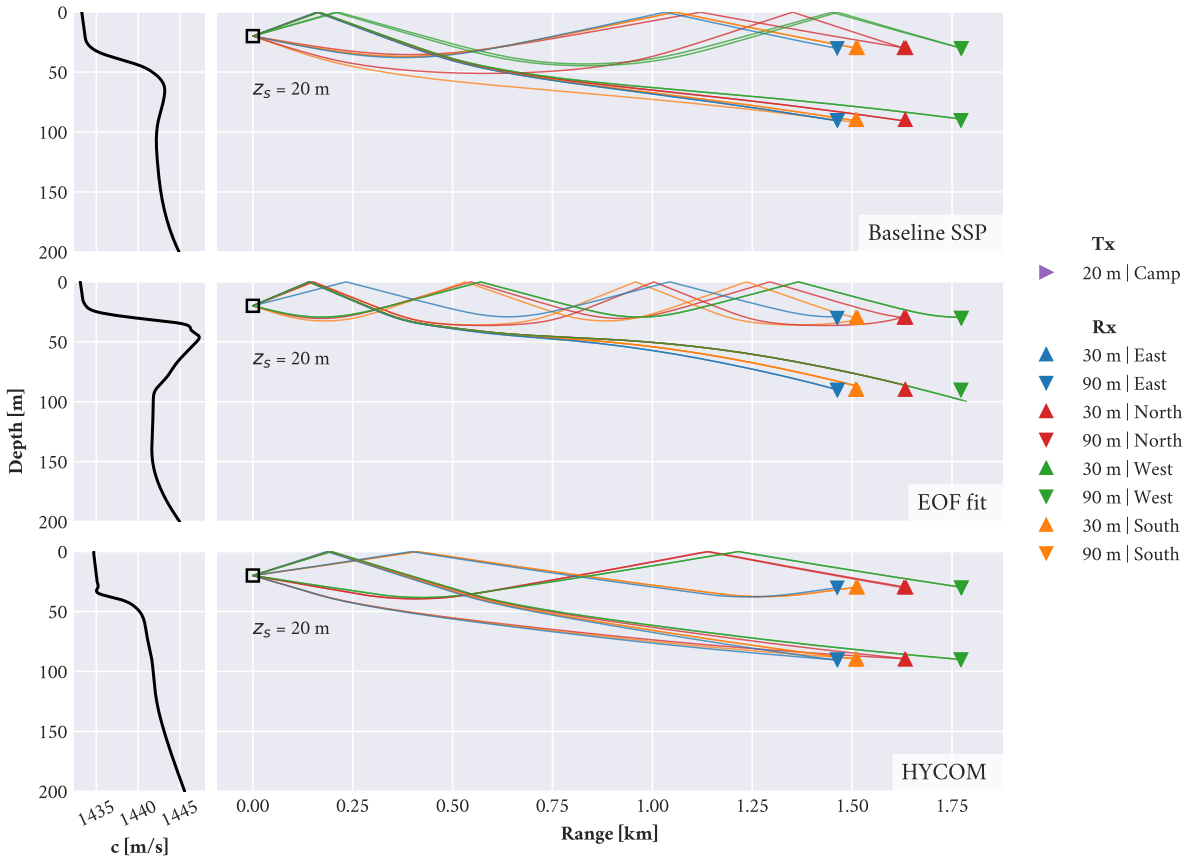


Figure 6.5: Ray traces for a source depth of 20 meters (Camp), under different environmental models. The eigenrays produced by ray tracing are matched to field measurements by the estimated travel time, to illustrate the most likely paths taken by the acoustic signals. Bottom bounce paths are not shown.

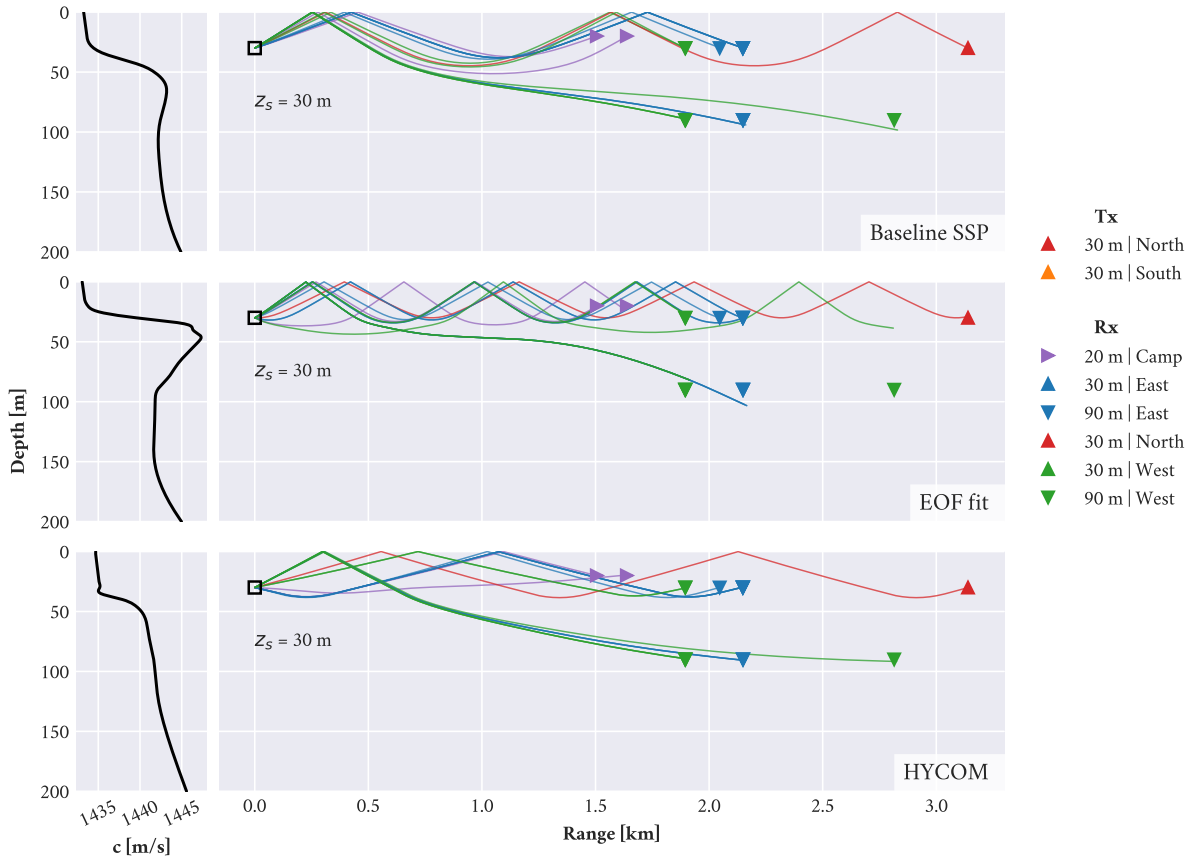


Figure 6.6: Ray traces for a source depth of 30 meters (North and South buoys), under different environmental models. The eigenrays produced by ray tracing are matched to field measurements by the estimated travel time, to illustrate the most likely paths taken by the acoustic signals. Bottom bounce paths are not shown.

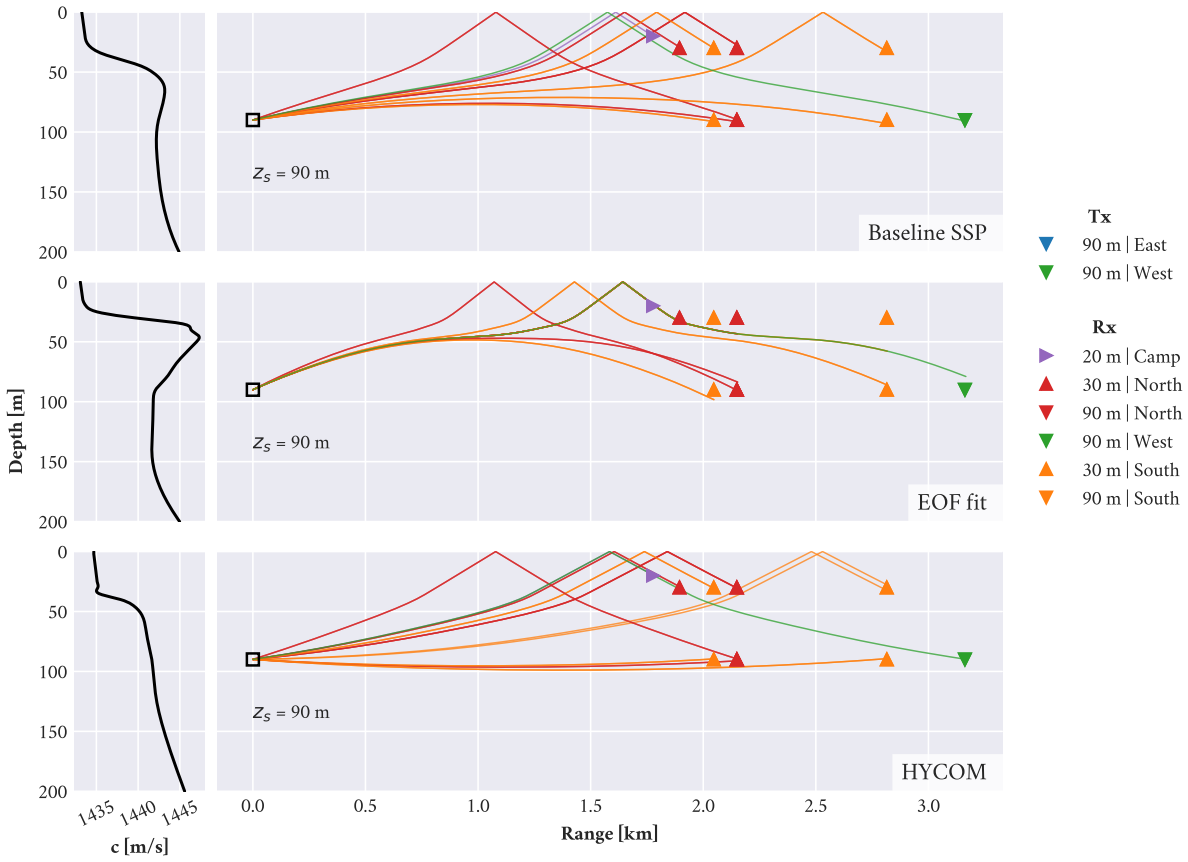


Figure 6.7: Ray traces for a source depth of 90 meters (East and West buoys), under different environmental models. The eigenrays produced by ray tracing are matched to field measurements by the estimated travel time, to illustrate the most likely paths taken by the acoustic signals. Bottom bounce paths are not shown.

virtual ocean simulator, since the reliability of the trilateration solution – and the correction terms computed thereafter – depends on how accurately the travel time measurements are converted to range estimates. As the environmental and acoustic propagation models become better representations of the real ocean, the lower the expected trilateration error will become. This relation between the ranging and positioning problems becomes essential when discussing the performance of the proposed algorithms with respect to vehicle operations.

Contrary to the modem test data discussed in the preceding sections, the datasets from AUV operations do not include GPS data for the submerged vehicle; no "ground truth" position information is available to validate the results. Thus, this section aims to bridge that gap by reevaluating the performance of the MBC and NBC algorithms with respect to the trilateration results produced for the same modem test experiment previously discussed, in such a way that the analysis may later be extended to include data from AUV deployments.

6.3.1 Baseline performance : *in-situ* trilateration

Expanding on the results briefly introduced in Section 5.5, Figure 6.8 illustrates the distribution of vehicle position corrections recorded during the modem test experiment, as part of ICEX-20. These corrections are separated both by the depth of the virtual vehicle (physically embodied by one of the ICNN buoys), and the active reception layer configured for the remaining buoys at each event.

As part of a generally conservative approach to vehicle operations in ICEX-20, the ICNN trilateration routine started with a default group speed estimate of 1430 meters per second. This value was provided to the tracking program as part of the mission configuration, and was intended to serve as a first approximation in line with the industry standard of a constant-valued time-range conversion coefficient. When the virtual ocean had produced a recent estimate of the acoustic propagation, however, the model-aided range estimation was preferred. These conversions were tracked independently for each transmitter-receiver pair – although the group speed estimate was expected to be

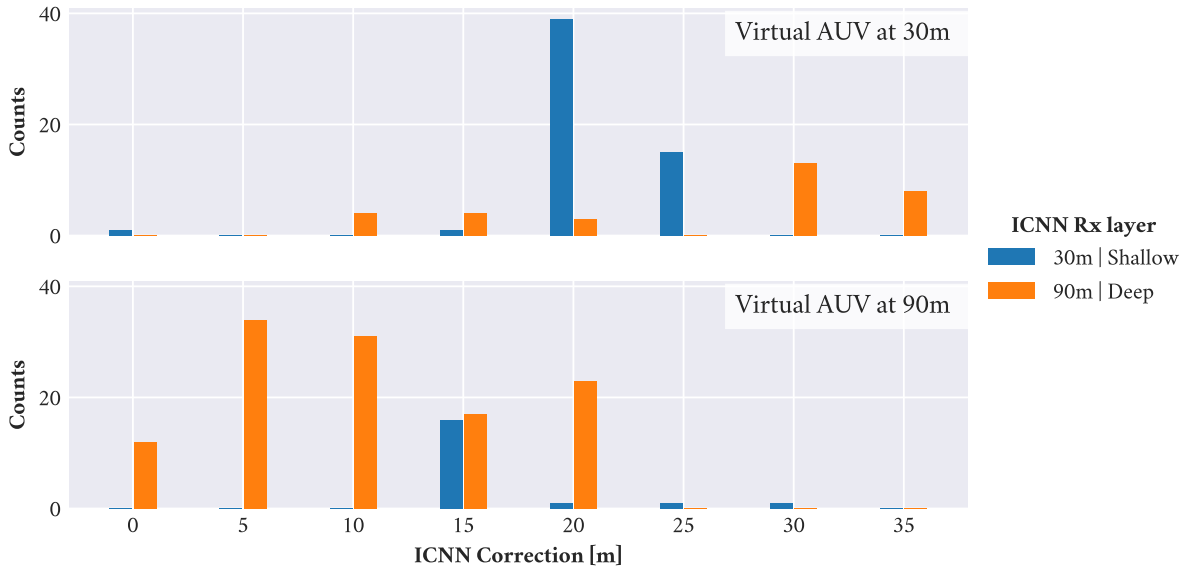


Figure 6.8: Distribution of vehicle position corrections computed by the ICNN during the modem test experiment. The corrected position estimates are the trilateration results computed by the ICNN, which are in turn derived from the recorded times of arrival and respective model-aided range conversions; position corrections are then transmitted as marginal differences with respect to the position estimate transmitted by the vehicle.

locally smooth near a given receiver, no such assumption was enforced for larger range differences related to distinct acoustic links.

Table 6.8: Trilateration data for the example scenario shown in Figure 6.9, based on the Minimum Bounce Criteria and data collected during the modem test experiment. The estimated ranges used for the propagation model are based on the latest position estimates received from the virtual vehicle at the time of model execution.

	h1 – East	h3 – West	h5 – Camp
Model data			
Estimated range [m]	2149.01	1895.85	1636.91
Predicted travel time [s]	1.487	1.312	1.139
Group Velocity [m/s]	1444.76	1444.54	1437.24

	h1 – East	h3 – West	h5 – Camp
Field data			
Measured travel time [s]	1.502	1.329	1.140
Range conversion [m]	2170.42	1919.84	1638.73
True range (GPS) [m]	2149.43	1894.76	1633.00

Thus, Figure 6.9 illustrates one such instance collected from the ICEX-20 data logs, where the ICNN tracker is relying on different model-driven velocity estimates to produce the ranges fed to the trilateration solver. Here, the arrival times reported by the different buoys are converted to travel times based on the TDMA schedule; the travel times are then converted to estimated ranges and the ranging circles are drawn centered at each receiver. If the range conversions are adequate, then the intersection of all circles should indicate the location of the source (see Sec. 2.4 for more information on positioning). The data associated with the arrival events, shown as range projections in the figure, is provided in Table 6.8. The magnitude of the resulting correction for this event was of 20.4 meters. The trilateration error, taken as the root-mean-square of the ranging errors for each acoustic link, was of 11.6 meters.

All the arrivals used for the event shown in Figure 6.9 corresponded to successful receptions. Thus, the initial position estimate used in the calculation was based on the node report contained in that same transmission. It should be noted that, although the virtual vehicle was connected to one of the buoys, all vehicle software was in fact running on a separate computer; the ICNN did not have access to that buoy's GPS data stream, for this particular event or any others recorded throughout the experiment, save for what was received through acoustic communications. The static nature of the experiment, along with the fact that the virtual vehicle reported its position based on its buoy's GPS data, meant that the initial estimate transmitted to the ICNN was in fact aligned with

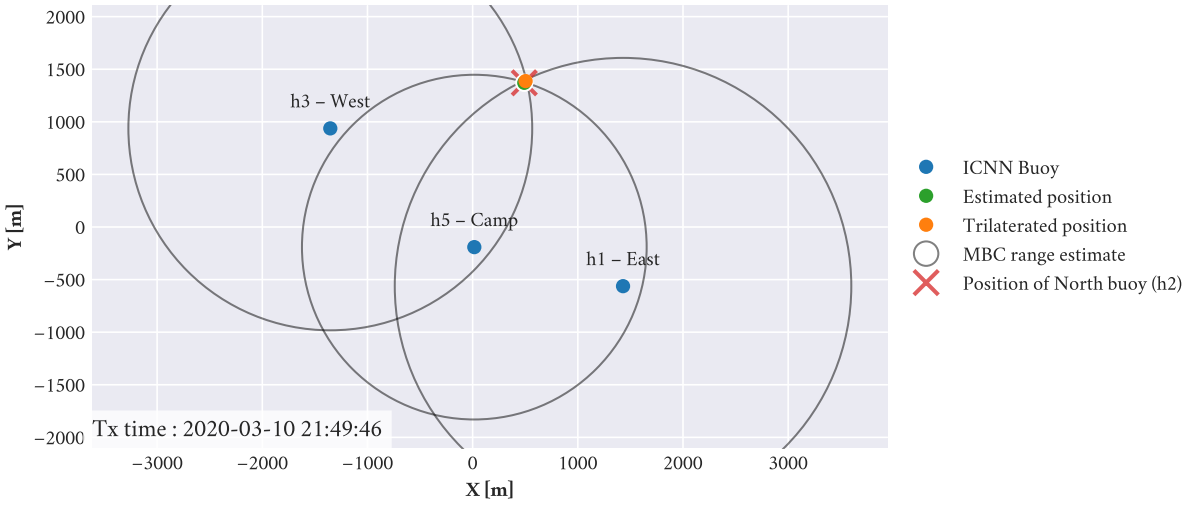


Figure 6.9: Acoustic positioning via trilateration, using model-aided range estimation. The reception events shown here occurred during the modem test experiment, while the northernmost buoy (h2) was acting as the virtual vehicle. The event contains a viable 3-beacon solution with a correction of 20.4 meters from the initial position estimate transmitted by the virtual vehicle. The associated trilateration error had a magnitude of 11.6 meters.

ground truth. Therefore, the distribution of corrections in Figure 6.8 effectively reflects positioning accuracy.

In more general terms, there is certainly a remarkable difference between these two terms – the magnitude of the corrections and the positioning accuracy. If the initial point of reference provided to the algorithm is far away from ground truth – a scenario that is expected to occur occasionally while the AUV is underway – then a larger correction may be necessary to produce a highly accurate result. Given a sufficient number of data points from which to perform the trilateration, the positioning accuracy may thus be better described by the trilateration error, which was previously defined as the root-mean-square of the remaining ranging errors for each acoustic link.

The caveat of requiring a sufficient number of data points is key to the usefulness of the error metric. The discussion of results obtained from the modem test data set has focused on the magnitude of corrections because the majority of entries recorded during this particular subset of the experiments were 2-beacon solutions. A total of 264 entries of this kind appear in the logs; by com-

parison, only 22 3-beacon solutions were recorded during the experiment, along with 2 4-beacon entries.

When limited to only two ranging circles for consideration, the trilateration algorithm reports one of the intersection points as the final answer. The initial position, taken from the vehicle's transmission, is used only as a tie-breaker to pick the closer of the two intersection points; but it has no influence on the reported error. Instead, computing the trilateration error with respect to the chosen intersection of two circles would yield a value of zero, as the solution is exact. In other words, the trilateration error term is not particularly helpful in understanding the reliability of a 2-beacon solution. This topic will be revisited later in this chapter, in the context of AUV operations.

6.3.2 Comparing the different ranging approaches

Figures 6.10 and 6.11 illustrate the performance of the positioning system with respect to both algorithms, the Minimum Bounce Criteria and the Nearest Bounce Critera, in a 3-beacon scenario. The MBC is represented in two ways: (1) the implementation fielded during ICEX-20, which included the local plane wave expansion of the gridded solutions posted by the acoustic model, and (2) the simplified form used as part of the validation work presented in Sections 6.1 and 6.2. With more than two entries to evaluate, the positioning task in these events becomes an over-constrained problem; the trilateration error becomes representative of how closely the various ranging estimates capture the optimal solution. The numerical results for both cases are provided in Tables 6.9 and 6.10 respectively. Note that the second event shown here is the same event shown previously in Figure 6.9.

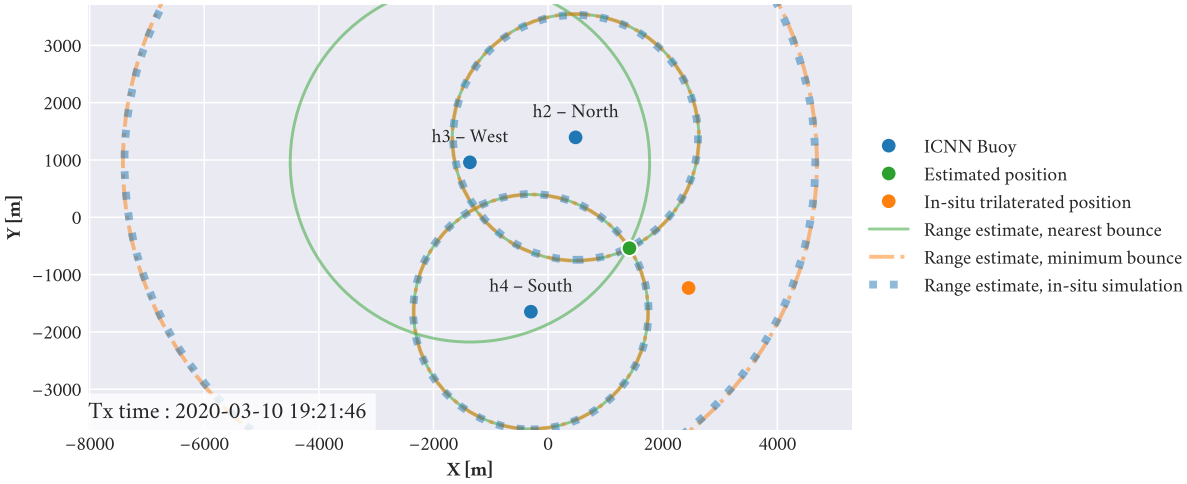


Figure 6.10: Trilateration with 3 beacons. The Minimum Bounce approach is unable to accurately estimate the effective group velocity for one of the acoustic links, leading to a large positioning error that can be easily recognized in this view. The Nearest Bounce approach is able to recognize the alternate path and adjust accordingly.

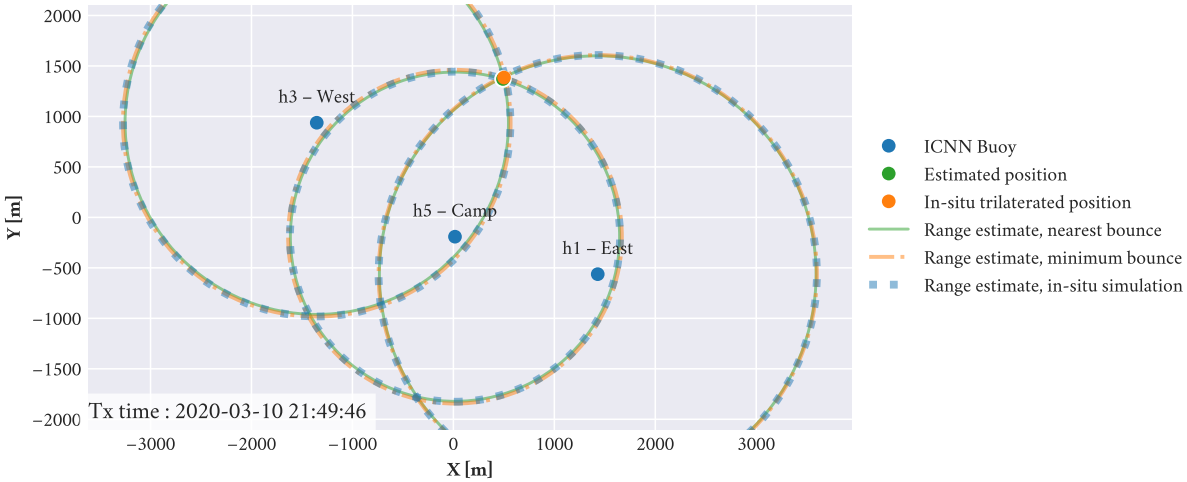


Figure 6.11: Trilateration with 3 beacons. The difference in performance of the various algorithms is not apparent in this view; however, the Nearest Bounce approach clearly outperforms its counterparts in approaching the GPS solution.

Table 6.9: Trilateration results for the example scenario shown in Figure 6.10. The limitations of the Minimum Bounce Criteria lead to errors in the order of 1 kilometer.

Method	Correction [m]	Error [m]
Nearest Bounce	14.066	10.436
Minimum Bounce	1258.776	1233.551
In-situ simulations	1243.126	1224.228

Table 6.10: Trilateration results for the example scenario shown in Figure 6.11. The Nearest Bounce Criteria outperforms the Minimum Bounce Criteria.

Method	Correction [m]	Error [m]
Nearest Bounce	6.775	6.575
Minimum Bounce	28.151	6.314
In-situ simulations	17.870	12.967

In the earlier of these two events, the advantage gained by the Nearest Bounce algorithm becomes apparent. Rather than assuming the most direct path will be the one to trigger the receiver's processing pipeline, the NBC tracks additional potential solutions up to the final range estimation step. This additional dimension of information enables the newer algorithm to correctly estimate the range from the detection reported by the West buoy (h3). Both flavors of the Minimum Bounce algorithm fail to capture an adequate approximation of the acoustic path that triggered the receiver, leading to correction and error values of more than 1200 meters.

The set of viable events recorded during ICEX-20, from which the preceding case studies originated, were in fact processed in bulk to assess the overall differences in performance between the different ranging approaches. Figure 6.12 shows the distribution of position corrections for all three methods under evaluation. Unlike with Figure 6.8, where the events were classified by transmitter

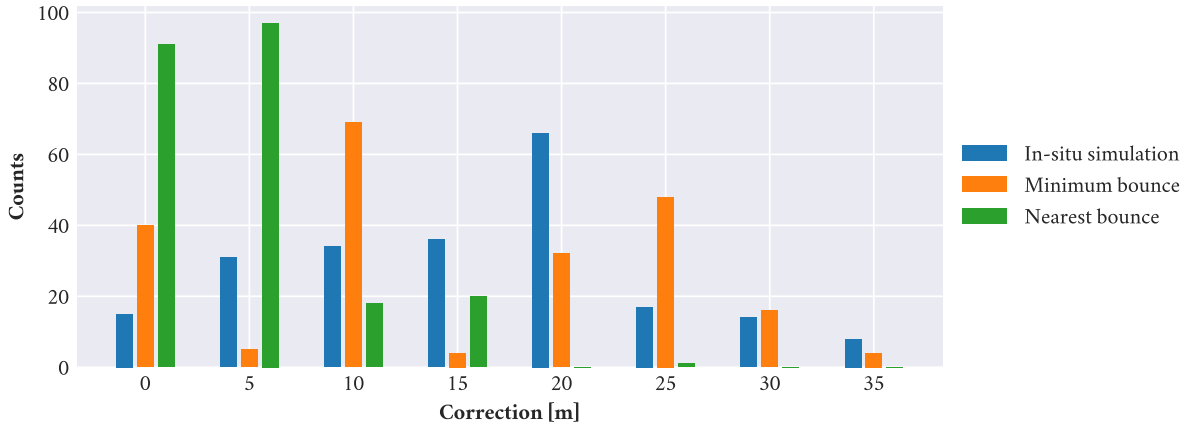


Figure 6.12: Distribution of vehicle position corrections computed in post-processing for the Minimum Bounce and Nearest Bounce algorithms. The MBC is shown as implemented during ICEX-20, with the local plane wave expansion, as well as in the simplified form used throughout this analysis.

and receiver depth, the events are now presented solely by ranging approach, for ease of comparison. Regardless of depth dependent characteristics, the NBC clearly outperforms its predecessor – the bulk of corrections go from being generally lower than about 25 meters in magnitude, to being smaller than about 10 meters.

In terms of the differences that can be observed in the distributions, particularly for the two flavors of the Minimum Bounce Criteria, it should be noted that the *in-situ* simulation results were collected directly from the mission logs, where the relevant ocean model alternated between the EOF fit and the baseline model in the EOF framework, per the timeline previously given in 6.3. Where these figures show post-processed simulation data, the results are based exclusively on the environment model obtained by the EOF fit with respect to local CTD measurements. Additionally, the two MBC sets also differ in the specific grid evaluated with the acoustic model, as well as the treatment of local solutions across the grid with respect to the plane wave expansion (or lack thereof). The case studies shown in Figures 6.10 and 6.11 correspond to events were the deployed mission, like the post-processed simulations, was configured to use the sound speed obtained by EOF fit.

6.4 AUV navigation performance

Earlier sections in this chapter centered on assessing the validity and performance of the positioning solution implemented as part of the Integrated Communications and Navigation Network (ICNN), using the static modem test as the reference data set. This section shifts focus to the data collected during vehicle operations, porting over elements of the previous analysis to report on the results attained in a more dynamic context. To set the tone, and following on earlier footsteps, Figure 6.13 first illustrates the modem-to-modem connections recorded during the entire set of operations involving AUV *Macrura*.

6.4.1 General data features

A notable feature of Figure 6.13, not so evident in its earlier counterpart, is that the new set clearly exhibits instances where the number of detections reported by a given receiver exceed the number of transmissions – meaning that the data necessarily contains instances where a given receiver detects the same transmission more than once. This can be observed for transmissions from Camp received by AUV *Macrura*, as well as transmissions from the vehicle that are later detected by its own receiver. The fact that this behavior is evident in the data is consistent with the expectation of complex multi-path in the Arctic Ocean; the lower number of successful receptions likewise aligns with the expectation that locations with a complex multi-path arrival structure may render the receiver unable to process the incoming signal. In all, the dataset contains 3260 transmissions and 12938 detection events; organized by the number of times a single beacon detects the same transmission event multiple times, the incoming events can be sorted as per Table 6.11.

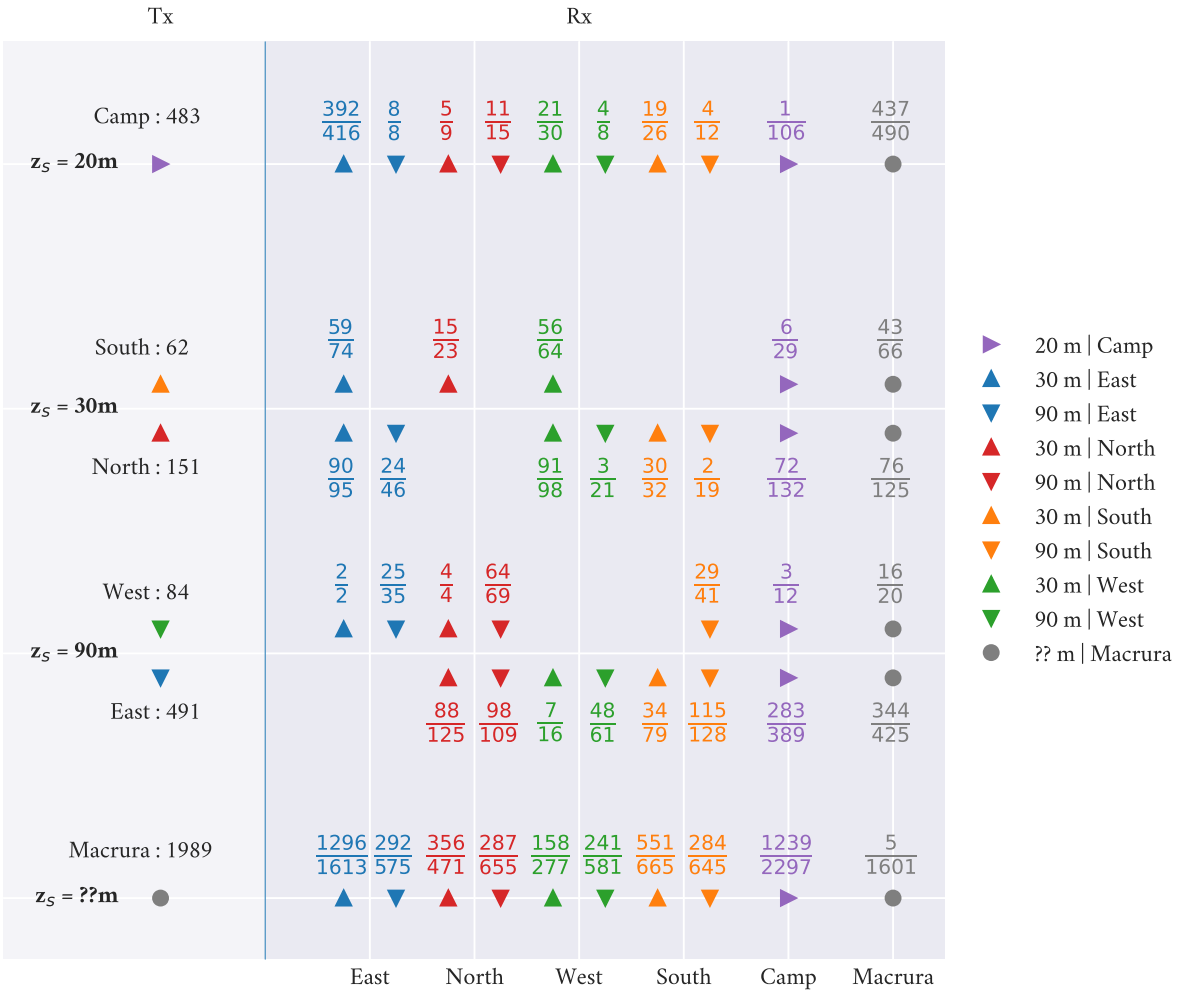


Figure 6.13: Overview of the modem connections recorded during the experiments involving vehicle operations (VLA, tethered and untethered runs). Reception event counts are shown as fractions, with the number of successfully decoded entries on top and the total number of detections on the bottom. The vehicle depth is variable, and spans 0-125 meters.

Table 6.11: Distribution of reception events, arranged by the number of times a given receiver detects the same transmission event. The instances where a beacon detects the same transmission only once are generally successful receptions; instances where the beacon reports three or more detections for the same transmission window are generally associated with an error such as incomplete data frames or headers.

No. of detections	No. of occurrences	Successful occurrences
1	7336	7310
2	1968	14
3	424	-
4	73	-
5	13	-
6	5	-
7	1	-

This phenomenon can similarly be observed in the context of data points available for trilateration. Selecting specifically for transmissions made by AUV *Macrura*, and received by the ICNN buoys, the distribution of the receptions is given by Figure 6.14. When the system is constrained to accept only successful receptions, a total of 4704 reception entries can be used for trilateration. Of the 86 instances where 5 data points are available, only 2 present an opportunity to use multiple arrivals from one of the beacons; the remaining 84 events corresponded to unique arrivals at each of the 4 ICNN buoys, plus an additional arrival at the towfish deployed from camp Seadragon to act as a fifth beacon. Loosening the constraints on the data pipeline, in order to allow receptions with PSK errors into the positioning system, would effectively mean that a total of 7779 reception records could be used for trilateration – this would increase the number of multi-path arrivals exploited for positioning, where a notable number of transmission events are associated with 6 or more data points across the 5 receivers.

Figure 6.14: Distribution of the reception events recorded across the ICNN, arranged by the number of individual detections available per individual transmission event. Shows only instances where the AUV is the transmitting node.

The additional data points exposed by relaxing the constraints of the acoustic network could introduce potential improvements in the trilateration results. It should be apparent from the case studies in the previous section that this may hold only under the assumption that some solution exists to identify the different paths that enable these arrivals. However, the ICNN was built on the assumption that the most direct path would dominate all arrivals, and that unsuccessful arrivals were necessarily unworthy of trust. In exploring the capabilities of the Nearest Bounce approach, it becomes apparent that some of the failure mode information produced by the acoustic modems could be used to identify unsuccessful but otherwise trustworthy arrivals, such as those where the failure was tied to corrupted data frames, in order to supplement the trilateration samples. The availability of appropriate modulation and data headers, for example, could be taken as indication that a particular entry could be used as part of the positioning framework.

6.4.2 Position corrections posted by the ICNN

Despite the potential gains that might come from using additional records, the ICNN was constrained to accepting only fully validated arrivals. Thus, the following performance review is like-

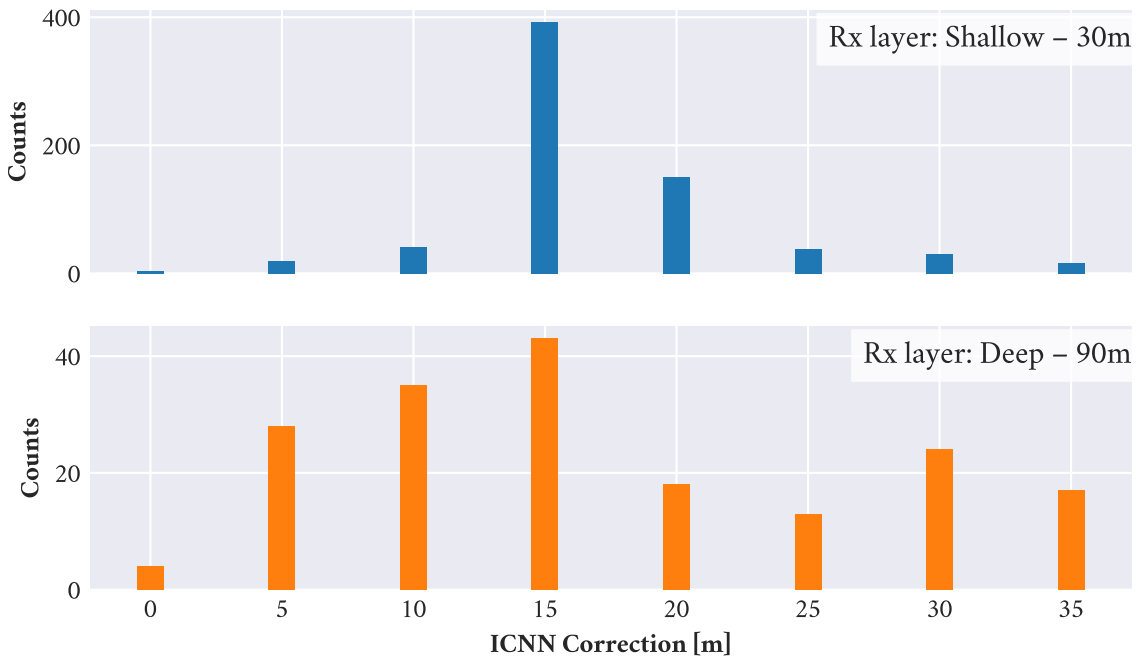


Figure 6.15: Distribution of vehicle position corrections computed by the ICNN during AUV operations. The corrected position estimates are the trilateration results computed by the ICNN, which are in turn derived from the recorded times of arrival and respective model-aided range conversions.

wise constrained to instances where the communications logs reflect only successful arrivals. Figure 6.15 shows the distribution of the corrections, by their magnitude, according to the ICNN's active reception layer. Per the data shown in Figure 6.8, the system was operated on the expectation that cross-layer links were more likely to fail than same-layer connections. Thus, the deep layer was generally only active while the vehicle was below the lens. For context, the vehicle depth exceeded 50 meters only about 20% of the time the vehicle was active.

In order to explore the performance of the two algorithms in the context of AUV operations, the communication events that could be associated with valid *in-situ* simulations were isolated from the rest of the data. Of these, the instances where the mission logs defaulted to the pre-configured conversion factor are not considered in the following analysis. However, instances where the logs show a simulation attempt with an invalid result (such as reporting a zero-valued range and travel time) were set to the pre-configured sound speed value of 1430 meters per second. This event

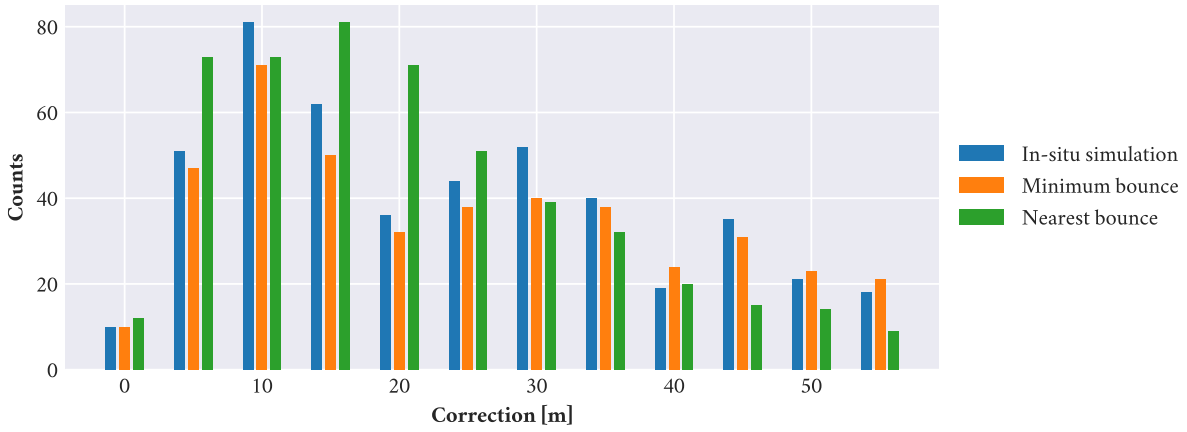


Figure 6.16: Distribution of vehicle position corrections computed in post-processing for the Minimum Bounce and Nearest Bounce algorithms. The MBC is shown as implemented during ICEX-20, with the local plane wave expansion, as well as in the simplified form used throughout this analysis.

selection resulted in Figures 6.16 and 6.17, showing the correction magnitudes and the trilateration errors for events with 3 or more data points available throughout the duration of AUV operations.

As was noted earlier, the insight garnered from the magnitudes of the position corrections is quite valuable in the modem test scenario, because the trilateration results can be compared with GPS logs to effectively illustrate positioning accuracy. In contrast this same insight is somewhat less valuable in the absence of a reliable ground truth reference, as is the case in AUV operations, because the correction magnitudes necessarily depend on the vehicle's internal navigation estimate – one that is prone to larger errors due to sensor drift, ocean current and other sources of error not captured in the navigation model. Under these conditions, larger corrections aren't necessarily indicative of worse performance.

Consider the distribution of corrections shown in Figure 6.16 for all algorithms. The *in-situ* simulations exhibit a peak around 10 meters, and another around 30 meters; a similar pattern appears for the post-processed MBC estimates. The Nearest Bounce approach, on the other hand, exhibits a single, wider peak spreading between 5 and 20 meters; its highest point is located at the 15 meter mark.

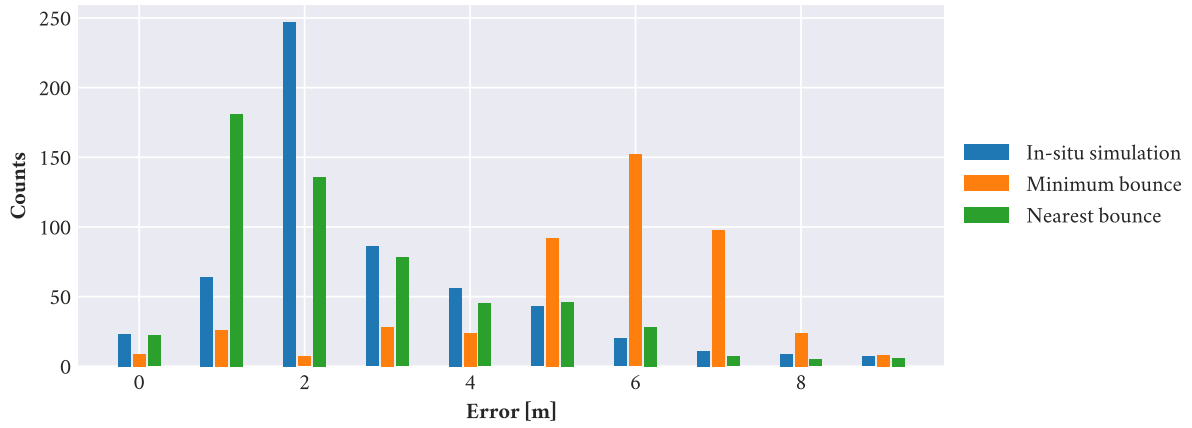


Figure 6.17: Distribution of vehicle position corrections computed in post-processing for the Minimum Bounce and Nearest Bounce algorithms. The MBC is shown as implemented during ICEX-20, with the local plane wave expansion, as well as in the simplified form used throughout this analysis.

When exploring the trilateration errors in Figure 6.17, it becomes apparent that the worst of the three methods is the post-processed MBC, with its error distribution for the selected cases peaking at around 6 meters. The difference in trilateration error between the remaining methods is less obvious. As a reminder, the remaining methods correspond to the plane wave expansion used during real-time operations for ICEX-20, and the simplified application (no plane wave expansion) of the NBC. The one difference that stands out between these two methods is that over 100 events shift from having about 2 meters in the reported trilateration error, to having a value closer to the 1 meter mark.

6.5 Summary

This chapter has presented two algorithms for model-aided range estimation, aimed at supporting the positioning stage of the Integrated Communications and Navigation Network. Both methods are validated against a GPS-tracked static experiment conducted during ICEX-20. The results illustrate the potential gains offered by the second approach, proposed as an improvement to the

algorithm fielded during ICEX-20. The chapter also extended the performance analysis beyond point-to-point ranging, to assess the impact on positioning by trilateration in the same static experiment used for the initial validation. Finally, the chapter presents the results of extending the trilateration analysis to vehicle operations conducted during ICEX-20, which lack the benefit of a reliable ground truth positioning estimate.

As of this writing, the validation work and improvements of the model-aided range estimation framework (based on the static modem test data; Sections 6.1 and 6.2) have been submitted for publication in "Under-ice acoustic navigation using real-time model-aided range estimation". A related discussion of these sections is also presented by Bhatt ("A Virtual Ocean framework for environmentally adaptive, embedded acoustic navigation on autonomous underwater vehicles"), coauthor of the submitted article. The *in-situ* performance of the ICNN trilateration solution during the modem test experiment (see Fig. 6.8) has also been submitted for publication, in "A high-resolution AUV navigation framework with integrated communication and tracking for under-ice deployments".

7 Machine learning, revisited

Strange about learning; the farther I go the more I see that I never knew even existed. A short while ago I foolishly thought I could learn everything—all the knowledge in the world. Now I hope only to be able to know of its existence, and to understand one grain of it.

Is there time?

– Daniel Keyes, *Flowers for Algernon*

The domain of machine learning (ML) is vast and constantly evolving, as it is increasingly taken up by various fields of study and application. Climatology and weather forecasting, speech recognition and image recognition – applications that impact lives on a daily basis in modern times – all lean heavily on the advances of machine learning techniques to produce the results expected of them. As was mentioned in Chapter 3, the means to achieve said results have evolved in three primary waves of development over the past century, starting under the name of *cybernetics* in the 1940s-1960s; later came the *connectionism* period around the 1980s-1990s; and finally reaching the field of ML as we know it today. With each wave, the field shifted both in the complexity of its tools and the level of abstraction it could work with.

Following a similar progression of increasing complexity and abstraction, this chapter aims to explore how the vehicle's capabilities for environmental adaptation may be enhanced by assimilating additional information from the vehicle's sensor data. The model-aided approach presented in Chapter 6 used direct environmental measurements to estimate the sound speed profile that was

then fed to the acoustic propagation model; the latter then supported the ranging problem by providing estimates of the effective group velocity. As with the behavior classification work in Chapter 3, this chapter seeks to augment the information provided to the learning framework, this time ingesting both the direct environmental measurements (namely, the CTD data) and the indirect measurements acquired by way of the acoustic data – such as is done in tomography – to inform the estimation of the sound speed profile. This augmented estimate would then be used to support the travel time to range computation performed by the ICNN.

To tackle this augmented data assimilation, the chapter first explores additional related works beyond the discussion of Chapter 2, specifically in the context of machine learning applications for environmental sciences and ocean acoustics. Advances in the subset of Deep Learning solutions that has come to be known as Physically Informed Neural Networks (PINNs) are also discussed, and their applicability to the task of the ICNN is explored using a benchmark environmental model. Finally, a pseudo-tomographic approach is demonstrated as a viable candidate to produce additional environment estimates beyond those produced by a combinatorial exploration of the EOF weight estimation problem.

7.1 ML in environmental and ocean sciences

The application fields within the scope of environmental sciences – such as weather forecasting, hurricane warning and other disaster management systems – have been powered by advances in computational techniques and the corresponding technology as early as the 1950's. For example, the Joint Numerical Weather Prediction Unit (JNWPU) – a collaboration between the U.S. Weather Bureau, Navy, and Air Force – opened its doors on 1 July 1954 with the objective of using forecasting methods developed through research, and implemented through numerical techniques, to produce updated weather charts on a regular and ongoing basis. Since those early days, the models have been continuously improved to consider new data points (such as adding observations made by satellite

remote sensing) and to expand the spatial resolution of the models, even within the constraints of limited observations.[1] Nowadays, the numerical weather prediction models are run by the National Centers for Environmental Prediction (NCEP).

7.1.1 The Landslide Hazard Assessment for Situational Awareness

The applications of numerical models extend well beyond weather forecasting. A recent paper presented advances to NASA's Landslide Hazard Assessment for Situational Awareness (LHASA) models, where machine learning techniques are used to provide improved landslide nowcasting at the global scale. LHASA version 2, the machine learning approach presented in the paper, is reported as being twice as accurate as its predecessor, LHASA version 1. Released in 2018, version 1 was not a machine learning model; instead, it combined satellite precipitation data with a global landslide susceptibility map to produce its nowcasts.[2] The updated version of the landslide warning system considers the same signals as its predecesor, but also evaluates additional variables with the potential to explain some of the variability in landslide occurrence.

Described in simple terms, the LHASA v2 model consists of a collection of 300 shallow decision trees built using the ML framework XGBoost. Each tree in the model has a depth of 2 layers, where each layer compares the value of one variable against a threshold; given the limited number of layers, at most two of the explanatory variables can interact in any given tree. A simple example of one such tree may determine whether the measured rainfall exceeded a certain value (1); depending on that result, the system then checks whether the rainfall exceeds yet another threshold (2.1), or whether the distance to a fault is greater than a third cutoff value (2.2). In all, each tree can produce a total of 4 different outcomes; when taking all 300 outcomes as an ensemble, the LHASA v2 system produces a probabilistic value representing the landslide hazard at each point in the nowcast grid.

[1] Harper, Uccellini, Kalnay, Carey, and Morone, "50th Anniversary of Operational Numerical Weather Prediction".
[2] Stanley, Kirschbaum, Benz, Emberson, Amatya, Medwedeff, and Clark, "Data-Driven Landslide Nowcasting at the Global Scale".

Among the chief concerns discussed in the LHASA paper, and throughout the ML literature, is the importance of data quality. Indeed, a significant portion of the Landslide Hazard Assessment paper is spent addressing limitations of some of the data sets used to inform the values of the explanatory variables evaluated in the paper. Portions of the world's landslide inventories could not be used reliably as part of the training set for the new model, for example, due to limited spacial precision. Other entries corresponded to events outside of the temporal window considered in their analysis. Certainly, biases in machine learning applications have been the subject of numerous studies, where large data sets are often blindly trusted and used extensively in derivative research despite the fact that they sometimes contain errors that may lead to undesirable consequences.[3] For a system that is intended to provide an early warning to communities around the globe, as is the case with LHASA, recognizing the potential limitations and sources of bias contained within its reference data during the early stages of model development can have a significant impact on model performance.

7.1.2 Learning data relations in Ocean Acoustics

Within ocean acoustics, as in many other fields, the advances in machine learning are generally regarded as a promising solution to challenges such as data sparsity, missing or corrupted data and generally large datasets with unknown patters captured therein. The data-driven nature of machine learning, and its ability to automatically detect and exploit patterns within the data, have thus been aimed at both environmental and operational challenges.

Within the environmental scope, different applications of dictionary learning have been proposed as an alternative to the use of Empirical Orthogonal Functions (EOFs) in the pursuit of effective yet sparse representations of model data, such as the sound speed profile. Where EOFs are generally orthogonal, sometimes introducing only a limited smoothing transform of the truly orthogonal shapes to produce the final basis functions, applications that rely on them generally benefit only from the

[3] Birhane and Prabhu, "Large image datasets: A pyrrhic win for computer vision?"; Northcutt, Athalye, and Mueller, *Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks*.

first few shapes in the set as they capture the majority of the explained variance. In dictionary learning, the resulting set is not bound to orthogonality, but rather can be an over-complete set. That is, multiple shapes in the dictionary can share features associated with the explained variance. One of the advantages reported for dictionary learning is the ability to attain comparable (and sometimes better) performance as the EOF form, with fewer non-zero weights. That is, the dictionary approach enables increased sparsity in the final representation of the measurements. The cost associated with this enhanced sparsity comes in the form of changes to the weight estimation process; as the dictionary learning method centers on sparsity and uses non-orthogonal functions, it generally requires a somewhat more complex iterative method to weight estimation, compared to the straight-forward least-squares approach used for EOF weights. Dictionary learning techniques have been applied to sound speed profile estimation as well as to travel time tomography problems.[4]

In addition to environment-centric applications, machine learning has also been applied to seabed characterization and range estimation – tasks that generally shift the focus to a more operations-centric perspective. As has been stated previously, data quality remains paramount to the successful implementation of ML; and since extensive, appropriately labeled data sets are not always available from the start for this kind of work, projects in this vein often use simulated data. Rather than relying on limited field data sets for the training process, these real measurement sets – if they exist at all – are instead used to inform model parameters. The source spectrum, acoustic propagation models, and expected signal to noise ratios, to name some of the parameters of interest, are all considered when producing the simulated training set. Using this type of synthetic training set, which is guaranteed to have correct labels by design, range estimation via Convolutional Neural Networks (CNNs) has been found capable of outperforming the conventional matched field processing (MFP),

[4]Bianco and Gerstoft, "Compressive acoustic sound speed profile estimation"; Bianco and Gerstoft, "Dictionary learning of sound speed profiles"; Bianco and Gerstoft, "Travel Time Tomography With Adaptive Dictionaries".

and to exhibit enhanced robustness to mismatch between the environmental model and the true environment.[5]

7.2 Physically Informed Neural Networks

The field of machine learning is constantly evolving, with contributions to the core techniques and algorithms coming in from a wide range of application fields. One of the recent advances that is of especial interest to complex physical systems is the presentation of Physically Informed Neural Networks, or PINNs. At their core, PINNs aim to solve forward and inverse problems described by nonlinear partial differential equations.[6] In order to better grasp the potential value of PINNs, this section first discusses the types of machine learning addressed in the literature, followed by the basic Neural Network model that underpins the architecture of PINNs. The differences between PINNs and other neural networks is discussed next. Applications of PINNs are briefly introduced thereafter, as a motivation for their value in the scope of this thesis.

7.2.1 A brief discussion on types of learning

There are numerous learning methods and different types of problems that can be handled with machine learning. In order to make the design decisions needed to implement ML in new applications, it is necessary to understand these differences, as they can dictate the reliability and complexity of the resulting system. Sorting by problem type, for example, machine learning can perform the following tasks:

[5] Van Komen, Neilsen, Howarth, Knobles, and Dahl, "Seabed and range estimation of impulsive time series using a convolutional neural network"; Chen and Schmidt, "Robustness Analysis of a Convolutional Neural Network Approach to Source-Range Estimation in a Simulated Arctic Environment".

[6] Raissi, Perdikaris, and Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations"; Haghighat and Juanes, "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks".

- **Binary classification**

  The behavior classification exercise developed in Section 3.2 is an example of binary classification, where the system is trying to decide between two possible outcomes.

- **Multi-class classification**

  The environment characterization exercise developed in Section 3.3 is an example of multi-class classification, where the algorithm is trying to decide between three possible outcomes.

- **Regression**

  NASA's Landslide Hazard Awareness system, briefly presented in Section 7.1.1 is an example of a regression problem, where the learning framework is geared towards producing a probability value rather than a class label.

In addition to the above, machine learning can also perform tasks such as transcription, machine translation, anomaly detection, denoising, and more.[7] When considering the type of learning methodology used, the three most commonly used are:

- **Supervised learning**

  The core concept behind supervised learning is to learn the mapping from a set of inputs to a corresponding set of outputs. The solutions – for example, the true classification labels – are made available to the system during the training process. This should be apparent in the behavior classification exercise from Section 3.2, where the training is based on simulation signals clearly labeled by motion pattern. In the LHASA training process, the probability values produced by the ensemble of decision trees are converted to a binary classification based on some threshold (the paper uses 12%[8]), and the system's ability to match the presence or absence of landslide reports in the reference data sets is used as a metric of its performance.

[7] Goodfellow, Bengio, and Courville, *Deep Learning*.
[8] Stanley, Kirschbaum, Benz, Emberson, Amatya, Medwedeff, and Clark, "Data-Driven Landslide Nowcasting at the Global Scale".

- **Unsupervised learning**

  In contrast to the previous approach, unsupervised learning instead aims to extract features from the data without aspiring to match predetermined solutions or labels. Examples of unsupervised learning include data grouping methods such as K-means, and latent model extraction like Principal Component Analysis (PCA) and dictionary learning. Recall that PCA is also the basis for generating Empirical Orthogonal Functions (EOFs).

- **Reinforcement learning**

  The third basic type in this list, reinforcement learning, aims to update the learned model based on feedback produced by its environment. Although not given expected solutions as with supervised learning, this type of learner is not entirely free to explore the data either, as would be the case with unsupervised learning. Instead, the learner must approach some central goal by choosing from a set of actions to take at each iteration. The feedback from the environment reflects whether the action taken is helping the learner achieve its objective – positive feedback is given as a numerical reward. The learning process, then, consists of identifying a model wherein the system can consistently make helpful decisions at each iteration, in order to maximize its reward.

In addition to these basic methods, hybrid approaches can be used as well. Semi-supervised learning, for example, is a combination of the first two approaches, where only a subset of the data is properly labeled but the learner seeks to exploit information contained in the unlabeled entries as well.

7.2.2 Conventional neural network architecture

The basic layout of neural networks can be described as a stack of layers, each with some number of perceptrons (neurons). Each individual perceptron accepts the output of the previous layer as its

input x, adjusting each signal by some weight w; the perceptron also adds a bias term w_0, such that the linear transform a_q is given by Eq. 7.1. The output of the q-th neuron in the current layer is then given by $z_q = g(a_q)$, where $g(\bullet)$ is a non-linear activation function applied to the linear transform.

$$a_q = \sum_{n=1}^{N} w_n x_n + w_0 \tag{7.1}$$

The training process is then used to determine the weights w needed at each of the inter-neuron connections, in order to successfully map the input samples to the expected outputs. When all weights connecting two layers are non-zero, the layers are described as being fully connected. Zero-valued weights can be used to disconnect a neuron from one of its inputs. To be considered Deep Learning, a machine learning framework using neural networks must satisfy three basic conditions: (1) the features are learned through the training process rather than handcrafted; (2) the features extracted by each of the layers are organized from low-level to high-level abstraction; and (3) there are at least two layers of non-linear transformations.[9]

7.2.3 How PINNs compare to other NNs

The fundamental distinction between Physically Informed Neural Networks and conventional Neural Network applications is in how the learning process is supervised. The conventional approach takes a set of known, labeled data points and conditions the network to minimize the error between the values predicted by the network and the real labels provided in the set. Physically Informed Neural Networks, on the other hand, are not necessarily given data with known labels or values as the output target. Instead, the output of the neural network is transformed per the partial differential equations (PDEs) that describe the problem, and the solution space is restricted to the subset of solutions where the underlying PDEs are valid (Fig. 7.1).

[9] Bianco, Gerstoft, Traer, Ozanich, Roch, Gannot, and Deledalle, "Machine learning in acoustics: Theory and applications".

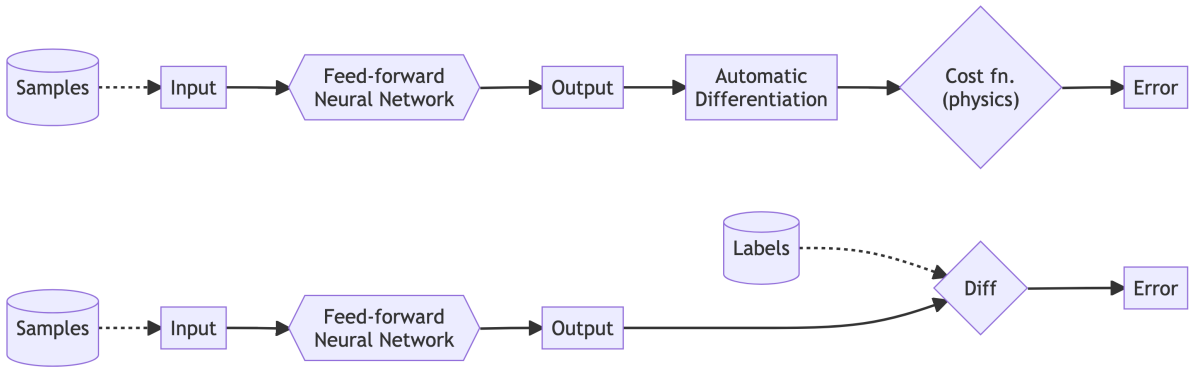


Figure 7.1: Diagrams illustrating the key differences between PINNs used for data-driven solution of PDEs (top), versus the conventional label-matching model for artificial neural networks (bottom).

Because PINNs are in effect supervised by a set of PDEs, the numerical solution to the problem need not be known in advance. Instead, the network can be arranged to learn the solution field based on the physics constraints described by the PDEs; any parameters captured in the functions would have to be provided, as the PINNs act as numerical rather than symbolic function approximators. Though solving the problem deterministically may be more accurate for one-off calculations, the PINN approach affords the potential to quickly compute predictions anywhere in the input space, after the network has been properly configured and trained. Alternatively, PINNs can also be arranged to ingest measurement data as the expected output, enabling the network to discover the parameter values in the underlying set of equations.[10] These two problem types are also referred to as (1) data-driven solution and (2) data-driven discovery of partial differential equations.

This novel approach to artificial neural networks has been used in demonstrations for various fields. In material modeling, for example, PINNs have been used to capture the potential energy surface of inter-molecular systems; other benchmark demonstrations include learning the solution to the one-dimensional, non-linear Schrodinger equation; or modeling incompressible fluid flow

[10] Haghighat and Juanes, "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks".

as described by the Navier-Stokes equation.[11] Of particular interest in the scope of this thesis is the use of PINNs to solve the Eikonal equation, as well as in tomography applications.[12] It may be noted that other Deep Learning techniques, such as the use of feed-forward networks with residual blocks, have also been applied to the resolution of the Eikonal equation.[13]

7.2.4 Solving the Factored Eikonal equation with PINNs

$$|\nabla T|^2 = \frac{1}{c^2(\mathbf{x})} \tag{7.2}$$

The Eikonal equation (Eq. 7.2) describes the relation between the local gradient (∇) of the time field T and the corresponding value of the sound speed $c(\mathbf{x})$ at any position $\mathbf{x}$ in the environment. However, the equation is non-linear and frequently has multiple solutions. Furthermore, the Eikonal equation is subject to a singularity, due to the presence of a point-source. This section summarizes prior work on the use of PINNs to solve an alternate form of the time field, as given by the factored Eikonal equation.[14]

Where the standard form of the equation is subject to the point-source singularity, the factored Eikonal equation seeks to separate the time field into two components. The first is a known term T_0, described by the euclidean distance from the source to any point in the environment and scaled by a known reference sound velocity $v(\mathbf{x}_s)$ – often of unit value, or equivalent to the value at the source. The second component is an unknown term τ that captures the distortion effect of the environmental model (sound speed) on the known time field T_0. In this form, the true time field $T(\mathbf{x})$ is given by Eq. 7.3.

[11] Pun, Batra, Ramprasad, and Mishin, "Physically informed artificial neural networks for atomistic modeling of materials"; Raissi, Perdikaris, and Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations".
[12] Waheed, Haghighat, Alkhalifah, Song, and Hao, "Eikonal solution using physics-informed neural networks"; Waheed, Alkhalifah, Haghighat, Song, and Virieux, *PINNtomo: Seismic tomography using physics-informed neural networks*.
[13] Smith, Azizzadenesheli, and Ross, *EikoNet: Solving the Eikonal equation with Deep Neural Networks*.
[14] Waheed, Haghighat, Alkhalifah, Song, and Hao, "Eikonal solution using physics-informed neural networks".

$$T(\mathbf{x}) = T_0(\mathbf{x})\tau(\mathbf{x}) \,, \qquad T_0(\mathbf{x}) = \frac{|\mathbf{x} - \mathbf{x_s}|}{v(\mathbf{x_s})} \tag{7.3}$$

$$T_0^2|\nabla\tau|^2 + \tau^2|\nabla T_0|^2 + 2T_0\tau(\nabla T_0 \cdot \nabla\tau) = \frac{1}{v^2(\mathbf{x})} \tag{7.4}$$

The original Eikonal equation can be expressed in terms of the factored time field, which yields the form shown in Eq. 7.4. The singularity introduced by the point-source is captured by the known term T_0, making the unknown component τ a smooth function in the vicinity of the source. Under the PINN framework, then, a neural network is used to convert a set of coordinate values into predictions of the time distortion field, $\hat{\tau}$. Like conventional neural networks, the various steps in the PINN benefit from the implementation of automatic differentiation (AD), which makes it possible to transform the output of the neural network such that the training target is to minimize the cost function $\mathcal{L}_{\text{eik}}$ in Eq. 7.5.

$$\mathcal{L}_{\text{eik}} = T_0^2|\nabla\hat{\tau}|^2 + \hat{\tau}^2|\nabla T_0|^2 + 2T_0\hat{\tau}(\nabla T_0 \cdot \nabla\hat{\tau}) - \frac{1}{v^2(\mathbf{x})} \tag{7.5}$$

Per Eq. 7.3 and in taking advantage of the implementation of AD, the known field T_0 can be expressed in the PINN architecture by its differential relation to the inputs, which consist of the source and receiver coordinates. The depth-dependent sound speed function is used in T_0 with respect to the source depth and in $\mathcal{L}$ with respect to the receiver depth; assuming the profile is known, these values can likewise be computed from the coordinate inputs. This leaves only the distortion field τ to be learned by training a feed-forward neural network, as was previously illustrated in Figure 7.1.

7.3 Augmenting the model-aided environment estimation framework

The objective of reduced-order representations, like those facilitated by EOFs and dictionary learning solutions, is to condense potentially very large data sets into a compact form that is suitable for use under constrained conditions. One such scenario, introduced in earlier chapters of this thesis, is that of sharing environmental model estimates via acoustic communications in a collaborative operational paradigm like the one employed during ICEX-20. To illustrate how ML is expected to impact the vehicle's environmental adaptation capabilities, this section first addresses the basic implementation of an EOF approach. With the basic concept in place, the text then explores how machine learning may be employed to augment the environment estimation framework by assimilating information from additional data sources available to an AUV.

7.3.1 The baseline EOF solution

The use of Empirical Orthogonal Functions (EOFs) as a representation basis starts with the discovery of primary variation modes in a data set. Like with Principal Component Analysis (PCA), the objective is to build the set of basis functions A directly from the data; the main difference is that EOFs may add a local smoothing step in some applications, which is not used in conventional PCA. In the context of range-independent SSP representation, the data set and resulting basis functions are bound to a common depth vector, which must be dense enough to capture the target regions of variability; only the first few basis functions are generally considered useful in capturing the majority of said variability. Accordingly, a new set of measurements must be discretized into the same depth bins, and have the baseline profile subtracted, to produce a new sample vector c_m with a matching same number of entries. The weights fitting the new sample in terms of the basis functions is then given by Eq. 7.6.

$$\mathbf{w} = \arg\min_{\mathbf{w}} \mathcal{L}\,, \quad \mathcal{L} = \|c_m - A\mathbf{w}\| \tag{7.6}$$

In the scope of ICEX-20 operations, where the weights were computed against CTD casts conducted from camp by human operators, this first form was considered sufficient. The human-operated casts consisted of deploying a tethered RBR Concerto to a depth of approximately 300 meters, which was done a total of 4 times; and deploying expendable CTDs (XCTDs) to a depth of about 1000 meters, which was done a total of 5 times. In all, 9 CTD casts were collected from camp. The relevance of the cast depths stems from the need for sufficient samples in the discretized form, to ensure a stable fit; ahead of the experiment, this depth relation was selected to align with the depths used in the HYCOM model.

With the CTD data processed and discretized, the basis functions were trimmed to the max depth captured by the cast and the weights were computed for all possible combinations of the leading EOFs – using the first 7 basis functions, this meant 127 scenarios were assembled. The resulting fits were then ranked by the regression error, and relayed to a Tactical Decision Aid where a human operator could assess the quality of the top-ranked entries in terms of realism and smoothness. For each weight combination reviewed, the TDA also presented the results of a bespoke acoustic propagation model, so the operator could account for the expected acoustic performance in their selection of the weights reported to the autonomy infrastructure. For ICEX-20, the TDA operator was a submarine officer from the U.S. Navy.[15]

Where the CTD casts conducted from topside went deep enough into the water column to ensure the regression was well constrained, the translation of the TDA system onto the vehicle autonomy stack faced two main challenges. First, the relatively shallow maximum depth of AUV *Macrura* meant that even when starting a mission with a max depth dive, the discretized samples would be

[15] Bhatt, Howard, and Schmidt, "Embedded Tactical Decision Aid Framework for Environmentally Adaptive Autonomous Underwater Vehicle Communication and Navigation".

further constrained. Second, the fact that CTD samples collected by the vehicle were not in fact initialized with a max-depth dive meant that the vehicle might reach a point in the mission where it has collected enough samples in different depth bins to make the weight computation possible but not necessarily stable in a physical sense. That is, the vehicle could produce weights that may lead to unrealistic sound speed profiles, since the unsampled depth bins are left unbound by the trimming step.

$$\mathcal{L} = \underbrace{\|c_m - A\mathbf{w}\|}_{\text{fit CTD data}} + \underbrace{\|\lambda I\mathbf{w}\|}_{\text{regularizer}} \tag{7.7}$$

This issue can easily be addressed by introducing a regularization term to the loss function minimized in Eq. 7.6, leading to the loss function in Eq. 7.7. The regularization term, scaled by the λ coefficient, produces an increasingly large penalty as the weights diverge from the default vector of zeros; in practice, this can be regarded as constraining all sound speed values – including those at unsampled depths – so that they remain close to the baseline profile, which is itself physically significant since it is obtained from field data when developing the EOF basis.

7.3.2 Accounting for acoustic time-of-travel data

As was done in the example applications from Chapter 3, this section explores how machine learning techniques may enable AUV operators – human and autonomous alike – to assimilate data from multiple channels, in order to expand the amount of knowledge about the environment that is available for the decision-making process. Expressed in terms of the loss function, the goal is to fit data from an environmental sensor in the form of a CTD, as discussed in the previous section; and also fit data from the acoustic communications component of the Integrated Communications and Navigation Network. Conveying that the relation between the EOF representation and the timing data

is not necessarily known, Eq. 7.8 simply states this component of the loss in terms of a function $f(\bullet)$ that depends on the EOF weights $\mathbf{w}$ and the travel time measurements τ.

$$\mathcal{L} = \underbrace{\|c_m - A\mathbf{w}\|}_{\text{fit CTD data}} + \underbrace{\|\lambda I \mathbf{w}\|}_{\text{regularizer}} + \underbrace{\| \overbrace{f(\mathbf{w}, \tau)}^{?} \|}_{\text{fit timing data}} \tag{7.8}$$

Of course, the assumption that $f(\mathbf{w}, \tau)$ is not known can be challenged as an incomplete statement. It may be more accurate to say that the relation between the two components is a complex one. After all, this text has already presented the Eikonal equation and its use in ray tracing: given a set of weights that represent a sound speed profile, the forward problem of the acoustic propagation model can be solved for that SSP to predict the travel times that might be detected at some set of receiver coordinates. These predictions would then be compared with the measurements in the time-based component of the loss function. The Eikonal equation, in its most familiar Cartesian form has known limitations – specifically with regards to multi-path arrivals. These were discussed in Section 2.2, which also presented the more involved ray coordinates and the method of characteristics, used in standard software tools for the field of ocean acoustics to address the lack of unique solutions in the Cartesian representation.

Under the guise of solving the inverse problem, a brute force search across some reduced-order representation of the environment (which could be built using EOF or learned dictionaries) might yield adequate results by enforcing the forward propagation model as given by the Eikonal equation. Indeed, similar approaches are used in beamforming and matched field processing (MFP), where target bearing or position are identified by comparing the sensor measurements with simulated replicas. But, such an approach comes with significant computational costs and well-known numerical limitations.

Consider, for example, the weight discretization illustrated in Section 4.3. As part of the simulation work conducted in preparation for ICEX-20, the weight of a single basis function was evaluated

across 15 values, and later 2 EOFs were likewise evaluated on a 15×15 grid. In a brute force approach, the system is forced to run calculations for the entire search space, regardless of its utility; and as the number of basis functions increases, so does the computational cost of searching for a solution.

While the impact of using more basis functions could be countered by sampling each domain more sparsely, enforcing a specific grid that fails to represent the variability of the loss function in the search space can also misrepresent the solution space and lead to erroneous estimates. This is especially important in light of the fact that the landscape of the loss function is sensitive to mismatch between the simulated environment and the real one. The tradeoff between resolution and robustness, related to parameter sampling density and model mismatch, has been well documented for applications of the conventional and optimal beamformers;[16] the underlying concepts apply to this discussion as well.

The aforementioned considerations have led to the use of machine learning solutions as potential alternatives to techniques like MFP, given that data-driven Convolutional Neural Networks (CNNs) have been shown to be more robust to model mismatch.[17] In a similar way, treating $f(\bullet)$ as an unknown in the environmental adaptation problem reflects the complexity of the inversion, and is intended to illustrate where machine learning techniques may yield the most benefit, as a replacement for a brute force approach.

7.4 Exploring the potential of PINNs for field use

The ability of Physically Informed Neural Networks to learn a solution field from the partial differential equations that describe the problem make them an attractive candidate for applications such as the pursuit for environmental adaptation discussed in this chapter, where direct measurements

[16] Van Trees, *Optimum Array Processing*.
[17] Chen and Schmidt, "Robustness Analysis of a Convolutional Neural Network Approach to Source-Range Estimation in a Simulated Arctic Environment".

may be exceedingly limited and sparsely distributed across the field. In a hybrid approach, PINNs can be tasked with (1) fitting the few measurements available, while (2) ensuring the validity of the physical principles anywhere within the target domain. The PINN training process would become a proxy for the inverse problem by learning the underlying model.

As with other neural networks, the model learned by a PINN would be a data-driven approximation of the true field. The network's capacity ought to be calibrated to align with the variability in the data, lest the network be prone to either underfitting or overfitting the data, and thus fail to learn a suitable generalization. Numerical artifacts can also impact the learning process – while the physical relation between meters and kilometers is well defined, alternating scales may lead to entirely different models learned by the neural network as each layer enforces a linear transform and a non-linear activation function.

7.4.1 PINN architecture

To better illustrate the impact of these sensitivities in the context of the problem at hand, this section presents three different attempts at recreating the travel time field by learning the time field τ for the factored Eikonal equation. Note that the limitations of the Eikonal equation in Cartesian space have already been discussed in the text; the choice to evaluate the equation in this form here will be addressed later in this section. For this exercise, the network used to learn the time distortion field is configured to have 20 hidden layers and 20 neurons per layer (Fig. 7.2). The grid used in these example scenarios spans 1000 points in range, and 500 points in depth. With the objective of enforcing the physical constraints in the entire domain at each step in the training process, all points are used for each batch. Lastly, the networks are trained for 5000 epochs, assuming a source depth of 90 meters.

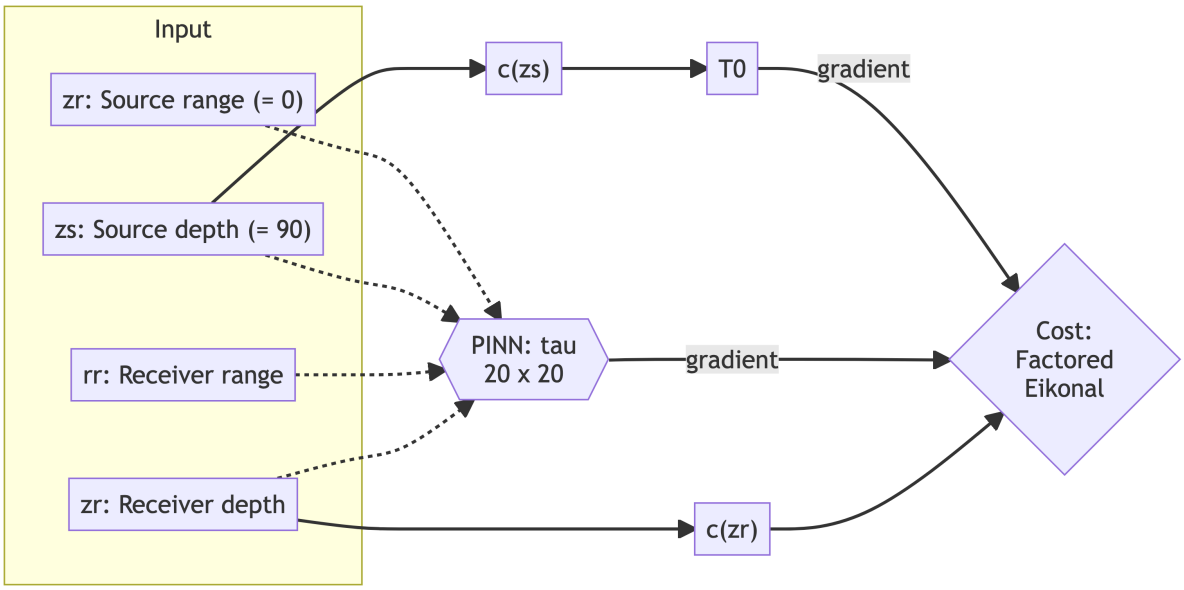


Figure 7.2: Diagram illustrating the PINN framework to learn the factored Eikonal equation.

7.4.2 A PINN case study in 3 parts

Figure 7.3 presents the first of these three scenarios, using a sound speed based on the Munk analytical shape, with the profile's minimum sound speed at a depth of 100 meters. It can be observed in the figure that this PINN can learn to reproduce the downward refracting effect of the upper water column. Though the upward refracting effect of the lower portion of the profile is less pronounced, the change of slope in the stream lines (which are used as proxies for the ray paths) suggest the network is likewise capturing the underlying physical constraints. The BELLHOP ray tracing solution for this environment, with a minimum at 100 meters, is shown in Figure 7.4.

While the first example appears promising, one of its limitations is that it does not capture paths extending to some nominal bathymetry; Chapter 6 discussed how bottom-bounce paths appeared in the ICEX-20 data, and may appear in similar data sets for other regions. Though the Eikonal equation alone doesn't capture multipath arrivals or boundary interactions as such, a composite model (much like the method of images) could be used as a reasonable approximation. The data collected in the Beaufort Sea also conveys the need for extending the model to longer ranges; the

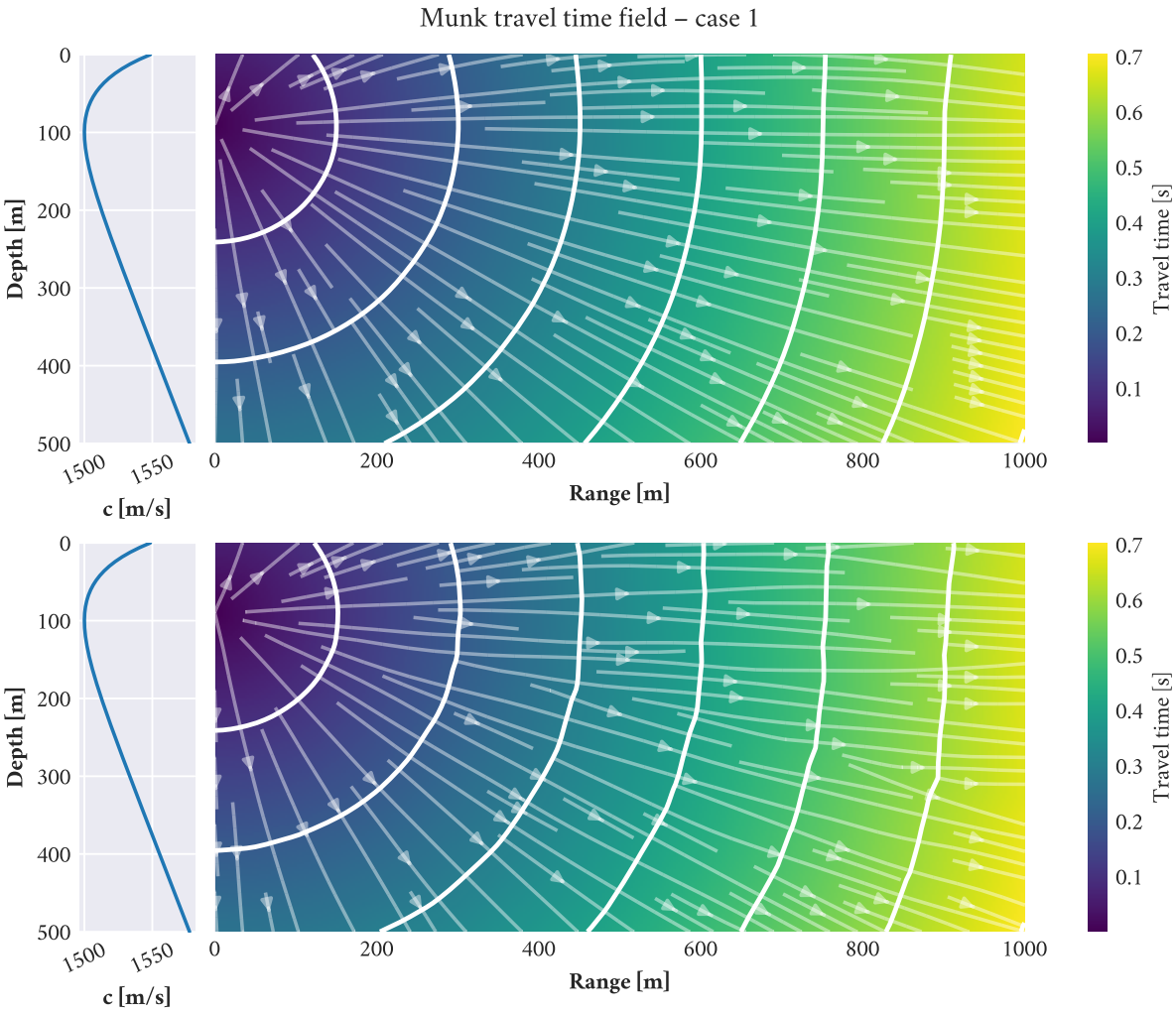


Figure 7.3: Learning the travel time field for a Munk-shaped environment (case 1). The sampled field is limited to 1 km in range and 500 m in depth; the sound speed minimum is located at 100 m depth. The top and bottom plots show two distinct repetitions of the training process; some variability appears across repetitions, as expected. In general terms, though, the learning process and results are fairly repeatable for this example.

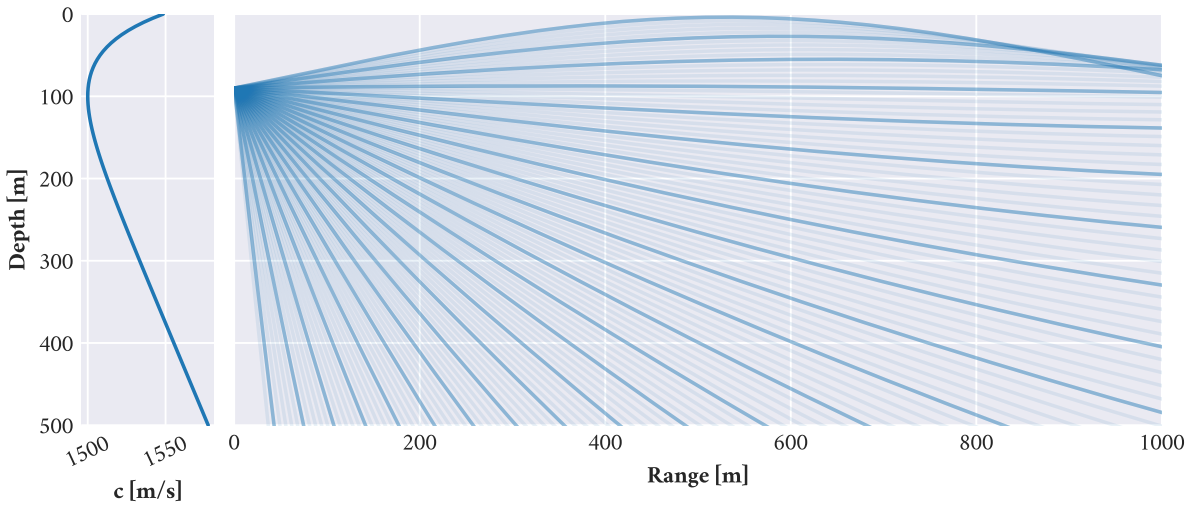


Figure 7.4: Ray tracing solution for direct paths in the Munk-shaped profile with the minimum at 100m, computed via BELLHOP. Traces with steep launch angles and surface bounces are not shown.

longest distance recorded during the experiment, between the vehicle and any beacon in the ICNN, was of approximately 4 km. Had there been no foul weather cutting the experiment short, longer ranges might have been recorded in subsequent deployments.

To that end, the cases shown in Figures 7.5 and 7.6 extend the sampled domain in range and depth. The second scenario in this case study covers a range of 2000 meters and a depth of 4 kilometers; the third scenario covers a slightly increased depth of 5000 meters and a longer range, out to 6 kilometers. The number of points in the grid remains unchanged across all scenarios despite the increased area – a limit driven in part by the computational resources available. Compared to the first case, both of these scenarios use another Munk-shaped profile, this time with the minimum at 500 meters.

For both cases, the sparser sampling in both depth and range leads to a break from the underlying physics – though the time front is still moving away from the source, the upward refracting effect of the lower water column appears to be lost in these images. The repeatability of the approximation is also significantly impacted; though some iterations of the process produce generally smooth field estimates, others see significant numerical artifacts enter the fold. PINNs are presumably su-

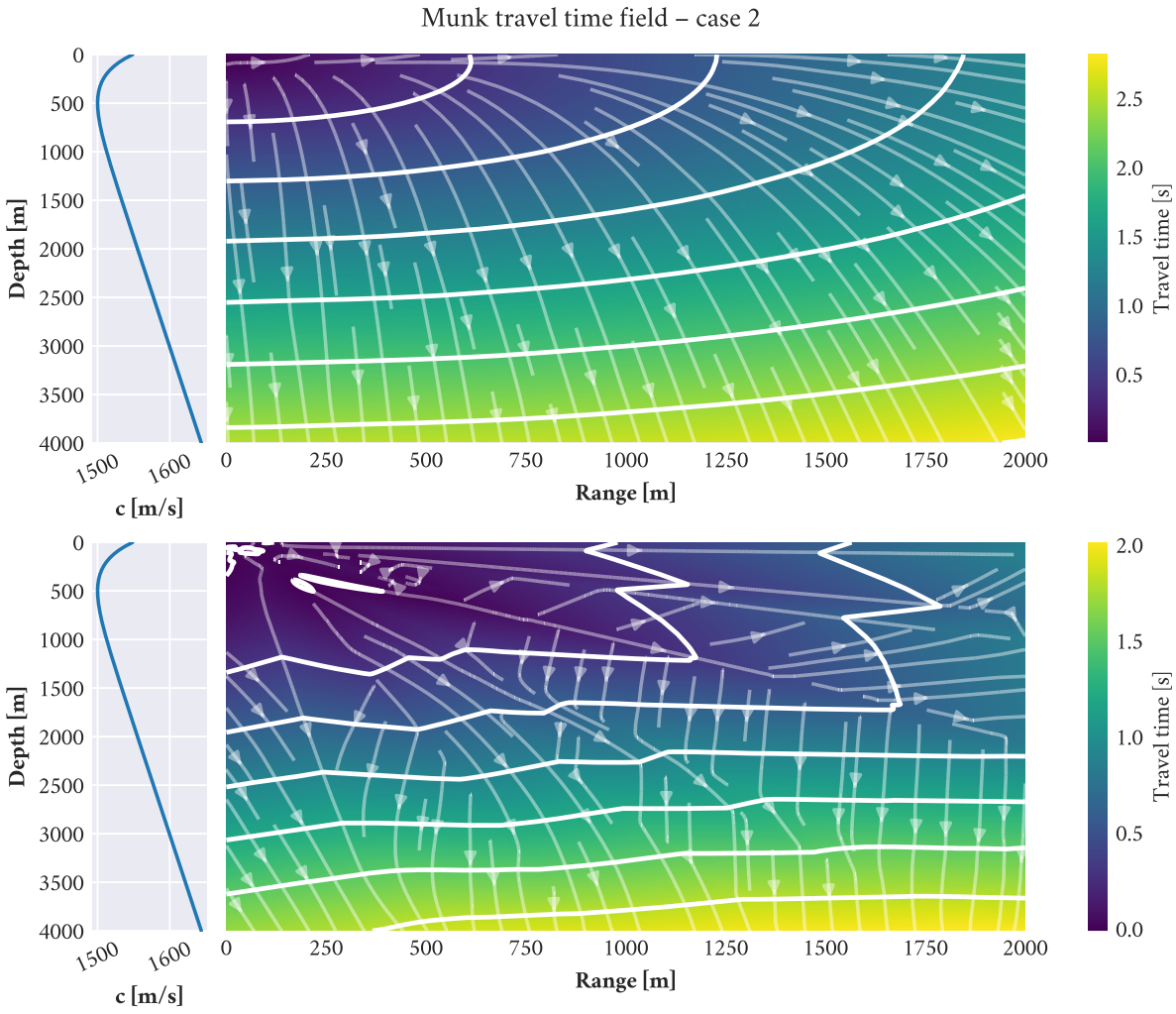


Figure 7.5: Learning the travel time field for a Munk-shaped environment (case 2). The sampled field extends to 2 km in range and 4 km in depth; the sound speed minimum is located at 500 m depth. With increasingly sparse sampling of the function space, the learning process becomes less consistent across repetitions; one instance may produce a smooth travel time field (top) while another execution of the same setup may produce a jagged field (bottom).

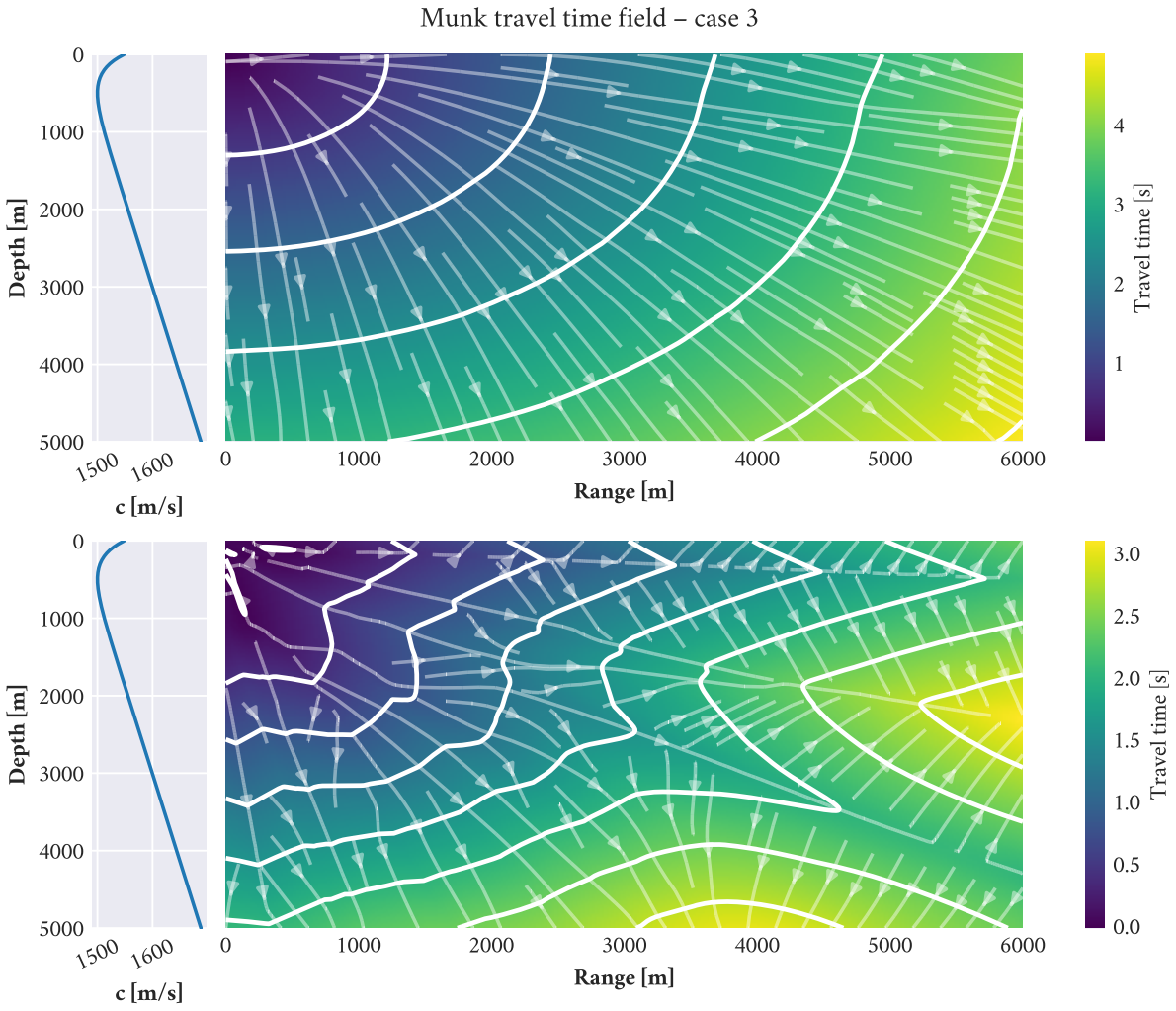


Figure 7.6: Learning the travel time field for a Munk-shaped environment (case 3). The sampled field is further extended to 6 km in range and 5 km in depth; the sound speed minimum is located at 500 m depth. With even sparser sampling of the function space, the results of the training process become less reliable still. Once again, top and bottom illustrate two distinct repetitions of the training process, one producing a smooth field and the other a jagged field.

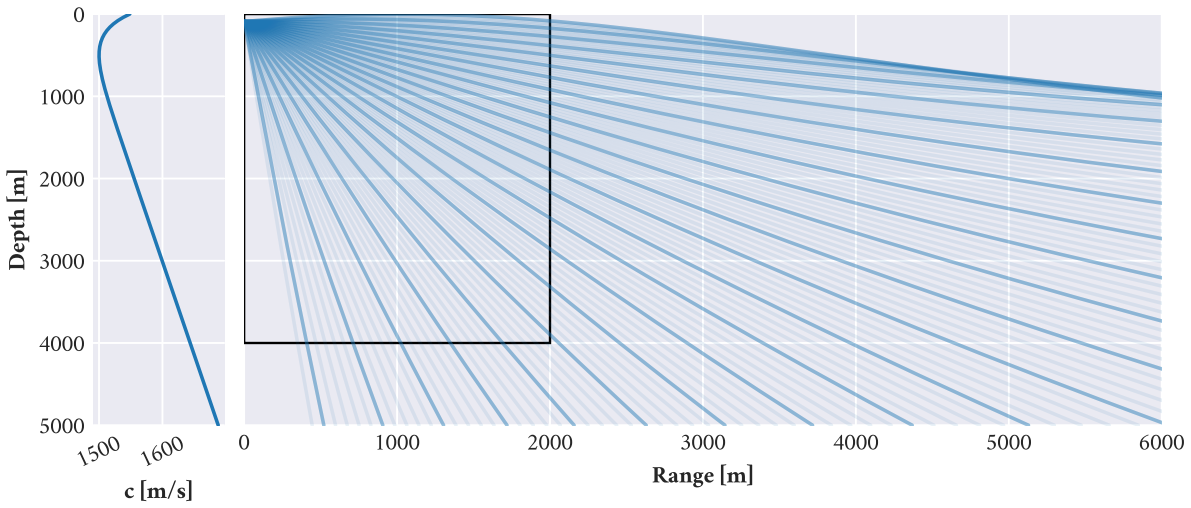


Figure 7.7: Ray tracing solution for direct paths in the Munk-shaped profile with the minimum at 500m, computed via BELLHOP. Traces with steep launch angles and surface bounces are not shown. The extents are matched to the longest-range scenario (case 3); the bounds marking the solution for case 2 are also shown.

pervised by the underlying physics; in practice, this supervision is actually executed in a numerical domain. Thus, the neural network used for these examples might lack the information needed to enforce the continuity of the field all the way from the source and across the increasingly large gaps between samples, compared to the denser grid from the first scenario[18]. It should be remarked that, although the limitations of the Eikonal equation in capturing multipath information were discussed earlier in the text, the BELLHOP ray tracing solution for this environment (Fig. 7.7) reveals that the sampled domain exhibits a smooth field; the poor performance is not a product of a complex environment.

Stated from the perspective of an operator in the field, the expectation set forth by learning a solution field with PINNs (or another machine learning solution) is that a user should later be able to sample any set of coordinates within the network's training domain, and the neural network should produce sensible values for the chosen coordinates. However, the approach exemplified

[18] From an intuitive perspective, the effect of sparse spatial sampling can be compared to that of the sampling frequency in time-series signal processing; collecting measurements at a rate lower than twice that of the target signal leads to signal aliasing.

here is highly sensitive to the spatial sampling used to train the network in the first place. The PINN will only perform as expected when the data used to train the network provided enough local information around regions of strong variability, such that the network could capture the salient features. Enough global information must also be provided, to ensure the continuity of the field across the network's domain – it was here that the impact of increasingly sparse sampling of the field became a challenge for the cases shown in Figures 7.5 and 7.6.

As shown with the first of the three demonstrations, and with the example applications covered by the related literature, a robust implementation of PINNs could ultimately serve as a powerful inversion solution in post-processing applications. However, their viability for a learning framework trained in the field is highly sensitive to the complexity of the model, the spatial distribution of the samples and the amount of data processed at each training step.

In terms of performance, each instance of the PINN presented in this section trained a total of 8101 parameters using 500,000 samples spread across the different grids. Each training cycle, spanning 5000 epochs, took approximately one hour on a first-generation NVIDIA AGX Xavier sporting a 16GB memory bank. The AGX is an edge computing solution designed for machine learning applications; while not intended as an optimal training platform for complex ML programs, its relatively moderate power budget (the device can be configured for 10W, 15W, or 30W operation) makes it a sensible candidate for embedded applications. Running the same code on a 48-core Xeon system with 64GB of memory took twice as long on average.

7.5 Environment estimation as a data assimilation problem

The central motivation behind this thesis is that, once deployed, autonomous systems often have access to information about their environment that would not be available to a human operator during the mission (due to a limited-throughput communications link or a lack of contact entirely); information that may directly influence the likelihood of success for a given deployment. The task

of interpreting said information can be made notably challenging when the samples are sparsely distributed, as is often the case with Autonomous Underwater Vehicles; but having some concept of that interpretation process programmed in the autonomy stack is necessary if the vehicle is to benefit from the available data.

As part of that interpretation process, the importance of using simple inversion algorithms for real-time training systems – whether they are built around PINNs or otherwise – should not be surprising. High-fidelity models require larger sample sets to constrain the increased number of parameters, and the resources required to perform the inversion must grow accordingly. Even with regards to forward-problem resolution, higher fidelity models may fall beyond the reach of an AUV's computational resources, as was discussed in Section 4.1.1. Rather than seeking to learn a high-resolution model, the aim with this environmental adaptation framework has been stated as exploiting different data streams in a unified approach, to compensate for the limitations of each individual data source. Much like prior work in ocean acoustic tomography,[19] the environmental adaptation problem is also a data assimilation problem.

7.5.1 A simplified inversion model

The aforementioned ocean tomography work relied on four defining equations to perform the inversion. These equations represent: (1) a direct measurement model, which relates data from a CTD to the true sound speed profile; (2) an oceanographic model, which may capture regional circulation data; (3) an acoustic propagation model, which relates the sound speed profile to the acoustic pressure field; and (4) an acoustic measurement model, relating the pressure measurements to the true acoustic field. Other work by the same author also discussed travel time tomography,[20] which was introduced in Section 2.3.1. The travel time formulation is of interest to this work, since the acous-

[19] Elisseeff, Schmidt, and Xu, "Ocean Acoustic Tomography as a Data Assimilation Problem".
[20] Elisseeff, "Fast acoustic tomography of coastal, tidally-driven temperature and current fields".

tic data available to AUV *Macrura* and the ICNN is the processed output of WHOI Micromodems rather than the raw time-series data.

In a similar fashion to what is done for tomographic work, then, Equation 7.8 can be modified to use the linear travel time measurement model in place of the unknown function $f(\mathbf{w}, \tau)$. A measurement related to travel time, τ_m, is approximated by an observation matrix B that transforms the perturbations $A\mathbf{w}$ of the baseline sound speed profile (Eq. 7.9). Note that in Eq. 2.12, the observation matrix was given as C, in accordance to the literature references; it is renamed here to avoid confusion with the direct measurement vector c_m.

$$\mathcal{L} = \underbrace{\|c_m - A\mathbf{w}\|}_{\text{fit CTD data}} + \underbrace{\|\lambda I\mathbf{w}\|}_{\text{regularizer}} + \underbrace{\|\tau_m - BA\mathbf{w}\|}_{\text{fit timing data}} \tag{7.9}$$

7.5.2 Travel time sample selection

As was discussed in the background, the observation matrix B captures the ray sampling function for each eigenray connecting a source and receiver pair, thus encapsulating information about the base environmental model and the spatial arrangement of the collected samples. Assuming a good enough baseline, measurement repetitions along the same base path could then be used as an indirect measurement of the sound speed profile, effectively adding a regularization term to the weights vector $\mathbf{w}$. However, the analysis in Section 6.2 showed that the eigenrays varied significantly across the example Arctic environments, especially for the near-surface paths. Furthermore, the nature of the experiment (and the eventual failure of the vehicle's propulsion control computer) meant that some range-depth combinations were densely sampled, while many others were not sampled at all. Figure 7.8 shows the coverage maps for acoustic samples collected by AUV *Macrura*, and by the ICNN buoys.

In order to demonstrate the application of a machine learning framework to assimilate both direct and indirect measurements of the environment, the set of samples collected by the ICNN in

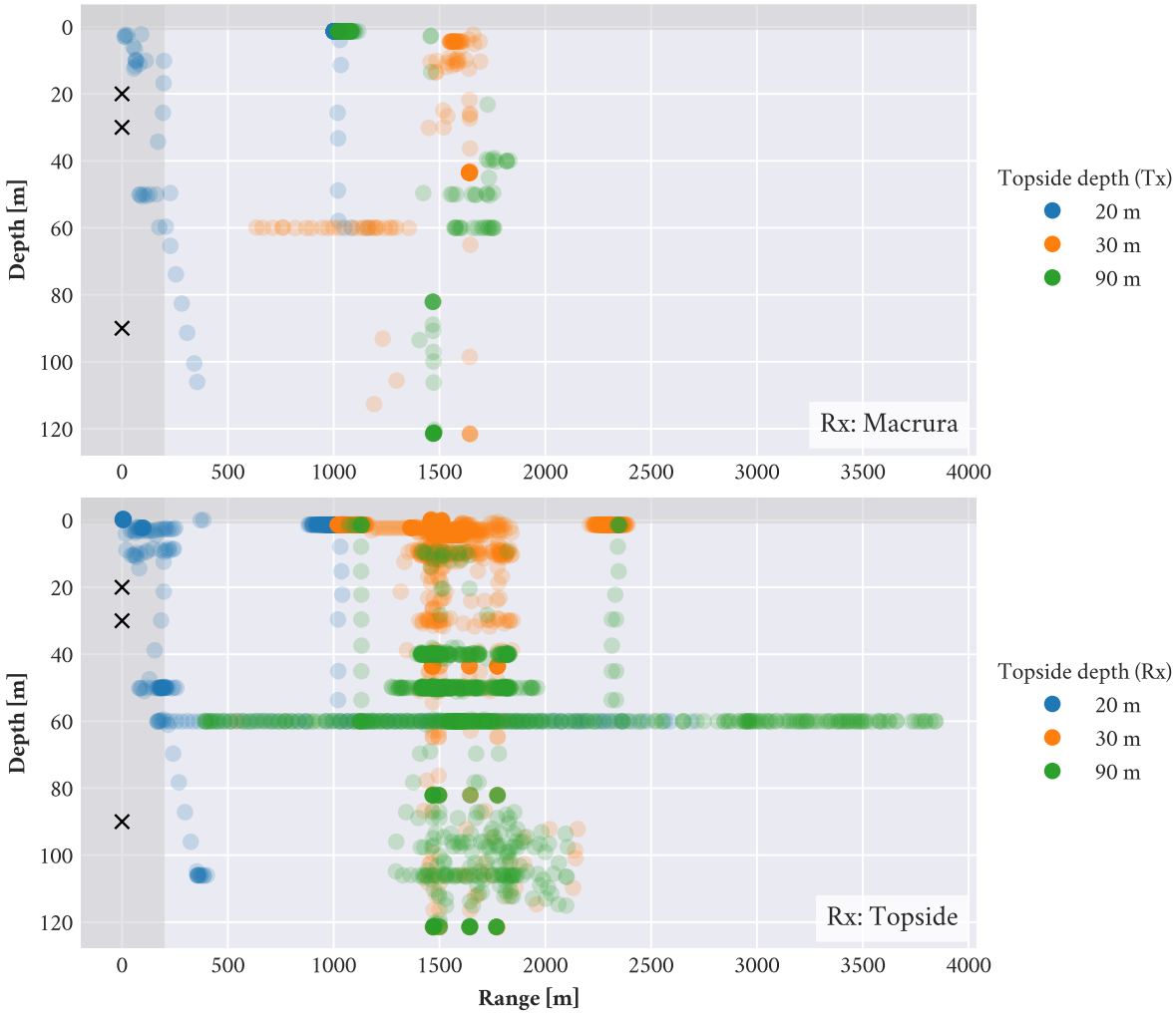


Figure 7.8: Coverage of the range-depth domain for one-way travel time samples collected by AUV *Macrura* (top) and the ICNN buoys connected to the topside computer (bottom), for acoustic links between topside and the vehicle; buoy-to-buoy samples are not shown. Ranges are given relative to the ICNN buoys, with the different depths illustrated. The markers indicating the range and depth of *Macrura* are color-coded by the depth layer of the ICNN transmitter or receiver involved in the link.

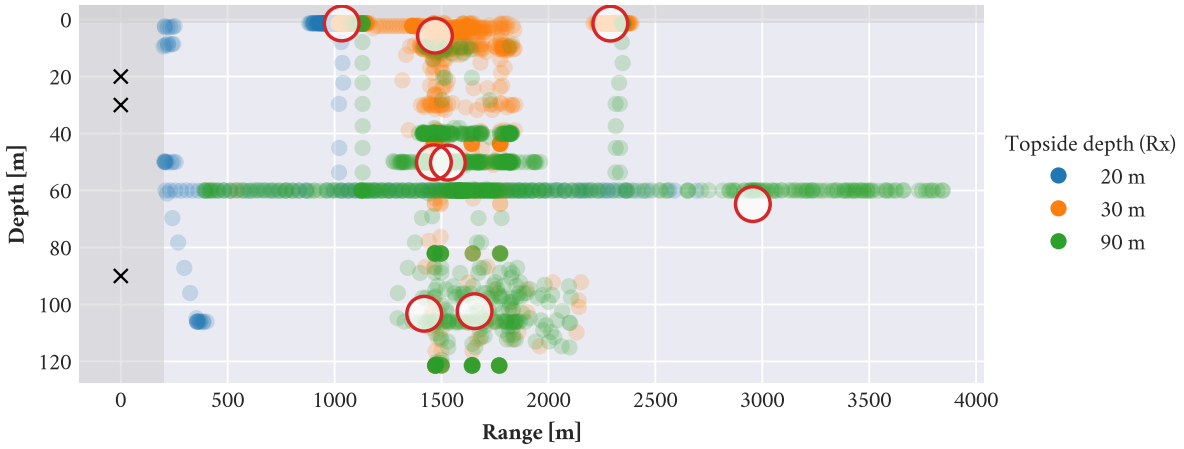


Figure 7.9: Sample selection from the set of acoustic data recorded by the topside-controlled ICNN. A k-means approach with 8 clusters is used to identify representative regions in the set.

the field are first reduced to representative examples, using a k-means approach. The clustering algorithm uses the sample's range, depth and measured travel time, as well as the depth of the ICNN component involved; all domains are normalized across the set of samples, to ensure they share equal influence in the optimization process. The resulting cluster centers for $k = 8$ are shown in Figure 7.9.

7.5.3 Learning the travel-time observation matrix

Generally speaking, the objective behind tomographic experiments is to solve the inverse problem and produce as realistic a model of the environment as may be possible from the measurements available. However, the concept of real-time environmental adaptation is somewhat less concerned with obtaining this detailed model; instead, the adaptive system is more concerned with identifying correction terms for its internal model, which may be used to improve its performance in the field. In other words, the environmental adaptation problem is focused on attaining incremental gains.

Recognizing both the preference for conservative operations in AUV deployments, and the exploratory capabilities of ML techniques, this section modifies the weight identification problem that

was first given in Equation 7.6. Rather than enforcing the ray sampling functions given by Equation 2.13 in the observation matrix B (Eq. 7.9) to solve for the weight vector $\mathbf{w}$, the problem is re-framed as solving for the terms in B which would be required to fit the travel time data with respect to the EOF-fitted sound speed profile $A\mathbf{w} + c_b$, where c_b represents the baseline SSP. This new system configuration is still subject to fitting the CTD measurements with the weight vector; recall that c_m represents the depth-dependent perturbations of the baseline profile, and not the discretized CTD samples directly. This form of the problem is given in Equation 7.10.

$$\underset{\mathbf{w},B}{\arg\min} \underbrace{\|c_m - A\mathbf{w}\|}_{\text{fit CTD data}} + \underbrace{\|\lambda I\mathbf{w}\|}_{\text{regularizer}} + \underbrace{\|t_m - B(A\mathbf{w} + c_b)\|}_{\text{fit timing data}} \tag{7.10}$$

The expectation with this variant of the problem is that, rather than forcing the time measurements to directly influence the values in the weight vector, B will instead capture information relating the travel time data and the direct environmental measurements. These should generally be analogous to the ray sampling functions, but are not expected to directly reflect true eigenray information, as the model has not been given any such information and the basis is now given in terms of the true fitted profile rather than the perturbations.

After training, the rows of B revealed functions similar to the fitted profile. This relation could be confirmed by zero-shifting and normalizing each of the rows and the sound speed profiles. Reverting the shifting and scaling process, using the values from the fitted profile for the rows of the observation matrix, yields the results shown in Figure 7.10.

The practical application of a pseudo-tomographic approach like the one demonstrated here would enable the autonomy system onboard a vehicle to look beyond the bounds of its internal environmental model and the EOF representation. Training the observation matrix using this approach produces perturbations of the EOF-fitted environment, which could be fed to the Virtual Ocean Autonomy Testbed as secondary models for further assessment. Indeed, the concept of ex-

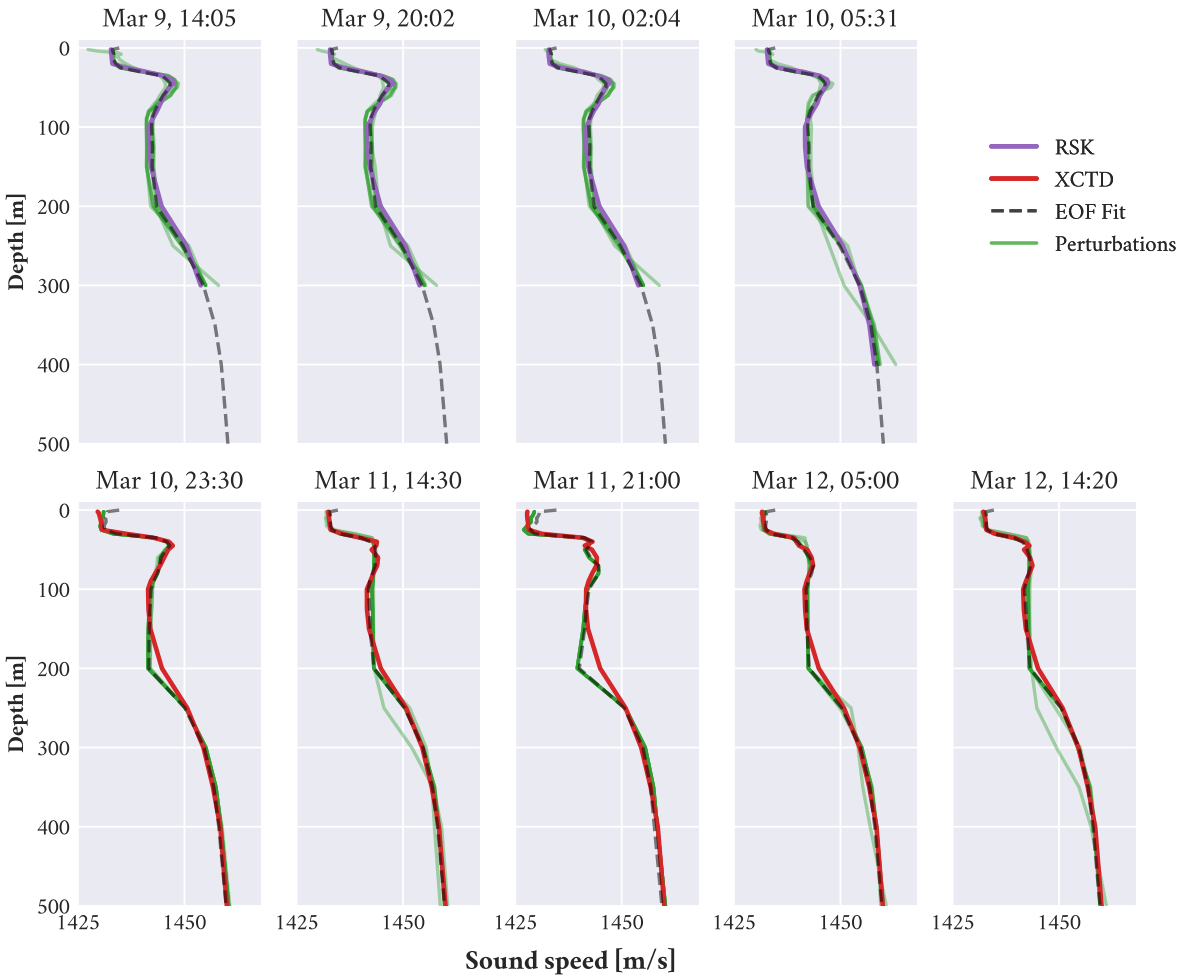


Figure 7.10: CTD casts collected during ICEX-20, along with the EOF fit obtained via conventional least-squares. The rows of the travel time observation matrix B are scaled and shifted to align with the data, illustrating how the training process captures perturbations of the EOF-fitted profile.

ploring multiple forms of the environment approximation has also been tackled with Tactical Decision Aids, for human-in-the-loop operations; however, those representations were all fully bound by the choice of basis functions.

7.6 Summary

This chapter revisited the topic of machine learning, exploring some of the considerations that affect ML use in oceanographic and environmental applications, as well as advances in ML frameworks for physical systems described by partial differential equations. The potential benefits and limitations of using Physically Informed Neural Networks in field applications were discussed with respect to a benchmark example in the form of a Munk-shaped sound speed profile, performed in three parts. Finally, a pseudo-tomographic approach to producing exploratory perturbations of an EOF-driven environmental model was demonstrated. These altered environmental models complement the combinatorial approach of the EOF-driven Tactical Decision Aid used during ICEX-20, and could be evaluated with bespoke tools or the Virtual Ocean Autonomy Testbed to validate or modify the vehicle's internal model.

8 Conclusions

"People have forgotten this truth," the fox said. "But you mustn't forget it. You become responsible forever for what you've tamed. You're responsible for your rose."

— Antoine de Saint-Exupéry, *The Little Prince*

This thesis brings a magnifying glass to some of the complexity involved in the operation of Autonomous Underwater Vehicles, with the intent to improve on the vehicle's environmental adaptation capabilities. The need for environmental adaptation can be appreciated when looking back on prior experience conducting AUV deployments in the new Arctic, where a layer of warm Pacific water creates a variable double-ducted acoustic environment that exacerbates the challenges of underwater navigation in a GPS-denied environment.[1] To tackle the challenge of environmental adaptation, this thesis presented a multidisciplinary approach centered on exploiting information available to the AUV or the vehicle operator in a collaborative framework.

8.1 Summary of contributions

In order to demonstrate the impact of foundational machine learning techniques, this thesis first presented a framework that exploits bearing data and time-to-intercept data to classify the behavior of a neighboring vehicle; both signals can be drawn by processing passive acoustic data. Building

[1] Schmidt and Schneider, "Acoustic communication and navigation in the new Arctic — A model case for environmental adaptation".

on this example, the classification framework was extended to a simplified example of environment characterization, presented in the form of riverbed material identification. This extension benefited from local surveys to inform the model generation process used to create the training set provided to the classifier. (Chapter 3)

Following this foundation, the text moved on to explore how similar data-driven techniques may be applied to the problem of environmental adaptation in the Arctic Ocean. As part of the experiments required to demonstrate such capabilities, the author was responsible for the vehicle's autonomy payload and other computational resources needed to support deployments in the Arctic. Additionally, simulations were performed ahead of field deployments in order to identify opportunities for improvement within the reach of the AUV's sensing capabilities. (Chapter 4)

An important contribution of this thesis consisted of an in-depth assessment of the data collected during ICEX-20. This breakdown of the mission logs and measurements supported subsequent work in this text, and also enabled contributions by other members of the Laboratory for Autonomous Marine Sensing Systems (LAMSS), who were likewise presenting research based on these data sets. The assessment included the identification of clock synchronization issues and the characterization of the acoustic data collected in the Arctic, as well as a discussion of the system limitations that led to the unavailability of some of the anticipated measurements. (Chapter 5)

Building on the pre-deployment simulations and the data assessment, this thesis then presents the range estimation algorithm fielded during ICEX-20, along with an improved solution that was developed in post-processing. Both methods are aided by acoustic propagation models. The modified algorithm exploits information about feature availability that was identified in the preliminary simulations. The added dimension of information was shown to improve the performance of the range estimation and the subsequent acoustic positioning calculations produced by the Integrated Communications and Navigation Network (ICNN) – the acoustic tracking network used during ICEX-20. (Chapter 6)

Lastly, this thesis expanded on the aforementioned contributions by demonstrating a pseudo-tomographic approach to environment estimation, implemented with the help of a machine learning framework. The perturbations of the environmental model that were identified using this approach could be used to introduce a slight bias to the vehicle's autonomous decision process, as part of meeting the vehicle's operational requirements – an example of this being that the vehicle may need to maintain a reliable acoustic communication path available at all times. (Chapter 7)

8.2 Opportunities for improvement and future work

8.2.1 Data management in the field

In her book about the early days of World War I, renowned historian Barbara Tuchman notes that *"in the midst of war and crisis nothing is as clear or as certain as it appears in hindsight"*.[2] The same might be said of scientific endeavors, where the process of exploration and discovery oft exposes clear avenues for improvement that might be pursued in subsequent iterations. Such is the case with the most forthright next steps discussed herein, which aim to tackle the limitations brought to light in the ICEX-20 field report (Chapter 5).

First in this list, therefore, comes the thought of expanding the concepts that led to the improved range estimation algorithm presented in Chapter 6 – the Nearest Bounce Criteria (NBC) – to reach beyond the model scope. As reported in Chapter 5, many of the data points available in the acoustic logs were left unused by the ranging solution fielded during ICEX-20, due primarily to corrupted data frames even in the presence of valid and verifiable headers. The design decision to reject any failure in the acoustic network was based on a conservative approach to operations, in the absence of evidence supporting an alternative path; evidence that may well come from the data assessment reported in this work. Even when multiple successful arrivals were detected at any of the buoys

[2] Tuchman, *The Guns of August*.

for a given transmission event, the expectation of a single valid arrival per buoy was enforced in the ICNN's codebase. However, by enabling an acoustic tracking system such as the ICNN to exploit additional information from the field measurements, in the form of multiple distinct arrivals at any of the tracking network's nodes, the performance of the tracker could be improved. Potentially weighted by the system's confidence in the sample (encoding the extent of the failure in the communications framework, for example), the additional range estimates could provide additional information to constrain the uncertainty of the tracking solution. Such a consideration further emphasizes the added value of the NBC, as it necessarily recognizes the multi-path nature of the acoustic environment in the new Arctic.

The second avenue for improvement exposed in the data assessment from Chapter 5 relates to the issues the LAMSS team experienced with the radio link between the ICNN buoys and the topside control station at Camp Seadragon. To understand how this path for improvement becomes apparent, some familiarity with the overhead costs of digital communication systems can be helpful. This overhead may be used to record the source and destination addresses, or a specific packet number for larger records that must be split into a series of transmissions; such examples apply to the Internet Protocol, which one could argue is usually taken for granted in the present day. This additional metadata makes it possible to implement mechanisms to validate the integrity of the data at the receiving node, for example; but it also cuts into the effective throughput of the system. Recognizing both the value of this overhead and the limitations inherent to acoustic communication systems has thus motivated the development of encoding optimizations for the headers that constitute this overhead, and for the transmitted content itself.[3]

Drawing from the work that's been put into acoustic communications, then, it follows that a careful assessment of the ICNN's data exchange needs may ultimately enable subsequent deploy-

[3] Schneider, "Transmitting Internet Protocol packets efficiently on underwater networks using entropy-encoder header translation"; Schneider, Petillo, Schmidt, and Murphy, "The Dynamic Compact Control Language version 3".

ments to collect the records that were missed during ICEX-20. The radio links used to connect the ICNN buoys to the operator computer were expected to provide more than enough capacity to handle all the data produced by the various remote acoustic modems – yet the data loss incurred in the form of absent impulse response estimates proved otherwise. Redressing the implementation of the radio links to bundle modem signals into larger packets, for example, could improve the resulting overhead ratio. A logical counterpoint to buffering the signals in such a way would be that doing so would introduce further delays in the information exchange between the remote modem and topside – but this point is mainly relevant in the context of establishing and maintaining the time synchronization between these systems. However, time queries and time setting commands involved in this process are already subject to variability in the radio interface; this too was discussed in Chapter 5. Thus, the system might benefit from abandoning the centralized control scheme implemented during ICEX-20, and instead transferring some of the responsibility for time synchronization to each of the buoys' GNSS receivers. Doing so would reduce the number of messages exchanged between the operator computer and the ICNN buoys, relieving some of the pressure on the radio network. Should the reduction of traffic on the network not be enough to ensure the system's needs are met, additional modifications such as the aforementioned stacking of packets into fewer transmissions, or the encoding of these transmissions into more efficient representations, could also be implemented.

8.2.2 Parallelizing the acoustic modeling tools

As stated in Section 2.2, there are numerous techniques that can be used to tackle the problem of acoustic propagation. The most commonly used implementations of these techniques for ocean applications can be accessed through the Ocean Acoustics Library (OALIB[4]), which serves as a central hub for the distribution of both software and data in this field. Outside of bespoke implementations

[4]https://oalib-acoustics.org

used in post-processing, the work presented in this text made use of two of the packages in this library. The section on riverbed characterization used the wavenumber integration code OASES, written by Prof. Henrik Schmidt. The components related to the Virtual Ocean, the ICEX-20 deployments and acoustic navigation in the Arctic used the ray tracing program BELLHOP, written by Dr. Michael B. Porter. These programs share two key commonalities of interest to the pursuit of paths for improvement: both are written in FORTRAN, and both are written as serial code.

The first commonality between these software suites is worth mentioning for the impact it has in terms of performance and maintainability. While not as popular as other programming languages in the wider software development community, FORTRAN is still in active development and is well established within the scientific community due to its speed when performing complex numerical calculations, and to the existence of legacy code in highly critical systems across the globe.[5] This seemingly niche positioning means that qualified developers and maintainers may be few and far between when compared to other languages; but they certainly exist. More importantly, the language is natively parallel and can be used in conjunction with NVIDIA's CUDA, which is perhaps the most popular solution to date for general purpose computing on graphics processing units (GPUs).

The second commonality lies in the fact that the current code for both projects is implemented in a serial approach, which has measurable consequences in both simulation and real-world deployments. As an illustration of this point, Section 4.1.1 addressed how the Virtual Ocean Simulator developed by the LAMSS team uses a nested modeling approach to handle this limitation, exploiting the various time scales of related processes to balance the need for up-to-date information with the cost of running the acoustic propagation model on the same CPU resources as the rest of the autonomy system. But as the number of nodes is increased – for example, by adding buoys to the ICNN – a larger number of model executions are required within a given time window, to reflect the

[5] Kedward, Aradi, Certik, Curcic, Ehlert, Engel, Goswami, Hirsch, Lozada-Blanco, Magnin, Markus, Pagone, Pribec, Richardson, Snyder, Urban, and Vandenplas, "The State of Fortran".

additional acoustic contacts; the nested model architecture alone may not be sufficient to support scaling operations in this way.

The issue at hand can ultimately be reduced to the processor time required for each execution of the propagation models. The limitations of this serial code execution are emphasized when running simulations with an accelerated time scale, since this time-warped approach effectively compresses the gap between iteration cycles of each of the apps in the autonomy ecosystem and is limited by the number of processor cycles left unused in the corresponding real-time execution. Where many of the applications in the autonomy stack are executed around four times per second but require a relatively small amount of work per iteration, these can be accelerated to double-digit time factors without much trouble, on the development platforms typically used by the members of the LAMSS team. In contrast, re-processing the ray tracing files collected during ICEX-20 takes about 0.2 to 0.3 seconds typically, with some instances spilling beyond a full second in duration – and these times reflect only the execution of the ray tracing program BELLHOP, in the absence of other autonomy applications competing for resources and without accounting for the steps involved in collecting updates for the environmental model or in converting the resulting grid into a travel time prediction. In short, the serial nature of the code affects both the scalability and the time warp compatibility of the Virtual Ocean.

It may be apparent by now that the improvement route emphasized here is the modernization and parallelization of the code contained in the Ocean Acoustics Library. As part of an internal review conducted by the author of this text, a bespoke implementation of ray tracing was shown to benefit from as much as a 10x acceleration factor, when converting from a serial CPU approach to a parallelized GPGPU solution executed on a first-generation AGX Xavier with 16 GB memory bank. These gains were obtained with only a basic port of the bespoke code into the GPU, and the entire demonstration was built natively in CUDA; further optimization should be attainable with a more careful adaptation of the routines. Converting the OALIB projects, regarded as golden standards

within the ocean acoustics community, should not require a complete rewrite – instead, the process would benefit from the compatibility of FORTRAN kernels in the CUDA ecosystem; and doing so may ultimately enable the use of time-warped simulations, and improve the scalability of systems like the Virtual Ocean Simulator.

It should be noted, of course, that the idea of parallelizing these acoustic propagation suites is not new. As maintainer of the OASES repository, the author of this thesis supported the integration of a parallelization wrapper for the wavenumber integration code into the upstream project. The wrapper was developed by Dr. Gaute Hope in collaboration with Prof. Henrik Schmidt, and was dubbed PAROASES for the nature of its function.[6] However, PAROASES takes a high-level approach to parallelization, splitting the model into subsets by the independent nature of the different frequencies and executing multiple calls of the CPU-bound, serial code. Similarly, the ray-tracing models executed for the post-processing portions of this thesis were often sent to a server with a large CPU core count, to take advantage of multiple concurrent threads. Neither of these approaches takes advantage of increased acceleration via GPU; and while prior efforts to upgrade BELLHOP using CUDA do exist in the literature,[7] the resulting project has not seen widespread adoption. A successful modernization of the OALIB will likely require the collaboration of new developers with the maintainers of the upstream projects, to ensure the upgrades are thoroughly tested and eventually merged into the well-known software suites to reach the wider network of users.

8.3 Closing remarks

Just as the Arctic captured the imagination of ancient Greece, so it captures the minds of the present. The region has earned increased interest from fishing, trade and military operations – after all, the vision of a blue Arctic comes with the promise of a convenient northern passage. It has also been

[6] Hope and Schmidt, "A parallelization of the wavenumber integration acoustic modelling package OASES".
[7] Lazzarin, "Parallel implementation of a ray tracer for underwater sound waves using the cuda li- braries: description and application to the simulation of underwater networks".

the subject of studies seeking to understand the role that the ice cap may play on changing climate conditions. This interest in the frigid northern waters is embodied in the biennial nature of the US Navy's Ice Exercise (ICEX), which aims to demonstrate military capabilities and also to support select scientific efforts; in addition to this thesis, other works[8] likewise benefited from the data collected by the LAMSS team during the 2016 and 2020 ICEX deployments. Furthermore, while members of the lab's team were on site, Ice Camp Seadragon was overflown by a Tu-142 maritime reconnaissance aircraft. Whether the plane's objective was to observe the US Navy submarines in the area, specifically, or the camp's operations more generally, the event serves to illustrate how even now, the Arctic is at the center of competition between global powers. Alas, advances in marine vehicle autonomy have the potential for direct impact in efforts to monitor conditions in the Arctic, which may ultimately influence commercial and military decisions to gain the upper edge in said competition.

Looking beyond the global power dynamics involving the Arctic, experiments like those conducted by LAMSS during ICEX-16 and ICEX-20 are only possible thanks to significant logistics operations involving a wide arrange of teams. Getting equipment and personnel across across the country, and eventually airlifted onto the ice, is no small feat; performing recovery operations like the one described in Section 5.5 are also only possible only with great effort to coordinate many competing resources and interests. Pilots, oceanographers, ice mechanics experts, medical personnel, and many others are essential to enable this type of work – and if the mission is to succeed, they are need to be trusted to perform their duties. And much like with all the human personnel, AUV *Macrura* also had to be trusted to perform as intended. Though an in-depth discussion of the role of trust in human-robot interaction is beyond the scope of this work (this area of research features

[8] Bhatt, "A Virtual Ocean framework for environmentally adaptive, embedded acoustic navigation on autonomous underwater vehicles"; Chen, "Ambient Acoustics as Indicator of Environmental Change in the Beaufort Sea: Experiments & Methods for Analysis"; Goodwin, "Environmental Effects of the Beaufort Lens on Underwater Acoustic Communications during Arctic Operations"; Howard, "Multipath Penalty Metric in Underwater Acoustic Communication for Autonomy and Human Decision-making".

the likes of Project Aquaticus[9]), it is nonetheless a necessary element when deploying autonomous systems – indeed, the goal of improving the vehicle's environmental adaptation capabilities for autonomous operation may ultimately help create more trustworthy robotic platforms in the future.

[9] Robinette, Novitzky, Fitzgerald, Benjamin, and Schmidt, "Exploring Human-Robot Trust During Teaming in a Real-World Testbed".

A Python MOOS and the LAMSS Python toolkit

The `python-moos` project provides language bindings to the original MOOS code, making it possible for developers to exploit the fast prototyping benefits of the Python language while interfacing with the underlying C++ code at the heart of the MOOS and MOOS-IvP projects, along with their associated ecosystem of autonomy software. The first commit for the `python-moos` interface is dated back to 2013, when Prof. Paul Newman (author of the MOOS middleware) began working on the language bindings using the Boost.Python libraries as a foundation.

Although the last commit recorded for Prof. Newman's original project is dated back to May of 2016, the language bindings have continued to grow at the hands of various MOOS users, including the author of this thesis. In December of 2016, Dr. Mohamed Saad Ibn Seddik began the effort of rewriting the bindings to use the PyBind11 library. The Boost.Python used in the original project provides a great range of backwards-compatibility and a rich feature set, but these perks come at a cost in terms of size. Shifting to the lightweight, headers-only PyBind11 thus introduced savings in terms of size costs associated with the project's dependencies. In April of 2020, Russ Webber joined in with fixes to the compilation and testing infrastructures, as well as updates to the PyBind11 implementation.

The author of this thesis joined the development efforts in November of 2020 after being a regular user for some years, in order to provide fixes to the Continuous Integration and Continuous Delivery (CI/CD[1]) infrastructure. The main goal was to ensure that potential users had easy, reliable access to any future updates to the bindings as they became more deeply integrated with the LAMSS operational paradigm.

A.1 The Python-MOOS bindings

The following serves as an introductory guide to the `python-moos` bindings, and how to use them for application prototyping and development. This guide is intended to facilitate the iterative nature of the design process; for well-established algorithms in critical autonomy applications, such as those that may be embedded into AUVs, it is still recommended that users consider translating finalized solutions to C++, as the compiled program will generally perform better than interpreted code.

A.1.1 Installation

The `python-moos` bindings are automatically installed as part of the standard LAMSS dependencies. However, they can also be installed easily for other MOOS-related systems. In order to do so, the minimum requirements are: (1)MOOS must be compiled and installed; (2) an appropriate Python installation must be available on the system, and (3) the Python package manager, `pip`, must also be installed. When all requirements are met, the `python-moos` bindings can be installed via `pip`:

```
python3 -m pip install pymoos
```

[1] CI/CD is alternatively expanded as Continuous Integration and Continuous *Deployment* when the updates are not just delivered to end users, but are in fact made active immediately on connected systems.

The previous command will fetch the project from the Python Package Index (https://pipy.org), perhaps the most popular solution for package management in the Python ecosystem. Alternatively, the project can also be installed from source:

```
git clone https://github.com/oviquezr/python-moos.git python-moos
cd python-moos
python -m pip install .
```

The compilation and installation can also be executed via the following commands:

```
cd python-moos
python3 setup.py build
python3 setup.py install
```

A.1.2 Understanding the bindings

As of this writing, the bindings include connections to the MOOS Message and the MOOS Communications Client components, as shown by the header inclusions in the source file `PyMOOS.cpp`:

```
#include "MOOS/libMOOS/Comms/MOOSMsg.h"
#include "MOOS/libMOOS/Comms/MOOSCommClient.h"
#include "MOOS/libMOOS/Comms/MOOSAsyncCommClient.h"
```

At the top level, this approach gives the user access to some of the core MOOS tools, such as time references and time warping. The synchronous and asynchronous Communications Client components enable the configuration of custom callbacks, and provide access to the variable registration and notification functions. Similarly, the MOOS Message bindings make it possible to access the same features of mail entries provided for C++ applications. The basic bindings needed to start using `python-moos` are given in Table A.1.

Table A.1: A sampling of basic bindings handled by the `python-moos` project. Parent namespaces are given in parenthesis for each group; the `msg` entry is an item from the vector returned by `pymoos.comms.fetch()`.

	C++	Python
Core MOOS tools	-	(pymoos)
	`MOOSLocalTime`	`.local_time()`
	`MOOSTime`	`.time()`
	`SetMOOSTimeWarp`	`.set_moos_timewarp()`
	`GetMOOSTimeWarp`	`.get_moos_timewarp()`
Synchronous Comms	(`CMOOSCommClient`)	(`pymoos.comms`)
	`::Register`	`.register()`
	`CMOOSCommClient::Notify`	`.notify()`
Async Comms	(`MOOS::MOOSAsyncCommClient`)	(`pymoos.comms`)
	`::Run`	`.run()`
Basic callbacks	(`MOOS::AsyncCommsWrapper`)	(pymoos.comms)
	`::SetOnConnectCallback`	`.set_on_connect_callback()`
	`::SetOnMailCallback`	`.set_on_mail_callback()`
	`::FetchMailAsVector`	`.fetch()`
Message interface	(`CMOOSMsg`)	(`msg` from `pymoos.comms.fetch()`)
	`::GetKey`	`msg.key`
	`::GetDouble`	`msg.double`

C++	Python
`::GetString`	`msg.string`

A.1.3 Getting started with Python-MOOS

The following sample code provides a minimal framework for a simple Python MOOSApp. The example provided will connect to a MOOSDB running on localhost and the default port (9000). Upon a successful handshake, it will register for the navigation variables `NAV_X` and `NAV_Y`. The iterate function will simply act as a heartbeat indicator; the main expression of this simple app is that, upon receiving mail associated with one of the requested variables, the `on_new_mail()` callback will print the information received.

As part of the given report, the mail handler will first print the current time as given by the underlying MOOSTime routine. It is important to note that the way MOOS handles time will be linked to the time warp configuration, so it is strongly recommended that apps be properly configured to match time warp across all active applications. Failure to do so may lead to unexpected behavior; when using `pLogger` to record the mission, the issue may become evident after inspecting the logged message times.

```
import pymoos
import time

# Set desired time warp
pymoos.set_moos_timewarp(1)

# Collect reference value for duration control
start_time = pymoos.time()

# Shorthand for the Async Comms Wrapper
pmc = pymoos.comms()
```

```
def on_connect():
    for var in ['NAV_X','NAV_Y']:
        pmc.register(var,0) # register for MOOS variables
    return True # REQUIRED: callbacks must return True

def on_new_mail():
    for msg in pmc.fetch(): # handle mail in queue
        print('Time   :',pymoos.time())
        print('Key    :',msg.key())
        print('Double :',msg.double())
        print('String :',msg.string())
    return True # REQUIRED: callbacks must return True

def iterate():
    print('.',end='',flush=True)

pmc.set_on_connect_callback(on_connect)
pmc.set_on_mail_callback(on_new_mail)
pmc.run('localhost',9000,'pymoos_app')
pmc.wait_until_connected(2000)

apptick = 4

while start_time+300 > pymoos.time():
    iterate()
    time.sleep(1.0/apptick)

# Close connection to MOOSDB
pmc.close(True)
```

A.1.4 Using Python-MOOS as part of a class

A more complex application, or set of applications, may warrant the use of classes. This approach would make it possible to handle specialized components of the applications in different source files, for example. Another advantage of the class-based approach, as is generally the case in software

solutions, is that it allows for common components to be centralized in a parent class, minimizing code duplication when possible.

The following code provides a simple example similar to the previous one, but this time using a class to define the application's instructions. The class in this code can be imported into another source file, but the file can also be executed directly; the top-level execution instructions are then given by the `__main__` routine defined near the end of the code.

```
#!/usr/bin/env python3

import numpy as np
import pymoos
import logging

# pymoos doesn't currently bind to CMOOSApp method
# for config parsing; use this for convenience
from lib_pylamss import moos_config

# Use uPlot's Ansi module for color-coding logs
from lib_pylamss.Ansi import Ansi as ansi

logger = logging.getLogger(__name__)

class SAMPLE_APP:
    '''
    This is a sample Python-MOOS application
    '''

    def __init__(self,*args,**kwargs):

        # Default server settings (override via config)
        self.server_host = 'localhost'
        self.server_port = 9000
        self.time_warp   = 1

        # Process args and read config
        self.name = args[2]
        self.cfg = {'config':args[1],'apptick':4,}
```

```
        self.read_config()

        # Initialize some variables
        self.got_stuff = False
        self.last_x = None
        self.last_y = None

        self.vars_to_reg = ['NAV_X','NAV_Y']

        # Log pymoos app's configuration:
        self.log_config()

        # start pymoos.comms, with callbacks
        pymoos.set_moos_timewarp(self.time_warp)
        self.pmc = pymoos.comms()
        self.pmc.set_on_connect_callback(self.on_connect)
        self.pmc.set_on_mail_callback(self.on_new_mail)

        self.pmc.run(self.server_host,self.server_port,self.name)
        self.pmc.wait_until_connected(2000)

    def iterate(self):
        if self.got_stuff:
            print()
            print('Time  :',pymoos.time())
            print('Last X:',self.last_x)
            print('Last Y:',self.last_y)
            self.got_stuff = False
            self.last_x = None
            self.last_y = None
            self.pmc.notify('SAMPLE_POST','the message')

    def handle_mail(self,msg):
        if msg.is_name('NAV_X'):
            self.last_x = msg.double()
        elif msg.is_name('NAV_Y'):
            self.last_y = msg.double()

        self.got_stuff = all([x!=None for x in [self.last_x,self.last_y]])

    # ================================================
```

```
    # moos_config parser in lib_pylamss for convenience
    def read_config(self):
        moos_config.read_config(self,self.cfg['config'],self.name)
    def log_config(self):
        moos_config.log_config(self)

    # ======================
    # pymoos.comms callbacks
    def on_connect(self):
        for var in self.vars_to_reg:
            logger.info(self.name +' Register for '+var)
            self.pmc.register(var,0)
        return True
    def on_new_mail(self):
        for msg in self.pmc.fetch():
            self.handle_mail(msg)
        return True

# ================================================
# ================================================
# This section defines how the program should behave
# when called directly, rather than as an import to
# another program
if __name__=='__main__':

    import time
    import sys

    # Logger should be configured by 'primary' program
    logging.basicConfig(
        level=logging.DEBUG,
        format=('%(asctime)s | %(name)s.%(funcName)-16s'
                +' %(levelname)-8s : %(message)s'),
        datefmt='[%Y-%m-%d %H:%M:%S]')
    logging.Formatter.converter = time.gmtime

    # create instance of our class
    app = SAMPLE_APP(*sys.argv)

    # call iterate based on a crude apptick interpretation
    while True:
```

```
        app.iterate()
        time.sleep(1.0/app.cfg['apptick'])
```

One notable feature of the above code is that it enables the use of configuration files, such as the `*.moos` files used by the similarly named middleware. Although the bindings have not yet been expanded to use the configuration parser provided in the MOOS code itself, the author of this thesis has provided a suitable substitute in the LAMSS codebase. When following the lab's setup guide to install the autonomy ecosystem, the necessary system paths are automatically configured to enable the imports as shown. Thus, a sample configuration file for the previous `SAMPLE_APP` could be given by:

```
// MOOS file : test_config.moos

ServerHost = localhost
ServerPort = 9000

MOOSTimeWarp = 1

ProcessConfig = ANTLER
{
  MSBetweenLaunches = 200
  Run = MOOSDB       @ NewConsole = false
  Run = sample_app     @ NewConsole = true
}

ProcessConfig = sample_app
{
    AppTick = 5
}
```

Assuming the source file's permissions have been updated to allow execution, the application could then be launched manually by calling:

```
./test_pymoos.py test_config.moos sample_app
```

The command arguments are (1) the application itself, (2) the configuration file (accessed by `args[1]` in the code), and (3) the application's name (`args[2]`), which is submitted to the MOOSDB when establishing a connection. This can be used to execute multiple instances of the same application with different aliases, as the MOOSDB requires that all clients register using unique names as identifiers. Alternatively, if the application is executable and accessible in the user's path under `sample_app` (as shown in the configuration file), then the app could be automatically launched with `pAntler` as part of a conventional MOOS mission.

A.2 The LAMSS Python plotting utilities

The LAMSS Display Center, designed as a solution for data visualization during mission runtime, is among the more involved uses of the Python-MOOS interface within the LAMSS codebase. At its heart, the LAMSS Display Center (or alternatively, lamssDC), sought to capture a series of legacy visualizations originally coded in MATLAB, and provide a more user-friendly, scalable solution. To better understand the driving forces behind this decision, it may be noted that the original code had the following limitations:

1. They required the MATLAB-MOOS interface `iMatlab`, which in turn required a MATLAB installation no newer than 2017b.

2. Each visualization tool required its own fully independent instance of MATLAB. As implemented, a series of benchmark tests revealed, this requirement implied a burden of approximately 1GB of memory per instance – a significant burden, when looking at scalability.

3. The original visualization scripts were highly opinionated with regards to display real estate, often taking control of the screen each time they updated their respective plots.

In order to tackle these issues, the Display Center was built in three parts: (1) a series of individual Python-MOOS applications charged with collecting the data for each of the sets of figures, (2) a website capable of ingesting updates via a websocket connection to a central server, and (3) a central server charged with accepting data from the Python-MOOS applications and relaying it as needed to the websocket clients. The capability of sending commands from the web interface, relayed through the central server to the appropriate destination, was also added – meaning that the central server application is both a websocket server and a Python-MOOS application. Figure A.1 illustrates this architecture schematically.

The website itself, mentioned earlier as the second component of the architecture, is a relatively simple site in terms of the underlying HTML and CSS (styling) code. Within the site, the websocket connection is managed with JavaScript code, as is the command and control system integrated therein. LAMSS mission control is typically handled via Goby Liaison, and the addition of similar capabilities to the LAMSS Display Center is not to supersede this convention. On the contrary, the intent with adding command and control capabilities in lamssDC was to facilitate access to configuration changes specifically related to the visualizations, minimizing the impact on more established components of the operational paradigm. An example of this is introducing the ability to change the transmitter and receiver nodes processed by the `tloss_display` tool from the same panel that displays the resulting transmission loss map.

Similarly, the site's plots are designed and rendered using BokehJS (a JavaScript library), as this approach creates clean yet interactive figures. An added benefit of this library is that the burden of re-rendering figures as needed is passed to the web client. Early development of the lamssDC setup used the Python flavor of the Bokeh library, preparing the figures server-side, but this meant the server was sending figure layout information in addition to the image or line plot data to each connected client, putting an additional burden on the local network. Furthermore, the clients still needed to render the figure, as it was received in the form of instructions to a related embedding

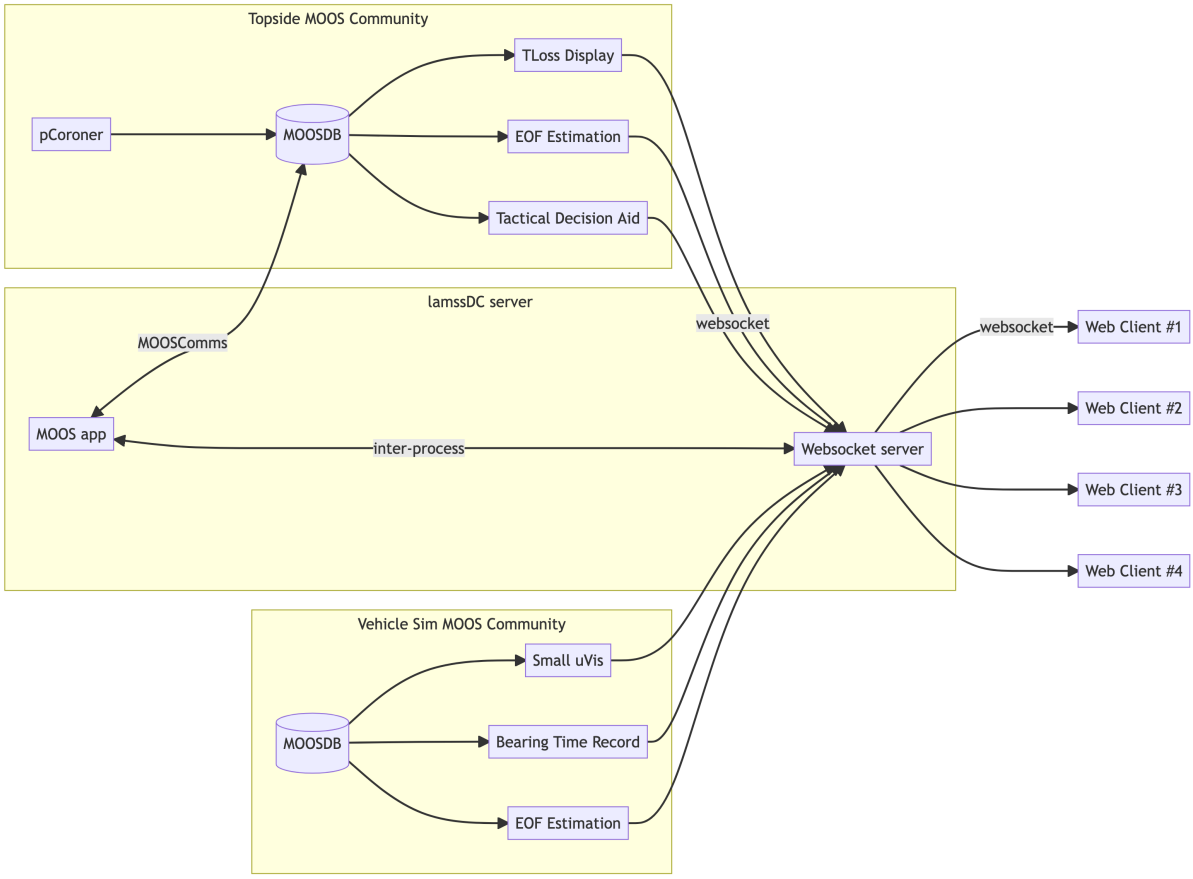


Figure A.1: Diagram illustrating the lamssDC server architecture. Plotting applications inherit from the uPlot superclass and send requisite data to the lamssDC server via a websocket connection.

engine rather than as static images, to preserve the benefit of interactivity. Updates in the original approach required that the client's web browser discard the previous figure entirely and then embed the new figure from scratch, whereas the current implementation takes advantage of more granular control granted by the native JavaScript API to minimize duplicate work when possible.

Though the web development aspect must be understood in order to properly extend the LAMSS Display Center's plot collection, the primary focus of this discussion is the use of the Python-MOOS interface as part of the data visualization framework. The next section discusses how the class-based architecture presented in Section A.1.4 was used for this purpose.

A.2.1 The uPlot superclass

As part of the initial translation effort, it became apparent that each of the modules used to replace the MATLAB visualizations had a common pattern – this was to be expected, given the common objective of rendering data collected from the MOOSDB. In order to reduce code duplication, and make the derivative projects robust to expected failure modes, most of the common patterns were transferred to a single module, called `uPlot`. This plot utility superclass provides a series of basic functions, including the following:

- `__init__()` (constructor):
  - Defines default values for required parameters, such as the MOOSDB server's host address and port number, if they haven't been defined in the subclass constructor already.
  - Provides configuration parameters for an optional replay mode, and for saving plots.
  - Reads the mission configuration file, if one has been defined.
  - Handles the basic `python-moos` configuration, connecting the callbacks for a new connection and new mail.
- `on_new_mail()`:

  - High-level callback, implements the `fetch()` call on the Python-MOOS interface, and passes individual mail entries to a dedicated handler.

- `on_connect()`:
  - High-level callback, implements the `register()` call on the Python-MOOS interface based on the configuration parameters.

- `iterate_looper()`:
  - High-level asynchronous loop, checks that the MOOSDB connection is alive before passing to a dedicated iterate handler that prepares the appropriate plot data sets.

- `websocket_client()`
  - Asynchronous loop, manages the websocket connection. This includes handling dropped connections, and recovering gracefully when the server becomes available again.

The high-level routines mentioned above ultimately connect to lower-level routines that are defined by default in the superclass, but are expected to be redefined in derivative classes. The expected implementation of a particular subclass, such as the `tloss_display` case metioned earlier, would be as follows:

- `__init__()` (constructor):
  1. Define application-specific defaults, including minimum configuration requirements and the application name to be passed to the MOOSDB in the absence of an appropriate override. Required variables should also be defined.
  2. Call the superclass initialization, to load additional generic defaults, and process the configuration file if one was provided.
  3. Define class-specific variables needed for the iteration logic, and process non-generic, keyword accessible configuration overrides.

- `handle_mail()`:
  - This is the basic mail-handling logic called by the corresponding superclass callback, where registered mail should be processed and loaded into the necessary variables.
- `handle_iterate()`:
  - This is the basic iteration logic for the plotting app, called by the corresponding superclass callback. This code should checks the mission state from the values of registered variables, to determine if data should be assembled into a plot-ready set. If appropriate, the resulting dictionary should be returned to the superclass method, that it may be relayed to the server's web clients.
- `__main__` loop:
  - The final block of each of the uPlot derivatives must determine that the app is being executed as a standalone program, and not being called as a library. When this is the case, the main loop must instantiate the asynchronous executions for the MOOS interface and the websocket client.

For additional information on the `uPlot` superclass and examples of how it is implemented to produce the plots shown in the LAMSS Display Center, readers are encouraged to look through the code provided in `lamss-shared/src/python/uPlot`.

B LAMSS Docker

The LAMSS autonomy software ecosystem generally requires a Linux Operating System to work properly, due to the extensive list of dependencies required. In particular, the LAMSS team generally supports LTS distributions of Ubuntu. By comparison, the MOOS-IvP project is primarily developed on machines running macOS, and is consistently tested for compatibility in Linux distributions as well. The historical solution undertaken by LAMSS members has been to configure virtual machines using hypervisor software such as VirtualBox or VMware; alternatively, many students have opted for operating computers with native Linux installations.

In 2019, members of the LAMSS team sought to revamp the documentation for the lab's software suite. The initial update was led by Dr. Michael Novitzky, Dr. EeShan Bhatt, and the author of this thesis. These documentation updates were then put to the test by a new class of students, who provided valuable feedback about the usefulness of the text and the software tools provided. Since then, continued efforts have been made to minimize the barriers to entry when it comes to deploying the LAMSS codebase. As part of these ongoing efforts, the author of this thesis has contributed automated configuration scripts and other solutions focused on one simple goal: to minimize the number and complexity of steps required to reach a deployment-ready state on any new machine – virtual or otherwise. This section is intended to serve as the initial documentation for the latest contributions of this kind, which is centered on the use of Docker containers.

B.1 From VMs to containers

The need for a containerized solution has two main roots. First, it has become clear over the past few years that the documentation efforts that began in 2019 have not been enough to make the LAMSS codebase accessible to the scientific community at large. Teams across the world who have sought to implement the Virtual Ocean simulator and other powerful tools of the LAMSS codebase in their own projects have needed to reach out to the lab for support – the author has been directly responsible for providing such support in numerous occasions. The common thread across many of these communications has been that although the documentation is generally clear in what is needed, it is specific artifacts of each individual system configuration – firewall restrictions, older installations and outdated packages – that generally create enough room for failure. Even when standard repositories may be reachable, some of the custom package repositories hosted by LAMSS members or affiliates can sometimes cause trouble. The vision of seamless portability lives at the heart of Docker, and similar containerization tools; their goal, in other words, is to address this need for a predictable, repeatable solution.

The second motivator for the move to containerization is tied to the present landscape of computational solutions. Some users of the LAMSS codebase, including the author, have transitioned to machines running on Apple Silicon. At the time of this writing, the landscape looks as follows: (1) the VirtualBox project does not support, nor does it intend to support, Apple Silicon natively; (2) VMware has released a public beta version of its macOS product, VMware Fusion, that provides native support on Apple Silicon – however, it is uncertain when the product will be officially released for general availability outside of the public tech preview program; and (3) Docker Desktop for Mac offers native support of the M1 chips, subject to the Docker Desktop terms. The latter means that larger enterprises with more than 250 employees OR more than $10 million USD in annual revenue now must obtain a paid subscription; personal use continues to be free.

B.2 The LAMSS Docker solution

The Dockerfile now provided with the LAMSS codebase is intended to serve as an initial solution to the challenges above. Although the image is not currently released to a public registry, it can be built locally using a convenience script provided with the codebase, or manually by calling `docker build`. To get started, a user must first collect the necessary repositories:

```
git clone https://github.com/GobySoft/goby3.git
git clone https://github.com/GobySoft/netsim.git

git clone git@github.mit.edu:lamss/lamss.git
git clone git@github.mit.edu:lamss/lamss-shared.git
#git clone git@github.mit.edu:lamss/lamss-internal.git
git clone git@github.mit.edu:lamss/missions-lamss.git
```

Note that `lamss-internal` is commented out in the previous snippet, as it is the lab's internal sandbox repository. Furthermore, the GitHub Enterprise server operated by MIT (`github.mit.edu`) requires Institute credentials and is not available to the general public. For the purpose of code distribution to approved collaborators, LAMSS has been using mirror servers. Teams interested in gaining access to the code should contact the lab for additional information.

After collecting the necessary repositories into a common directory, users are encouraged to run the following configuration script in their host machine (macOS, WSL, Linux):

```
cd lamss/scripts; ./config_lamss_bash
```

The `config_lamss_bash` script will scan the user's `.bashrc` file for entries related to historical configuration instructions, in order to avoid creating conflicts for experienced users. If the scan returns no conflicts, the program will then link the provided augmentation file to the current setup. This is recommended, but not required, as many of the adjustments relate to the compiler options

and the system's path. However, the augmented configuration does provide a number of convenient aliases and also sets the user path to include other tools in `lamss/scripts`; thus why it is recommended.

With the repositories now available on the host, and the configuration loaded, users can build the Docker image by calling:

```
lamss_docker build
```

This is a simple redirect to `docker build` with some preset options, provided mostly for convenience; the build can also be executed manually. The benefit of using the preset solution – or using the same tag, at a minimum – becomes clear when looking to run the container. The basic idea behind this Docker-based solution is that the user will have a reliable environment for code compilation and execution, but further development is performed from the host rather than the container. The repositories are linked to the container using the `--volume` argument, and the volumes are automatically identified and connected when using the following command:

```
lamss_docker run
```

For a typical setup, without the lab's sandbox repository, the above becomes equivalent to manually entering the following command:

```
docker run --rm -it \
    --name lamss \
    --publish 50022:22 \
    --publish 50080:80 \
    --publish 55900:5900 \
    --publish 50001:50001 \
    --publish 50002:50002 \
    --volume $LAMSS_HOME/goby3:/home/lamss/goby3 \
    --volume $LAMSS_HOME/netsim:/home/lamss/netsim \
    --volume $LAMSS_HOME/lamss:/home/lamss/lamss \
```

```
    --volume $LAMSS_HOME/lamss-shared:/home/lamss/lamss-shared \
    --volume $LAMSS_HOME/missions-lamss:/home/lamss/missions-lamss \
    lamss
```

When the `lamss-internal` repository is available on the host, the same command expands to provide the additional volume connection. The convenience of a short-hand command should be apparent.

The LAMSS Docker image is configured to behave as the lab's typical development environment. To that end, a number of ports are exposed to the host, as shown in Table B.1. All standard ports (SSH, HTTP, VNC) are adjusted by 50000, to match the range used by Goby Liaison.

Table B.1: Ports exposed in LAMSS Docker image.

Port	Application
22	SSH + X Forwarding
80	Web server (apache2)
5900	VNC server
50001	Goby Liaison
50002	lamssDC websocket server

As instructed by the Dockerfile, the image is configured to enable X Forwarding over SSH, and to disable password access. To gain access to the container via SSH, the appropriate keys must be copied over – assuming the image was built with the default arguments:

```
docker cp $HOME/.ssh/id_ed25519.pub \
    lamss:/home/lamss/.ssh/authorized_keys
```

One notable caveat to using X Forwarding to gain access to graphical applications is that some of the tools in the MOOS-IvP project, and by extension of the LAMSS ecosystem, do not behave well

with this type of connection. The main culprit to motivate an alternate solution was the `pMarineViewer` application, which triggered failures when testing over SSH with X Forwarding, with both native installations and docker containers hosting the application. To address this issue, the Dockerfile provided with LAMSS uses a VNC server as an alternate access point for graphical applications. This is achieved by using a threepart solution: first, a virtual framebuffer is configured using `xvfb` to handle the lack of a physical display; next, a lightweight window manager (`fluxbox`) is connected to the framebuffer; and third, the VNC server `x11vnc` is configured to present the contents of the virtual display. To connect to the VNC server, a user can employ their VNC client of choice (for example, the Screen Sharing app on macOS) by pointing it to `localhost:55900` on the host machine.

The conventional philosophy of containerized solutions is that a single container should handle a single process. To handle the limitations of this approach with respect to underlying services, such as the web and SSH servers, the image is configured to prompt the user about starting those services at first launch. A MySQL server, which is used by `geov`, is also installed by default and started upon user agreement. The password used to access the VNC connection is also requested at launch.

C Additional Python tools for LAMSS

In addition to the `python-moos` utilities presented in Appendix A, the LAMSS codebase has also been augmented with some of the more useful data processing tools developed as part of the work presented in this thesis. The tools, while not exhaustive in their coverage, are intended to facilitate the most common tasks required to assimilate mission logs recorded by the autonomy suite, or output files produced by the acoustic propagation models. This section introduces a number of these tools, generally replicating the embedded documentation. For more detailed coverage, readers are encouraged to explore the packages using Python's `help()` function. Alternatively, interested readers are encouraged to read through the referenced source code, which is extensively documented with the appropriate docstrings.

C.1 The BELLHOP utility library

The BELLHOP utility library, `lib_pybellhop`, was developed as a near direct translation of the corresponding MATLAB tools provided in the original Acoustics Toolbox. Not all files in the original codebase were translated, as not all were required for the work herein presented. In its present state, the following top-level functions are included:

- `read_env( ENVFIL , MODEL )`
  - Reads an environmental file (ENVFIL)

  - Returns dictionary with organized data

- `read_arrivals_asc( ARRFile )`

  - Reads a Bellhop-generated ARR file
  - Returns 3 separate data containers:

    1. Arr is a numpy array which contains the arrival structure, organized by coordinates.
    2. Pos is a dictionary with the coordinate values for the Arr[] indexing grid
    3. Freq is the frequency used by bellhop to generate this ARR file

- `read_ray( RAYFile )`

  - Reads a Bellhop-generated RAY file
  - Returns a list of dictionaries containing:

    1. `i_src` : the source index (in case there are multiple sources)
    2. `alpha0` : the launch angle of the ray
    3. `r` : the range coordinates produced by the ray tracing routine
    4. `z` : the depth coordinates produced by the ray tracing routine

- `read_shd( *args )`

  - Takes a Bellhop-generated SHD (shade) file and attempts to determine the correct reader function (ASCII or BIN) to call.
  - Returns a dictionary containing:

    1. `Pos` : the source and receiver positions
    2. `pressure` : the pressure recorded in the SHD file
    3. `freqVec` : the vector of frequencies

Additional functions provided in the Acoustic Toolbox, which the above top-level functions required as dependencies, were also translated. For additional information, see the Acoustic Toolbox documentation and the docstrings provided in this library.

C.2 The LAMSS utility library

Where the BELLHOP tools were translations of existing code, the LAMSS utility library, `lib_pylamss`, includes a more extensive collection of new code. The initial motivation for creating the library was to centralize frequently-used routines, particularly for processing of navigation and acoustic communications logs, into a widely reusable form. As python tools became more deeply integrated with the LAMSS paradigm, this library grew to include a superclass for plot utilities, and other convenience tools such as a mission configuration parser.

C.2.1 The Acoustic Comms tools

The modem interface used by LAMSS is built on Goby3, and two specialized applications were involved in managing this interface for ICEX-20. The apps, `lamss_icex_tracker` and `pAcommsHandler`, communicate with the modems via NMEA sentences transmitted over a serial interface, and multiple interfaces could be configured at any given time, such as for managing the Integrated Communications and Navigation Network. In order to facilitate the intake of acoustic communications logs, the following tools are provided:

- `read_umodem_nmea( line )`
  - Reads a full line from AComms logs (`lamss_icex_tracker`, `pAcommsHandler`), where the line payload is an NMEA message posted by `goby::acomms::modemdriver`
  - Returns a list with:
    1. timestamp, as epoch

2. interface mode (input/output)
3. modem number
4. NMEA string

- `read_acomms_logfile( acomms_file, get_diagnostics=False)`
  - Reads a full AComms log file (`lamss_icex_tracker`, `pAcommsHandler`), and calls `read_umodem_nmea()` for each line that contains a micromodem NMEA sentence.
  - Returns a dictionary with numpy arrays:
    1. `time` : timestamp, as epoch
    2. `mode` : interface mode (input/output)
    3. `thr_id` : thread number, for umodem driver
    4. `nmea` : NMEA string
    5. `config` : umodem config (SRC, BND, IRE)
    6. `diagnostics` : sub-dictionary, optional
  - `get_diagnostics=True` : If get_diagnostics=True, a sub-dictionary is to the return under the `diagnostics` key, containing the above elements 1-4 for NMEA sentences used in the goby micromodem driver for diagnostics and drift correction. These include: `**CFG`,`**CFQ`, `**TMS`,`**TMQ`,`CAREV`

- `sort_comms( comms_data )`
  - Reads comms data produced by read_acomms_logfile(), identifies source name (per ICEX 2020 modem id table) and sorts comms data by source name.
  - Returns dictionary with data sorted into:
    * h1-h4 : tracking range buoys
    * macrura_10k
    * macrura_25k

        * camp_10k
        * camp_25k
    - **NOTE**: This utility was written to process the ICEX 2020 data. It does *NOT* currently accept a reference modemidlookup.txt file

- `read_caire_line( line )`
    - Reads CAIRE line (NMEA sentence) and converts hex-character representation of 16-bit integers to appropriate numerical form.
    - CAIRE line format: `$CAIRE,N,<data>*<checksum>`
    - Returns tuple with (N,`<data>`), where N is the index from the NMEA sentence

- `read_cacst_line( line )`
    - Reads CACST line (NMEA sentence) and converts to dictionary representation.
    - CACST line format: `$CACST,<Field1>,<Field2>,...,<Field26>*<checksum>`
    - Returns dictionary with fields as defined in the WHOI Micromodem User Manual.

C.2.2 The navigation and general log tools

The navigation and general log tools are intended to provide easy access to Python-powered analysis of mission logs.

- `read_klog_num( klog_dir, var_name )`
    - Reads a `*.klog` file generated by alogsplit, in which the message payload consists of numerical values only.
    - Returns a numpy array containing the log time and numerical value for each line in `*.klog`
    - Example source file: `NAV_X.klog`

- `read_klog_str( klog_dir, var_name )`
  - Reads a `*.klog` file generated by alogsplit, in which the message payload consists of string values.
  - Returns a dictionary containing numpy arrays with the data from each line in `*.klog`:
    1. `time` : the log time
    2. `data` : the string value
  - Example source file: `IVPHELM_BHV_RUNNING.klog`
- `read_lamssPB_to_dict( line )`
  - Reads a full line from LAMSS & MOOS logs, where the line's payload is a serialized protobuf message identified by a `@PB[<message-name>]` tag. The substring that follows the `@PB` tag is refactored into a JSON-like string and unpacked by `json.loads()`
  - Returns a dictionary representation of the protobuf payload, as captured by `json.loads()`
  - Example: `NAV_DATA` logs posted by `pLamssMissionManager`
- `read_node_report_to_dict( line, is_log=True )`
  - Reads a NODE_REPORT line:
    1. is_log=True (default): Reads a full line from LAMSS & MOOS logs, where the line's payload is a string matching the `NODE_REPORT` format. The string is converted to a dictionary, which is then refactored to match the key-value pairs from: `@PB[goby.moos.protobuf.NodeStatus]` where nested dictionaries (`global_fix`, `local_fix`) are unpacked into the top layer, to match the output of: `read_nav_data()`
    2. is_log=False: Reads a standalone `NODE_REPORT`, such as when called from a pymoos instance
  - Returns dictionary with `{key : single-value}` pairs

  - **NOTE**: Time-series data can be reassembled from calling process by converting list of dictionary entries (one per line processed) into dictionary of lists or dictionary of arrays:

```
Y = {key: numpy.array([it[key] for it in X]) for key in X[0]}
```

- `read_nav_data( klog_dir, nav_str="NAV_", debug=False )`
  - Reads the navigation data available in the `*.klog` directory generated by alogsplit, by default: `<alog_name>_alvtmp/`
  - Prioritizes `NAV_DATA.klog` and `NODE_REPORT.klog` if available. Otherwise, processes all `NAV_*` files in the klog directory.
  - Prefix `NAV_` can be modified using `nav_str=<prefix>`, which will search for the following first:
    - `<prefix>_DATA.klog`
  - Alternatively, the prefix override will instruct the program to consider single-number variables such as:
    - `<prefix>_X.klog`
    - `<prefix>_Y.klog`
    - `<prefix>_DEPTH.klog`
  - Returns a dictionary formatted as a flattened version of `@PB[goby.moos.protobuf.NodeStatus]`; nested dictionaries (`global_fix`, `local_fix`) are unpacked into the top layer.
  - Use debug=True to enable debug print statements
  - **NOTE**: `<prefix>*.klog` files are filtered with `pandas.DataFrame.join()` using the `inner` method, removing rows with NaN entries

Acronyms

AI	Artificial Intelligence
AUV	Autonomous Underwater Vehicle
DL	Deep Learning
ICNN	Integrated Communication and Navigation Network
ML	Machine Learning
NRL	Naval Research Laboratory
PCA	Principal component analysis
PINN	Physically Informed Neural Network
PSK	Phase-Shift Keying modulation
TDA	Tactical decision aid
TDMA	Time Division, Multiple Access
VOAT	Virtual Ocean Autonomy Testbed

Glossary

CACST	WHOI Micromodem's communication cycle receive statistics
CAIRE	WHOI Micromodem's impulse response estimate, given as the deconvolution of a known signal replica against a limited observation of the acoustic sensor data. A match exceeding a predetermined energy threshold is regarded as a detection and activates the modem's receiver.
CAXST	WHOI Micromodem's communication cycle transmit statistics
GLONASS	The Russian Global Navigation Satellite System, an alternative to the US GPS
GNSS	Global Navigation Satellite System
GPS	The US Global Positioning System
MOOS	The Mission Oriented Operating Suite, a lightweight middleware for autonomous vehicles
SOFAR	The Sound Fixing and Ranging channel

Bibliography

1. E. R. Anderson. *Sound speed in seawater as a function of realistic temperature – salinity – pressure domains*. Technical report TP 243. Ocean Sciences Department, Naval Undersea Research and Development Center, 1971.

2. H. Beech and M. Haag. "10 Missing After U.S. Navy Ship and Oil Tanker Collide Off Singapore". *The New York Times*, 2017.

3. M. R. Benjamin, H. Schmidt, P. M. Newman, and J. J. Leonard. "Nested autonomy for unmanned marine vehicles with MOOS-IvP". *Journal of Field Robotics* 27:6, 2010, pp. 834–875. DOI: https://doi.org/10.1002/rob.20370.

4. E. Bhatt, O. Viquez, and H. Schmidt. "Under-ice acoustic navigation using real-time model-aided range estimation". *The Journal of the Acoustical Society of America*, 2021. Submitted August 2021.

5. E. C. Bhatt, O. A. Viquez, M. Novitzky, and H. Schmidt. "An information theory approach to assess acoustic-environmental significance". *The Journal of the Acoustical Society of America* 146:4, 2019, pp. 2965–2965. DOI: 10.1121/1.5137309.

6. E. C. Bhatt. "A Virtual Ocean framework for environmentally adaptive, embedded acoustic navigation on autonomous underwater vehicles". PhD thesis. Massachusetts Institute of Technology, 2021.

7. E. C. Bhatt, B. A. Howard, and H. Schmidt. "Embedded Tactical Decision Aid Framework for Environmentally Adaptive Autonomous Underwater Vehicle Communication and Navigation". *Journal of Ocean Engineering*, 2021. Accepted for publication.

8. M. Bianco and P. Gerstoft. "Compressive acoustic sound speed profile estimation". *The Journal of the Acoustical Society of America* 139:3, 2016, EL90–EL94. DOI: 10.1121/1.4943784.

9. M. Bianco and P. Gerstoft. "Dictionary learning of sound speed profiles". *The Journal of the Acoustical Society of America* 141:3, 2017, pp. 1749–1758. DOI: 10.1121/1.4977926.

10. M. J. Bianco and P. Gerstoft. "Travel Time Tomography With Adaptive Dictionaries". *IEEE Transactions on Computational Imaging* 4:4, 2018, pp. 499–511. DOI: 10.1109/TCI.2018.2862644.

11. M. J. Bianco, P. Gerstoft, J. Traer, E. Ozanich, M. A. Roch, S. Gannot, and C.-A. Deledalle. "Machine learning in acoustics: Theory and applications". *The Journal of the Acoustical Society of America* 146:5, 2019, pp. 3590–3628. DOI: 10.1121/1.5133944.

12. A. Birhane and V. U. Prabhu. "Large image datasets: A pyrrhic win for computer vision?" In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2021, pp. 1536–1546. ISBN: 9781665404778.

13. C. M. Bishop. *Pattern recognition and machine learning*. Information science and statistics. Springer, New York, 2006. ISBN: 0387310738.

14. D. T. Blackstock. *Fundamentals of physical acoustics*. Wiley, New York, 2000. ISBN: 0471319791.

15. E. P. Chassignet, H. E. Hurlburt, O. M. Smedstad, G. R. Halliwell, P. J. Hogan, A. J. Wallcraft, R. Baraille, and R. Bleck. "The HYCOM (HYbrid Coordinate Ocean Model) data assimilative system". *Journal of Marine Systems* 65:1, 2007. Marine Environmental Monitoring and Prediction, pp. 60–83. ISSN: 0924-7963. DOI: https://doi.org/10.1016/j.jmarsys.

2005.09.016. URL: https://www.sciencedirect.com/science/article/pii/S0924796306002855.

16. C.-T. Chen and F. J. Millero. "Speed of sound in seawater at high pressures". *The Journal of the Acoustical Society of America* 62:5, 1977, pp. 1129–1135. DOI: 10.1121/1.381646.

17. R. Chen. "Ambient Acoustics as Indicator of Environmental Change in the Beaufort Sea: Experiments & Methods for Analysis". PhD thesis. Massachusetts Institute of Technology, 2021.

18. R. Chen and H. Schmidt. "Robustness Analysis of a Convolutional Neural Network Approach to Source-Range Estimation in a Simulated Arctic Environment". In: *OCEANS 2019 MTS/IEEE SEATTLE*. 2019, pp. 1–6. DOI: 10.23919/OCEANS40490.2019.8962829.

19. H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML. *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. 2018.

20. H. A. Dau, D. F. Silva, F. Petitjean, G. Forestier, A. J. Bagnall, A. A. Mueen, and E. J. Keogh. "Optimizing dynamic time warping's window width for time series data mining applications". *Data Mining and Knowledge Discovery* 32, 2018, pp. 1074–1120.

21. M. Deffenbaugh, J. Bellingham, and H. Schmidt. "The relationship between spherical and hyperbolic positioning". In: *OCEANS 96 MTS/IEEE Conference Proceedings. The Coastal Ocean - Prospects for the 21st Century*. Vol. 2. 1996, 590–595 vol.2. DOI: 10.1109/OCEANS.1996.568293.

22. M. Deffenbaugh, H. Schmidt, and J. Bellingham. "Acoustic positioning in a fading multipath environment". In: *OCEANS 96 MTS/IEEE Conference Proceedings. The Coastal Ocean - Prospects for the 21st Century*. Vol. 2. 1996, 596–600 vol.2. DOI: 10.1109/OCEANS.1996.568294.

23. M. Deffenbaugh. “Optimal Ocean Acoustic Tomography and Navigation with Moving Sources”. PhD thesis. Massachusetts Institute of Technology, 1997.

24. V. A. Del Grosso. “New equation for the speed of sound in natural waters (with comparisons to other equations)”. *The Journal of the Acoustical Society of America* 56:4, 1974, pp. 1084–1091. DOI: 10.1121/1.1903388.

25. V. A. Del Grosso. “Tables of the speed of sound in open ocean water (with Mediterranean Sea and Red Sea applicability)”. *The Journal of the Acoustical Society of America* 53:5, 1973, pp. 1384–1401. DOI: 10.1121/1.1913484.

26. V. A. Del Grosso and C. W. Mader. “Speed of Sound in Sea-Water Samples”. *The Journal of the Acoustical Society of America* 52:3B, 1972, pp. 961–974. DOI: 10.1121/1.1913202.

27. T. F. Duda. “Modeling and Forecasting Ocean Acoustic Conditions”. *Journal of Marine Research* 75:3, 2017, pp. 435–457. DOI: doi:10.1357/002224017821836734.

28. T. F. Duda, W. G. Zhang, and Y.-T. Lin. “Effects of Pacific Summer Water layer variations and ice cover on Beaufort Sea underwater sound ducting”. *The Journal of the Acoustical Society of America* 149:4, 2021, pp. 2117–2136. DOI: 10.1121/10.0003929.

29. B. D. Dushaw, P. F. Worcester, B. D. Cornuelle, and B. M. Howe. “On equations for the speed of sound in seawater”. *The Journal of the Acoustical Society of America* 93:1, 1993, pp. 255–275. DOI: 10.1121/1.405660.

30. P. Elisseeff. “Fast acoustic tomography of coastal, tidally-driven temperature and current fields”. PhD thesis. Massachusetts Institute of Technology, 1991.

31. P. Elisseeff, H. Schmidt, and W. Xu. “Ocean Acoustic Tomography as a Data Assimilation Problem”. *IEEE Journal of Oceanic Engineering* 27:2, 2002, pp. 275–282. DOI: 10.1109/JOE.2002.1002482.

32. E. M. Fischell, O. Viquez, and H. Schmidt. "Passive acoustic tracking for behavior mode classification between surface and underwater vehicles". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 2383–2388. DOI: 10.1109/IROS.2018.8593981.

33. E. M. Fischell, O. A. Viquez, and H. Schmidt. "Machine learning for behavior classification of passively tracked vessels". *The Journal of the Acoustical Society of America* 144:3, 2018, pp. 1685–1685. DOI: 10.1121/1.5067491.

34. N. P. Fofonoff and R. C. Millard. *Algorithms for computation of fundamental properties of seawater*. UNESCO Technical Papers in Marine Science, No. 44. Place de Fontenoy, 75700 Paris.: Division of Marine Sciences, UNESCO, 1983.

35. J. W. Fouquette. "Multipath Arrival Tracking for Marine Vehicles Utilizing Pattern Recognition". MA thesis. Massachusetts Institute of Technology, 2018.

36. A. T. Gardner and J. A. Collins. "A second look at Chip Scale Atomic Clocks for long term precision timing". In: *OCEANS 2016 MTS/IEEE Monterey*. 2016, pp. 1–9. DOI: 10.1109/OCEANS.2016.7761268.

37. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org.

38. D. W. Goodwin. "Environmental Effects of the Beaufort Lens on Underwater Acoustic Communications during Arctic Operations". MA thesis. Massachusetts Institute of Technology, 2021.

39. M. Greenspan and C. Tschiegg. "Speed of sound in water by a direct method". *Journal of Research of the National Bureau of Standards* 59:4, 1957, p. 249. DOI: 10.6028/jres.059.028.

40. M. Greenspan and C. E. Tschiegg. "Tables of the Speed of Sound in Water". *The Journal of the Acoustical Society of America* 31:1, 1959, pp. 75–76. DOI: 10.1121/1.1907614.

41. T. Griggs and D. Wakabayashi. "How a Self-Driving Uber Killed a Pedestrian in Arizona". *The New York Times*, 2018.

42. E. Haghighat and R. Juanes. "SciANN: A Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks". *Computer Methods in Applied Mechanics and Engineering* 373, 2021, p. 113552. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2020.113552.

43. P. J. Haley and P. F. J. Lermusiaux. "Multiscale two-way embedding schemes for free-surface primitive equations in the "Multidisciplinary Simulation, Estimation and Assimilation System"". *Ocean Dynamics* 60:6, 2010, pp. 1497–1537. DOI: 10.1007/s10236-010-0349-4. URL: https://doi.org/10.1007/s10236-010-0349-4.

44. D. Halpern. "1. Oceanography before, and after, the Advent of Satellites". In: *Satellites, Oceanography and Society*. Elsevier, 2000. ISBN: 978-0-44-450501-9. URL: https://app.knovel.com/hotlink/khtml/id:kt004QBFDA/satellites-oceanography/oceanography-before-after.

45. K. Harper, L. W. Uccellini, E. Kalnay, K. Carey, and L. Morone. "50th Anniversary of Operational Numerical Weather Prediction". *Bulletin of the American Meteorological Society* 88:5, 2007, pp. 639–650. DOI: 10.1175/BAMS-88-5-639. URL: https://journals.ametsoc.org/view/journals/bams/88/5/bams-88-5-639.xml.

46. G. Hope and H. Schmidt. "A parallelization of the wavenumber integration acoustic modelling package OASES". *The Journal of the Acoustical Society of America* 144:3, 2018, pp. 1819–1819. DOI: 10.1121/1.5068023.

47. B. A. Howard. "Multipath Penalty Metric in Underwater Acoustic Communication for Autonomy and Human Decision-making". MA thesis. Massachusetts Institute of Technology, 2021.

48. B. M. Howe, J. Miksis-Olds, E. Rehm, H. Sagen, P. F. Worcester, and G. Haralabus. "Observing the Oceans Acoustically". *Frontiers in Marine Science* 6, 2019, p. 426. ISSN: 2296-7745. DOI: 10.3389/fmars.2019.00426. URL: https://www.frontiersin.org/article/10.3389/fmars.2019.00426.

49. A. S. Huang, E. Olson, and D. C. Moore. "LCM: Lightweight Communications and Marshalling". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 4057–4062. DOI: 10.1109/IROS.2010.5649358.

50. F. B. Jensen, W. A. Kuperman, M. B. Porter, and H. Schmidt. *Computational ocean acoustics*. 2nd ed. Modern acoustics and signal processing. Springer, New York, 2011. ISBN: 9781441986771.

51. L. Kedward, B. Aradi, O. Certik, M. Curcic, S. Ehlert, P. Engel, R. Goswami, M. Hirsch, A. Lozada-Blanco, V. Magnin, A. Markus, E. Pagone, I. Pribec, B. Richardson, H. Snyder, J. Urban, and J. Vandenplas. "The State of Fortran". *Computing in Science & Engineering*, 2022. DOI: 10.1109/MCSE.2022.3159862.

52. D. A. Kolb. *The experiential learning theory of career development*. eng. M.I.T. Alfred P. Sloan School of Management. Working paper ; no. 705-74. M.I.T., Cambridge, 1974.

53. R. Krishfield, J. Toole, A. Proshutinsky, and M.-L. Timmermans. "Automated Ice-Tethered Profilers for Seawater Observations under Pack Ice in All Seasons". *Journal of Atmospheric and Oceanic Technology* 25:11, 2008, pp. 2091–2105. DOI: 10.1175/2008JTECHO587.1. URL: https://journals.ametsoc.org/view/journals/atot/25/11/2008jtecho587_1.xml.

54. M. Lasky. "Review of undersea acoustics to 1950". *The Journal of the Acoustical Society of America* 61:2, 1977, pp. 283–297. DOI: 10.1121/1.381321.

55. M. Lazzarin. "Parallel implementation of a ray tracer for underwater sound waves using the cuda li- braries: description and application to the simulation of underwater networks". MA thesis. University of Padova, 2012.

56. P. K. LeHardy and C. Moore. "Deep ocean search for Malaysia airlines flight 370". In: *2014 Oceans - St. John's*. 2014, pp. 1–4. DOI: 10.1109/OCEANS.2014.7003292.

57. K. V. Mackenzie. "Nine-term equation for sound speed in the oceans". *The Journal of the Acoustical Society of America* 70:3, 1981, pp. 807–812. DOI: 10.1121/1.386920.

58. H. Medwin. "Speed of sound in water: A simple equation for realistic parameters". *The Journal of the Acoustical Society of America* 58:6, 1975, pp. 1318–1319. DOI: 10.1121/1.380790.

59. F. Monaldo, C. Jackson, and W. Pichel. "SEASAT TO RADARSAT-2: RESEARCH TO OPERATIONS". *Oceanography (Washington, D.C.)* 26:2, 2013, pp. 34–45. ISSN: 1042-8275.

60. National Transportation Safety Board. *Mid-Air Collision: Ongoing Investigation, Accident No. CEN21FA215*. Retrieved Oct 11, 2021. 2021. URL: https://www.ntsb.gov/investigations/Pages/CEN21FA215.aspx.

61. National Transportation Safety Board. *Midair Collision over George Inlet de Havilland DHC-2, N952DB, and de Havilland DHC-3, N959PA*. Retrieved Oct 11, 2021. 2019. URL: https://www.ntsb.gov/investigations/Pages/CEN19MA141AB.aspx.

62. Natural Resources Conservation Service. *Web Soil Survey*. Last checked Nov 2021. US Department of Agriculture. URL: https://websoilsurvey.sc.egov.usda.gov/App/WebSoilSurvey.aspx.

63. P. M. Newman. "MOOS-mission orientated operating suite", 2008.

64. C. G. Northcutt, A. Athalye, and J. Mueller. *Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks*. 2021. arXiv: 2103.14749 [stat.ML].

65. G. P. P. Pun, R. Batra, R. Ramprasad, and Y. Mishin. "Physically informed artificial neural networks for atomistic modeling of materials". *Nature Communications* 10:1, 2019, p. 2339. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10343-5.

66. M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. "ROS: an open-source Robot Operating System".

67. M. Raissi, P. Perdikaris, and G. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". *Journal of Computational Physics* 378, 2019, pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045.

68. S. Randeni, T. Schneider, E. Bhatt, O. A. Viquez R., and H. Schmidt. "A high-resolution AUV navigation framework with integrated communication and tracking for under-ice deployments". *Field Robotics*, 2021. Submitted, under review.

69. S. Randeni, T. Schneider, and H. Schmidt. "Construction of a high-resolution under-ice AUV navigation framework using a multidisciplinary virtual environment". In: *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*. 2020, pp. 1–7. DOI: 10.1109/AUV50043.2020.9267950.

70. S. A. T. Randeni P., N. R. Rypkema, E. M. Fischell, A. L. Forrest, M. R. Benjamin, and H. Schmidt. "Implementation of a Hydrodynamic Model-Based Navigation System for a Low-Cost AUV Fleet". In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 2018, pp. 1–6. DOI: 10.1109/AUV.2018.8729758.

71. M. Rich. "7 Navy Sailors Missing AFter U.S. Destroyer Collides With Merchant Vessel Off Japan". *The New York Times*, 2017.

72. P. Robinette, M. Novitzky, C. Fitzgerald, M. R. Benjamin, and H. Schmidt. "Exploring Human-Robot Trust During Teaming in a Real-World Testbed". In: *2019 14th ACM/IEEE International*

Conference on Human-Robot Interaction (HRI). 2019, pp. 592–593. DOI: 10.1109/HRI.2019.8673134.

73. N. R. Rypkema, E. M. Fischel, and H. Schmidt. "Closed-Loop Single-Beacon Passive Acoustic Navigation for Low-Cost Autonomous Underwater Vehicles". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 641–648. DOI: 10.1109/IROS.2018.8593626.

74. N. R. Rypkema. "Distributed Autonomy and Formation Control of a Drifting Swarm of Autonomous Underwater Vehicles". MA thesis. Massachusetts Institute of Technology, 2015.

75. N. R. Rypkema. "Underwater & Out of Sight: Towards Ubiquity in Underwater Robotics". PhD thesis. Massachusetts Institute of Technology, 2019.

76. H. Schmidt and M. Benjamin. *System and Method for Collision Avoidance in Underwater Vehicles*. U.S. Patent 8,830,793. 2014.

77. H. Schmidt and T. Schneider. "Acoustic communication and navigation in the new Arctic — A model case for environmental adaptation". In: *2016 IEEE Third Underwater Communications and Networking Conference (UComms)*. 2016, pp. 1–4. DOI: 10.1109/UComms.2016.7583469.

78. T. Schneider. "Goby3: A new open-source middleware for nested communication on autonomous marine vehicles". In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. 2016, pp. 236–240. DOI: 10.1109/AUV.2016.7778677.

79. T. Schneider. "Transmitting Internet Protocol packets efficiently on underwater networks using entropy-encoder header translation". In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. 2016, pp. 241–245. DOI: 10.1109/AUV.2016.7778678.

80. T. Schneider, S. Petillo, H. Schmidt, and C. Murphy. "The Dynamic Compact Control Language version 3". In: *OCEANS 2015 - Genova*. 2015, pp. 1–7. DOI: 10.1109/OCEANS-Genova.2015.7271608.

81. T. Schneider and H. Schmidt. "NETSIM: A Realtime Virtual Ocean Hardware-in-the-loop Acoustic Modem Network Simulator". In: *2018 Fourth Underwater Communications and Networking Conference (UComms)*. 2018, pp. 1–5. DOI: 10.1109/UComms.2018.8493188.

82. T. Schneider, H. Schmidt, and S. Randeni. "Self-Adapting Under-Ice Integrated Communications and Navigation Network". In: *2021 Fifth Underwater Communications and Networking Conference (UComms)*. 2021, pp. 1–5. DOI: 10.1109/UComms50339.2021.9598012.

83. B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2018. ISBN: 9780262256933. DOI: 10.7551/mitpress/4175.001.0001. URL: https://doi.org/10.7551/mitpress/4175.001.0001.

84. K. Schulz. "Literature's Arctic Obsession". *The New Yorker*, 2017.

85. Sea-Bird Electronics. *Application Note 6: Determination of Sound Velocity from CTD Data*. Last checked July 2021. 2004. URL: https://www.seabird.com/asset-get.download.jsa?code=250965.

86. Sea-Bird Electronics. *SBE 37-SI MicroCAT C-T (P) Recorder (User Manual)*. Last checked August 2021. 2020. URL: https://www.seabird.com/asset-get.download.jsa?id=54627862347.

87. Sea-Bird Electronics. *Seasoft V2: SBE Data Processing (Software Manual)*. Last checked August 2021. 2017. URL: https://www.seabird.com/asset-get.download.jsa?id=54627862745.

88. A. Singhvi and K. Russell. "Inside the Self-Driving Tesla Fatal Accident". *The New York Times*, 2016.

89. J. D. Smith, K. Azizzadenesheli, and Z. E. Ross. *EikoNet: Solving the Eikonal equation with Deep Neural Networks*. 2020. arXiv: 2004.00361 [physics.comp-ph].

90. S. Sontag. *Blind man's bluff : the untold story of American submarine espionage*. eng. Public Affairs, New York, 1998. ISBN: 1891620088.

91. T. A. Stanley, D. B. Kirschbaum, G. Benz, R. A. Emberson, P. M. Amatya, W. Medwedeff, and M. K. Clark. "Data-Driven Landslide Nowcasting at the Global Scale". *Frontiers in Earth Science* 9, 2021. ISSN: 2296-6463. DOI: 10.3389/feart.2021.640043. URL: https://www.frontiersin.org/article/10.3389/feart.2021.640043.

92. J. M. Toole, R. A. Krishfield, M.-L. Timmermans, and A. Proshutinsky. "The Ice-Tethered Profiler: Argo of the Arctic". *Oceanography*, 2011. DOI: 10.5670/oceanog.2011.64.

93. M. Tschudi, W. N. Meier, J. S. Stewart, C. Fowler, and J. Maslanik. *EASE-Grid Sea Ice Age, Version 4*. Last checked September 2021. NASA National Snow and Ice Data Center Distributed Active Archive Center. Boulder, Colorado USA, 2019. DOI: https://doi.org/10.5067/UTAV7490FEPB.

94. M. Tschudi, W. N. Meier, J. S. Stewart, C. Fowler, and J. Maslanik. *Polar Pathfinder Daily 25 km EASE-Grid Sea Ice Motion Vectors, Version 4*. Last checked September 2021. NASA National Snow and Ice Data Center Distributed Active Archive Center. Boulder, Colorado USA, 2019. DOI: https://doi.org/10.5067/INAWUW07QH7B.

95. B. W. Tuchman. *The Guns of August*. Kindle Edition. Barbara W. Tuchman's Great War Series. Random House Publishing Group, New York, 2014.

96. R. J. Urick. *Sound propagation in the sea*. Dept. of Defense, Office of the Secretary of Defense, Defense Advanced Research Projects Agency, Arlington, Va, 1979.

97. D. F. Van Komen, T. B. Neilsen, K. Howarth, D. P. Knobles, and P. H. Dahl. "Seabed and range estimation of impulsive time series using a convolutional neural network". *The Journal of the Acoustical Society of America* 147:5, 2020, EL403–EL408. DOI: 10.1121/10.0001216.

98. H. L. Van Trees. *Optimum Array Processing*. Part IV of Detection, Estimation and Modulation Theory. Wiley-Interscience, New York, 2002. ISBN: 0-471-09390-4.

99. L. J. Van Uffelen. "Global Positioning Systems: Over Land and Under Sea". *Acoustics Today* 17:1, 2021, pp. 52–60.

100. O. A. Viquez, E. C. Bhatt, M. Novitzky, and H. Schmidt. "Model-aided acoustic environment estimation via data fusion for autonomous underwater vehicles". *The Journal of the Acoustical Society of America* 146:4, 2019, pp. 2965–2965. DOI: 10.1121/1.5137307.

101. O. A. Viquez, E. M. Fischell, N. R. Rypkema, and H. Schmidt. "Design of a general autonomy payload for low-cost AUV R&D". In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. 2016, pp. 151–155. DOI: 10.1109/AUV.2016.7778663.

102. O. A. Viquez, E. M. Fischell, and H. Schmidt. "Estimation of the acoustic environment through machine learning techniques". *The Journal of the Acoustical Society of America* 144:3, 2018, pp. 1743–1743. DOI: 10.1121/1.5067730.

103. U. bin Waheed, T. Alkhalifah, E. Haghighat, C. Song, and J. Virieux. *PINNtomo: Seismic tomography using physics-informed neural networks*. 2021. arXiv: 2104.01588 [physics.comp-ph].

104. U. b. Waheed, E. Haghighat, T. Alkhalifah, C. Song, and Q. Hao. "Eikonal solution using physics-informed neural networks". *arXiv preprint arXiv:2007.08330*, 2020.

105. W. D. Wilson. "Equation for the Speed of Sound in Sea Water". *The Journal of the Acoustical Society of America* 32:10, 1960, pp. 1357–1357. DOI: 10.1121/1.1907913.

106. W. D. Wilson. “Speed of Sound in Distilled Water as a Function of Temperature and Pressure”. *The Journal of the Acoustical Society of America* 31:8, 1959, pp. 1067–1072. DOI: 10.1121/1.1907828.

107. W. D. Wilson. “Speed of Sound in Sea Water as a Function of Temperature, Pressure, and Salinity”. *The Journal of the Acoustical Society of America* 32:6, 1960, pp. 641–644. DOI: 10.1121/1.1908167.

108. C. Wunsch. “Right Place, Right Time: An Informal Memoir”. *Annual Review of Marine Science* 13:1, 2021, pp. 1–21. DOI: 10.1146/annurev-marine-021320-125821.

109. C. Zimba, M. Sacarny, M. Yoder, and B. Bray. *Chart of the Lower Charles River*. MIT Sea Grant College Program and Charles River Alliance of Boaters. 2017.