Underwater & Out of Sight: Towards Ubiquity in Underwater Robotics

by

Nicholas Rahardiyan Rypkema

Submitted to the Department of Electrical Engineering and Computer Science
and to the Joint Program in Applied Ocean Science and Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

and the

WOODS HOLE OCEANOGRAPHIC INSTITUTION

September 2019

Author .....
Joint Program in Applied Ocean Science and Engineering
August 30, 2019

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

Accepted by .....
David Ralston
Associate Scientist with Tenure, Applied Ocean Physics & Engineering Department
Chair, Joint Committee for Applied Ocean Science and Engineering

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Underwater & Out of Sight:
Towards Ubiquity in Underwater Robotics

by

Nicholas Rahardiyan Rypkema

Submitted to the Department of Electrical Engineering and Computer Science
and to the Joint Program in Applied Ocean Science and Engineering
on August 30, 2019
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Abstract

The Earth's oceans holds a wealth of information currently hidden from us. Effective measurement of its properties could provide a better understanding of our changing climate and insights into the creatures that inhabit its waters. Autonomous underwater vehicles (AUVs) hold the promise of penetrating the ocean environment and uncovering its mysteries; and progress in underwater robotics research over the past three decades has resulted in vehicles that can navigate reliably and operate consistently, providing oceanographers with an additional tool for studying the ocean.

Unfortunately, the high cost of these vehicles has stifled the democratization of this technology. We believe that this is a consequence of two factors. Firstly, reliable navigation on conventional AUVs has been achieved through the use of a sophisticated sensor system, namely the Doppler velocity log (DVL)-aided inertial navigation system (INS), which drives up vehicle cost, power use and size. Secondly, deployment of these vehicles is expensive and unwieldy due to their complexity, size and cost, resulting in the need for specialized personnel for vehicle operation and maintenance.

The recent development of simpler, low-cost, miniature underwater robots provides a solution that mitigates both these factors; however, removing the expensive DVL-aided INS means that they perform poorly in terms of navigation accuracy. We address this by introducing a novel acoustic system that enables AUV self-localization without requiring a DVL-aided INS or on-board active acoustic transmitters. We term this approach Passive Inverted Ultra-Short Baseline (piUSBL) positioning. The system uses a single acoustic beacon and a time-synchronized, vehicle-mounted, passive receiver array to localize the vehicle relative to this beacon. Our approach has two unique advantages: first, a single beacon lowers cost and enables easy deployment; second, a passive receiver allows the vehicle to be low-power, low-cost and small, and enables multi-vehicle scalability.

Providing this new generation of small and inexpensive vehicles with accurate navigation can potentially lower the cost of entry into underwater robotics research and further its widespread use for ocean science. We hope that these contributions in low-cost underwater navigation will enable the ubiquitous and coordinated use of robots to explore and understand the underwater domain.

Thesis Supervisor: Henrik Schmidt
Title: Professor of Mechanical and Ocean Engineering, Massachusetts Institute of Technology

Acknowledgements

This thesis is the result of six wonderful and fulfilling years at MIT, thanks to the support of family, friends and colleagues.

I would like to thank my mother, Sri Hartati, for her unconditional and unfailing love, support, and faith – I appreciate it more than I can express; and to my father, Henderikus Alfred, for his love, confidence, and sacrifice throughout the years. Where I am today is thanks to their hard work and the opportunities that they provided for me. I thank my brother, Henderikus Budi Kurniawan, for his counsel and comfort. And I would like to thank my fiancée, Alessia, for her love, encouragement, and happiness – I cannot imagine going through this journey with anyone else.

To my friends and colleagues, I thank them for making these years a little less stressful and a lot more fun. Thanks to Pedro, Arturo, Dehann and Mike for being great friends. And thank you to Oscar, Greg, Supun, Rui, Kristen, Eeshan and others in MIT LAMSS for their help, support and friendship over the years.

To my advisor, Henrik, I offer my truly sincere thanks for his unwavering support, advice, and friendship, and for the opportunity to work on a project that could not be better suited for me – I cannot imagine a better advisor. To members of my committee, Leslie Kaelbling, Arthur B. Baggeroer, John J. Leonard, and Erin M. Fischell, thank you for the discussions, suggestions, and for making my work better. An additional thanks to Erin for her friendship – seeing her path has been an inspiration.

This research was funded and supported by a number of sponsors; we gratefully acknowledge them below.

- Defense Advanced Research Projects Agency (DARPA) and SSC Pacific via Applied Physical Sciences Corp. (APS) under contract number N66001-11-C-4115.
- SSC Pacific via Applied Physical Sciences Corp. (APS) under award number N66001-14-C-4031.
- Air Force via Lincoln Laboratory under award number FA8721-05-C-0002.
- Office of Naval Research (ONR) via University of California-San Diego under award number N00014-13-1-0632.
- Defense Advanced Research Projects Agency (DARPA) via Applied Physical Sciences Corp. (APS) under award number HR0011-18-C-0008.
- Office of Naval Research (ONR) under award number N00014-17-1-2474.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ADAPT	DARPA ADAPTable Sensor System
AHRS	Attitude and Heading Reference System
ASV	Autonomous Surface Vehicle
AUV	Autonomous Underwater Vehicle
BFF	Body-Fixed Frame
CBF	Conventional Beamformer
CDF	Cumulative Distribution Function
CSAC	Chip-Scale Atomic Clock
CTD	Conductivity-Temperature-Depth
CZT	Chirp Z-Transform
DAQ	Digital Acquisition Device
DARPA	Defense Advanced Research Projects Agency
DFT	Discrete Fourier Transform
DGPS	Differential GPS
DOA	Direction of Arrival
DVL	Doppler Velocity Log
ENU	East-North-Up
EPD	Element Pair Decomposition
GNSS	Global Navigation Satellite System

GPS	Global Positioning System
HMM	Hidden Markov Model
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
LAMSS	Laboratory for Autonomous Marine Sensing Systems
LBL	Long Baseline
LFM	Linear Frequency Modulation
LLF	Local-Level Frame
LTI	Linear Time-Invariant
MEMS	Micro-Electro-Mechanical Systems
MIT	Massachusetts Institute of Technology
MLE	Maximum Likelihood Estimate
NIST	National Institute of Standards and Technology
OWTT	One-Way Travel-Time
PCR	Pulse Compression Ratio
PHAT	Phase Transform
PID	Proportional-Integral-Derivative
piUSBL	Passive Inverted Ultra-Short Baseline
PPS	Pulse-Per-Second
PSD	Power Spectral Density
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
RPM	Rotations-Per-Minute
SBL	Short Baseline
SLAM	Simultaneous Localization and Mapping

SMCB	Sequential Monte-Carlo Beamformer
SNR	Signal-to-Noise Ratio
TCXO	Temperature Compensated Crystal Oscillator
TDOA	Time Difference of Arrival
USB	Universal Serial Bus
USBL	Ultra-Short Baseline
VCF	Vehicle-Carried Frame
WHOI	Woods Hole Oceanographic Institution

Chapter 1

Introduction

Seventy percent of the Earth is covered by the ocean, and less than five percent of the ocean has been explored – as a result, approximately sixty-six percent of our planet remains unknown and hidden to humanity. The primary reason for this opacity is the nature of the underwater environment: it is inherently hostile to human life and technology – enormous pressures, corrosiveness, darkness, and opaqueness to standard radio communications all contribute to an environment which is extremely difficult for humanity to explore. In his 1989 paper, 'Fast Cheap and Out of Control' [1], Rodney Brooks argued for the use of large numbers of cheap and simple robots for the exploration of the solar system, rather than complex and expensive systems. In many ways the exploration of our oceans is analogous to space exploration; and robots are ideally suited to be the pioneers of discovery in the underwater domain – the ocean environment is vast, and it is difficult, dangerous and dull for human explorers.

Since the development of the first generation of underwater remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) in the 1960s and 1970s, advances in inertial navigation and acoustic sensing technologies have enabled these underwater robots to perform reliably, particularly in terms of navigational accuracy. As a result, they have now become an integral instrument for a variety of applications in oceanographic science and defense. Unfortunately, the sophisticated inertial and acoustic sensors (namely the high-grade inertial measurement unit (IMU) and the Doppler velocity log (DVL)) that have enabled this navigational reliability are expensive, power-hungry and large, and have driven up vehicle price and size. The high value and large size of these vehicles make them unwieldy to deploy, often requiring the use of a support ship at considerable expense. The cost and complexity of deployments also makes operators extremely risk averse.

The vastness of our oceans is a strong argument in favor of using multiple AUVs to explore the underwater environment; but due to the price and difficulty of deployment of conventional AUVs, this goal is out of reach to all but a few wealthy companies and nations.

In much the same way that Brooks argued for the use of many small, inexpensive robots for the exploration of space, the same argument holds true for ocean exploration. However, while much has been written on how the coordinated use of multiple underwater vehicles would enable more operationally robust and efficient ocean sampling and surveying, this long-held dream has yet to be really achieved. Low-cost underwater vehicles are now arriving, but they are hampered by the lack of a similarly low-cost navigation suite – this is the missing key that will allow multi-AUV ocean deployments to be democratized.

This is what we mean when we talk about working towards ubiquity in underwater robotics: ubiquity in terms of democratizing underwater vehicle technology, making it more widely available and accessible to a larger number of people. And ubiquity in terms of deploying many such underwater vehicles at once – not just one, two or ten vehicles – but tens of vehicles. Brooks vision of planetary and space exploration using swarms of robots has yet to pass – the prevailing paradigm of ever more costly and complicated one-of-a-kind rovers, from *Sojourner*[1], to *Spirit* and *Opportunity*[2], to *Curiosity*[3], and the next rover for Mars 2020[4], is maintained by the momentum of its success, leading to greater expense, more complex science-specific payloads, and a greater psychological investment in their success or failure. The logistical costs and limitations, as well as advantages, of space exploration has meant that the time is not yet right for robot swarms to explore our solar system, although this may soon change with the confluence of lowering launch costs, and continued advances in swarm and robotic navigation and autonomy research. Meanwhile, in the underwater domain, a different set of constraints and incentives has meant that the time is now right to make this transition from single high-cost vehicle deployments to deployments of multiple low-cost vehicles, with many of Brooks' original ideas about behavior-based autonomy and control [2] having been proven out on AUVs over the past two decades. Underwater and space robotics share many similarities – both deal with harsh environments, in which communication is severely limited by some combination of bandwidth and latency, where platforms must be hardened or mechanically reinforced, where a team of operators must work to plan, deploy, and execute missions, and where the cost of operations is high. The difference between the two comes in the greater communications restrictions underwater, but the much lower comparative cost of operations – this has both forced and given the opportunity to underwater robotics researchers to develop robust underwater navigation and autonomy algorithms over faster iteration cycles. The additional incentive of underwater applications that can *only* be accomplished using multi-AUV deployments, means that the underwater environment is a prime domain for significant developments in multi-robot exploration – motivations and costs are aligned for multi-AUV deployments to make an enormous impact.

[1] https://www.nasa.gov/mission_pages/mars-pathfinder/
[2] https://mars.nasa.gov/mer/
[3] https://mars.nasa.gov/msl/
[4] https://mars.jpl.nasa.gov/mars2020/

Figure 1.1: Our fleet of three low-cost, miniature commercial Bluefin SandShark autonomous underwater vehicles (AUVs) ready for deployment dockside at the Massachusetts Institute of Technology (MIT) Sailing Pavilion. They are named *Platypus*, *Quokka* and *Wombat*.

This work describes two things: a technological approach for low-cost, low-power acoustic navigation for underwater robots that is scalable to many vehicles, and which does not require the sophisticated and expensive sensors traditionally used for underwater navigation; and an operational approach for a single operator to command and control multiple underwater vehicles using this navigation system. The hope is that this new paradigm will help push the field toward the democratization of underwater vehicle technology, and enable multi-AUV deployments to become commonplace. To demonstrate the reliability and robustness of our approach, this work details multiple deployments of three low-cost, miniature AUVs (pictured in Fig. 1.1) using our navigation and operating paradigm, providing one of the few instances performed of sustained and repeated multi-AUV deployments.

1.1 Underwater Robotics

The decades since the emergence of the first ROVs and AUVs has seen rapid progress in the technological capabilities of these underwater robots, especially in the 20 years between 1990 and 2010 – a large number of different types of underwater vehicle have been developed, inspired by and tailored for various applications in the underwater domain. The surrounding technologies that support these vehicles have matured to the point of reliability, in areas such as navigation, communication, sensing and the manufacture of high pressure housings, which themselves have been enabled by the rise in cheaper and faster computing, micro-electro-mechanical systems (MEMS) manufacturing processes, and advances in signal processing, sensor fusion, and autonomy algorithms [4]. We briefly discuss the history of underwater

robotics, starting from the development of the first ROVs in the 1960s, through the development of AUVs and their maturation in the 2000s, and now to the emergence of low-cost and miniature underwater platforms in both the commercial and consumer sector.

1.1.1 Remotely Operated Vehicles (ROVs)

The development of the first ROVs began in the 1960s, where they were developed in parallel with some of the first manned submersibles such as *Alvin*[5] for the purposes of fine manipulation and inspection tasks near the seafloor [6]. These vehicles are teleoperated via a tether connection between them and a 'mother' surface ship, through which the operator sends direct control commands, receives sensor information and telemetry, and from which the vehicle receives power [11].

The genesis of the first ROV was from the development under contract for the U.S. Navy of a system designed for the recovery of torpedoes lost on the seafloor – developed in 1961, the Cable-controlled Underwater Research Vehicle (CURV) was essentially a large, heavy, and highly-maneuverable underwater camera system that could be equipped with a claw for object retrieval. The CURV famously retrieved a lost atomic bomb off the coast of Spain in 1966, and its successor, CURV III, saved the pilots of the manned submersible PISCES III during a rescue operation off the coast of Ireland in 1973 [7]. The CURV family of vehicles are still in use today, with the latest incarnation, CURV-21, equipped with an array of modern vision and acoustic sensors, and manipulators.

The success of early government-funded ROVs in the U.S. and Europe led to a surge in the number of ROVs built in the decade starting from 1975, spurred on by technological advancements such as the growth of the electronics industry, which allowed the necessary on-board systems and sensors to be miniaturized. By the end of the 90s, ROV technology had reached a point of maturity, having been developed to that point through investment from industry, and in particular offshore oil and gas, which required this technology to install, operate, maintain and monitor subsea infrastructure at depths well beyond that which divers can reach [7].

These ROVs, used by the military, oil and gas industry, and oceanographic organizations for fine manipulation or sampling tasks at large depths, are often classified as *Work Class* ROVs. This class of ROV, such as *Jason*[6] belonging to the Woods Hole Oceanographic Institution (WHOI) [8], are box-shaped, utilize multiple thrusters for movement in any direction, are often equipped with one or two robotic arms, weigh in the multi-1000 kg range, and can dive to depths greater than 4 km [9].

[5] https://www.whoi.edu/what-we-do/explore/underwater-vehicles/hov-alvin/history-of-alvin/
[6] https://www.whoi.edu/what-we-do/explore/underwater-vehicles/ndsf-jason/

The maturation of ROV technology during the 1980s and 1990s also saw the emergence of smaller vehicles used primarily for observation and close inspection of underwater structures. These so called *Observation Class* ROVs, such as the VideoRay[7], are typically equipped with a camera and a sonar for use for close survey and inspection; like their larger brothers, they are box shaped and equipped with multiple thrusters, but are considerably smaller, weighing in the range of tens of kilograms, and can typically dive only to depths of hundreds of meters.

ROVs distinguish themselves by their ability to finely control their movements thanks to their multiple thrusters, and to manipulate the environment around them through the use of underwater manipulators. Teleoperation via the surface tether means that they essentially lack all but basic autonomy, and are tethered to a surface platform for power and control, limiting their range. As such, they are designed specifically for applications that require dexterous intervention, close inspection, or precision sampling [10].

1.1.2 Autonomous Underwater Vehicles (AUVs)

Unlike ROVs, AUVs typically have no tether to the surface and are not teleoperated, having to navigate and perform pre-programmed or adaptive missions fully autonomously. The first AUVs were built in the 1960s and 1970s, including the Self-Propelled Underwater Research Vehicle (SPURV) from the University of Washington [5] and its successor, SPURV II, and were used to study oceanographic processes, underwater physical properties, and submarine wakes [3]. Other AUVs developed in the 1970 include the Autonomous Underwater Search System (AUSS), motivated by the loss of the Navy submarines USS Thresher and USS Scorpion, as well as *L'Epaulard* from the French institute IFREMER, which was used to perform photographic and bathymetric surveys.

The same technological advancements that spurred the growth of ROVs in the 1980s enabled the renewed academic interest in AUVs in the late 1980s and early 1990s, including smaller and lower power computers and denser memory which enabled complex guidance and control algorithms to be implemented for autonomous operation, as well as advances in software systems for simpler system development [4]. Both the Massachusetts Institute of Technology (MIT) and WHOI developed AUVs in the 90s, with the MIT Sea Grant[8] AUV Lab's *Odyssey*[9] vehicles providing inspiration to transition this knowledge into industry with the incorporation of Bluefin Robotics[10] in 1997; WHOI developed the Autonomous Benthic Explorer (ABE)[11] during this same time-frame (unfortunately lost at sea near Chile in 2010),

[7] https://www.videoray.com/
[8] https://seagrant.mit.edu/
[9] https://auvlab.mit.edu/vehicles/
[10] https://gdmissionsystems.com/en/underwater-vehicles/bluefin-robotics/
[11] https://www.whoi.edu/oceanus/feature/abe-mdash-the-autonomous-benthic-explorer/

and one of the most successful commercial AUVs, the Remote Environmental Monitoring UnitS (REMUS)[12], was developed by the WHOI Oceanographic Systems Lab in the 90s [3].

The decade starting in the year 2000 saw the maturation and commercialization of AUV technology, with the continued improvement of its technological components, including sensor miniaturization via MEMS processes, improvements in battery technology enabling longer endurance, and algorithmic improvements in navigation and autonomy increasing reliability. AUVs come in a variety of sizes, but generally all have a familiar torpedo-shaped body for reducing drag and power-use and improving vehicle run-time. The largest vehicles, such as the Bluefin-21, the REMUS 6000 and the HUGIN are between 50 cm to 90 cm in diameter, weigh hundreds of kilograms, and can dive to depths of multiple kilometers. Mid-sized AUVs such as the REMUS 600 and the Gavia have a diameter of 20 cm to 30 cm, and can dive to a depth of up to 1 km. Man-portable AUVs are smaller, with a diameter of around 20 cm, a weight of 30 kg to 100 kg, can dive to a depth of up to 200 m, and include the Bluefin-9 and REMUS 100. These vehicles are typically ballasted to have a slight positive buoyancy, allowing them to surface automatically in cases of system failure – this means that they must maintain forward thrust in order to remain submerged. Vehicle lengths vary between 1.5 m to 6 m, and vehicle speeds can range between 0.5 $\mathrm{m\,s^{-1}}$ up to 3 $\mathrm{m\,s^{-1}}$. Torpedo-shaped AUVs were originally focused for use in performing survey and mapping missions of the seafloor, typically using sonar or camera sensors, where their high speed allows them to survey large swaths efficiently; however, these vehicles are increasingly being used to collect water column geochemical and oceanographic measurements [12].

A second class of AUV is known as the underwater glider, a concept originally developed in the 1990s and motivated by the need to collect mid-column oceanographic measurements, which the original propeller-driven AUVs were not designed for. Originally devised by Webb and Stommel at Woods Hole, the concept of the glider was to make use of changes in buoyancy for forward propulsion via 'wings' which impart a force during vehicle ascent or descent. Changes in pitch and roll are achieved through the fore and aft and roll movement of an internal battery pack. Because these vehicles use very little energy to fill and empty a buoyancy reservoir (and only at the end of an ascent or descent), its means of propulsion is extremely efficient, using on average less than 1 W of power, and enabling several 1000 km long transects. Gliders such as the Slocum, WHOI Spray, and Seaglider can have an endurance of tens to hundreds of days, with a forward speed of 0.2 $\mathrm{m\,s^{-1}}$ to 0.5 $\mathrm{m\,s^{-1}}$. Their low-power use and consequent endurance make them ideal for sampling interior ocean properties, but limit the sensors that can be included in their payload [13].

The past decade has also seen the development of hybrid ROV/AUVs that balance the advantages and drawbacks of both types of underwater robot; these vehicles include the WHOI

[12] https://www.whoi.edu/what-we-do/explore/underwater-vehicles/auvs/remus/

Nereus[13], which was unfortunately lost in 2014 [14], and the Seaeye Sabertooth [15], vehicles that would allow the operator to switch between a fast-moving survey mode, and a slow, fine-control inspection and manipulation mode.

1.1.3 A New Generation: Low-Cost and Miniature AUVs and ROVs

The past few years has seen the rise of a new generation of low-cost and miniature underwater vehicle, driven by further improvements in the miniaturization of computation and sensing, as well as advances in additive manufacturing – in particular, 3D printing has enabled rapid development and testing of these miniature AUVs, such as the Riptide micro-AUV, the eco-SUB [190], and the Bluefin SandShark [189]. These AUVs have a diameter between 10 cm and 15 cm and a length of 1–2 m, with a weight of 10 kg to 20 kg. They can generally dive to around 200 m, and have a fairly low endurance on the order of 10 hours. Consumer ROVs, such as the Blue Robotics BlueROV2, have also emerged, providing a platform inexpensive enough for the average hobbyist to acquire. These miniature and low-cost underwater robots hold the potential to democratize underwater robotics, but their low-cost and small size prevent the use of traditional navigational and acoustic sensors, limiting their utility as a result of their low navigational performance [190] [152] [153].

1.1.4 Current and Potential Applications

Far from being used only for their original applications of manipulation and seabed surveying, ROVs and AUVs nowadays perform a wide variety of different applications. These include sampling dynamic ocean processes as a Lagrangian point sampler, tracking ocean plumes, mine countermeasures and harbor patrol, surveying the underside of ice-sheets, and detecting, localizing and tracking acoustic sources in the ocean, among other things [16]. Here we describe a few applications performed by these underwater robots, and illustrate how their utility is directly reliant on their ability to accurately self-localize underwater, and to the uncertainty of their position estimate – measurements gathered by an underwater robot are only useful to human operators if they know where the data was collected from.

Surveying and Mapping

AUVs were originally designed to map and survey the seafloor, and this remains one of their primary use-cases. Typically these vehicles are equipped with a sidescan sonar, which allows them to acoustically image large swaths of the seabed either side of the vehicle – sidescan sonar surveying has been used for mine detection [17] [18], analysis and characterization of seafloor texture [19], for mapping and classifying benthic marine habitats [20], to image hydrothermal

[13] https://web.whoi.edu/hades/nereus/

plumes [21], and for search and salvage operations [22] [23]. Surveys often complement sidescan sonar data with data from other sonars, such as multibeam or pencil sonars, which allow the vehicle to generate bathymetric height-maps of the seafloor – multibeam sonars like these allow operators to generate highly detailed maps, allowing them to analyze seafloor morphology, detect coral mounds, and classify bottom types using acoustic backscatter [24]; AUVs mounted with these sonars have also enabled novel applications, such as the acoustic imaging of the underside of ice sheets and ridges [25] [26] [27], providing scientists with a direct measure of changes to sea ice thickness over time; an additional sonar sensor known as a subbottom profiler can also enable AUVs to image structures below the seafloor using low frequency acoustics – the combination of these three sonar types allow oceanographers to map and classify in great detail the structure and form of benthic habitats [28].

Sonars have also been used in conjunction with camera-based imagery for detailed mapping in marine archaeology, using both AUVs and ROVs. Vision systems underwater require strong artificial illumination, and rapid attenuation by water of frequencies in the visible wavelength mean that imagery must be captured with the vehicle close to the object – as a result, thousands of images of the object are often captured and mosaicking techniques are employed offline to 'stitch' together the scene of the seafloor or wreckage. Vision and sonar imagery have been used together to map ancient shipwrecks by AUVs [29] and ROVs [30], as well as to survey archaeological sites [31], reef and coral habitats [32], and hydrothermal fields [34]. More recently, structured light and laser-based approaches have been used on board ROVs to perform very high resolution mapping of underwater objects and the seafloor [33] [35], in which a line laser shone on the seafloor is observed by a single camera to capture shape and structure.

Mapping and surveying remains one of the most important applications of underwater robotics, and provides great utility to the operator in providing an understanding of the makeup and formation of the seafloor and seabottom objects. In order to create these maps, sensor data gathered by the robot is combined with its navigation and positional data offline to geo-reference sonar or visual imagery – as a consequence, the quality of the map, and the accuracy of data positioned within it, are directly dependent on the estimate of vehicle position [36]. These are just a few applications of AUVs and ROVs for mapping, with ever more novel use cases being found in the fields of marine geology, fisheries, and oceanography.

The deployment of multiple low-cost AUVs has the potential to revolutionize many surveying and mapping applications, simply through the ability to parallelize the mapping process – whether through vision or acoustic imaging, the use of multiple vehicles has the potential to speed up surveying, or to make maps more robust via the survey of a single area multiple times. If multi-AUV deployments are made easier, it opens up the potential to survey the same habitat more often, providing scientists with a better understanding of habitat and environmental changes.

Sampling and Tracking

The earliest AUVs were test-beds used to acoustically and physically sample oceanographic processes, and nowadays all AUVs and ROVs carry a conductivity-temperature-depth (CTD) sensor to measure seawater salinity. Some of the earliest concepts for the use of multiple AUVs and gliders has been for the measurement and sampling of ocean processes to better understand their spatiotemporal dynamics, as described by Curtin and Bellingham in [37]. To some degree this vision of mobile underwater sensor networks has been realized over large spatial scales and low resolution using underwater gliders, primarily through the work of Leonard [38] [39]. AUVs and ROVs have also been equipped with chemical sensor payloads to sample and map chemical plumes such as methane [40], Rhodamine dye [41], and plumes from hydrothermal vents [42], as well as from chlorophyll blooms [43].

There is a wide body of literature focused on using AUVs to dynamically track ocean processes and plumes in the ocean, an application well-suited to these vehicles due to their ability to autonomously direct themselves. AUVs have been used to track oil spills [44], thermoclines or temperature gradients [45] [46], ocean fronts [47] [48], and oceanographic plumes [49].

AUVs have also been used to detect and track acoustic sources. These include tracking non-cooperative sources using a nose-mounted linear hydrophone array [50] [51], as well as towed arrays [52]; bistatic and multistatic methods, in which a source is used to insonify an area and a receiver on a single or multiple vehicles is used to detect and track reflective seabed objects has also been demonstrated [53] [54] [55]; AUVs with towed arrays have also been used to monitor and track natural acoustic events, such as ice cracking [56].

As with mapping and surveying applications, the use cases of sampling and tracking with underwater robots has a strong need for accurate navigation, although this dependence is somewhat weakened. Since sampling strategies are moving towards greater autonomy, with vehicles dynamically altering their behavior to track and sample oceanographic processes, or to detect and track acoustic events, the global position of these samples or events may have less importance than their structure. In addition, various ocean processes happen on vastly different temporal and spatial scales – depending on the type of process that is sampled, the positional accuracy desired can range from sub-meter to multi-kilometer [57].

Multi-AUV missions have the greatest potential to make significant contributions in applications that involve sampling and tracking. One of the biggest challenges of sampling from ocean processes is that of spatio-temporal aliasing – if samples are taken too far apart in some combination of space and time, then the features of the process can not be resolved, and may be attributed to either a spatial or temporal change [57]. The use of multiple vehicles can break this trade-off between fine spatial *or* temporal sampling, by simply sampling the process at multiple positions over the same timescale, effectively resolving this ambiguity. In

terms of tracking, multi-AUV deployments also hold great potential to upend existing strategies for acoustic event tracking, through the use of multiple vehicles as single elements in a *virtual acoustic array* [59] [60] – such an array could be commanded to dynamically alter its geometry, optimizing the placement of its 'elements' in order to track an acoustic event [58].

Inspection

ROVs have long been used to perform inspection and observation tasks, an application that they are especially suited for due to their fine motion control. These applications use such robots to inspect man-made underwater objects [10], including complex structures such as subsea facilities [61] and ship hulls [62]. However, these inspection tasks are becoming increasingly more automated, with specialty vehicles such as the Bluefin Hovering AUV (HAUV) being used to autonomously scan and map ship hulls [63], generating motion plans dynamically to inspect complex sections of the hull including the propeller [64]. These essentially 'autonomous' ROVs have also been used to inspect structures like hydroelectric dams [65].

Traditional torpedo-shaped AUVs have also been used for inspection tasks, in particular the inspection of seabed pipelines and underwater cables [66] [67], tasks which are particularly suited to them due to the length of this infrastructure, and the desire to autonomously detect and track pipeline or cable features.

Recent work has also focused on acoustic inspection and classification of underwater targets using bistatic sampling techniques – acoustic energy reflected off a seabed object is collected by an AUV, and is used to classify the object [68].

Inspection tasks such as these often use relative control algorithms to maintain a relative position between the vehicle and the object being inspected – as such, there is less need for accurate global positioning. In addition, the presence of structures often mean that advanced processing techniques such as simultaneous localization and mapping (SLAM) and photogrammetry are used for navigation. Multi-AUV deployments do have the potential to impact these applications, but to a lesser extent than mapping or sampling – for example, multiple vehicles can be used to ease the need to perform pipeline tracking perfectly, by flying vehicles in formation side-by-side. In addition, acoustic inspection of seabed objects via multistatic scattering is a possible application of multiple AUVs.

1.2 Localization, Navigation, and Tracking

We have seen how AUVs and ROVs have revolutionized surveying, mapping, sampling and inspection tasks in the ocean. This has been achieved through sophisticated and expensive navigational sensors that allow these underwater robots to navigate and perform reliably underwater. The drastically lower cost of the new generation of miniature underwater vehicle

holds the potential to enable multi-vehicle deployments, but their size, cost, and power budget unfortunately make these traditional navigational sensors ill-suited for these new platforms. One of the primary contributions of this work is in providing a low-cost localization and navigation suite that is well suited for this new generation of vehicles. In this section we review current solutions for underwater vehicle navigation, and discuss some related concepts from other fields which are relevant to the approach described in this work.

1.2.1 Underwater Navigation

Accurate localization for a robotic vehicle is often essential for the purposes of path planning or geo-referencing of scientific measurements. For underwater robots, accurate and reliable navigation in unstructured underwater environments is a major challenge: global positioning system (GPS) and radio frequency signals do not work underwater, and communication is unreliable and limited in bandwidth. Commercial AUVs typically rely on a DVL-aided inertial navigation system (INS) [69] [70] to navigate between periodic GPS surface fixes to limit unbounded error growth.

The Standard Navigation Technique

The vast majority of underwater vehicles navigate through the use of a DVL-aided INS to estimate their longitudinal (x) and latitudinal (y) position; estimation of depth is decoupled through estimation via a pressure sensor. Pressure sensors allow vehicles to easily and accurately estimate their depth via a polynomial fit of pressure to depth developed by Fofonoff using the hydrostatic equation and Knudsen-Ekman equation of state [183]. Depth estimates using a pressure sensor typically have an uncertainty of 0.01% of their full scale, which translates to less than 7 cm [71]. Vehicles are also typically equipped with a GPS receiver, allowing them to localize themselves when on the surface – absorption of the electromagnetic spectrum by water prevents the use of GPS when submerged.

The workhorse navigational sensor for any conventional AUV or ROV is the Doppler velocity log (DVL). The principle of the DVL is fairly simple: a set or array of transducers are used to transmit an acoustic pulse, which is reflected off the seafloor and returns to the sensor – since the vehicle is moving, these reflections are recorded by the transducers with a frequency (or Doppler) shift that is proportional to the speed at which the vehicle is moving; when compensated for with the orientation of the vehicle, the sensor is able to determine the vehicle's velocity in all three axes of the global frame. Older DVLs typically use four transducers mounted at an angle to the seafloor in a Janus configuration, allowing it to solve for the Doppler shift via the solution of four simultaneous equations [72]; these DVLs are large, at approximately $20 \times 20 \times 20$ cm and weighing about 15 kg. However, newer DVLs are now available which used a phased array for digital beam steering, and which are

significantly smaller and lighter (around $20 \times 10 \times 10$ cm and 4 kg) allowing them to be used on miniature underwater robots [73]. The DVL allows the vehicle to obtain accurate readings of its speed-over-ground, with an uncertainty on the order of centimeters per second or less.

All underwater vehicles are equipped with an inertial measurement unit (IMU) or attitude and heading reference system (AHRS). These sensors typically consist of a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer. An IMU and an AHRS differ in that the AHRS will use the raw linear accelerations from the accelerometers, rotational velocities from the gyroscopes, and heading from the magnetometers, and combine these measurements to produce a vehicle attitude (orientation) solution – this is performed by estimating the gravity vector using accelerations, and using this vector with integrated gyroscope velocities to obtain an estimate of roll, pitch and yaw in the world frame. The cost of an AHRS can vary drastically depending on the quality of the solution, and is mostly related to the noise and drift inherent in the gyroscopes. AHRS that use MEMS gyroscopes cost less than $1000, but drift at a rate of tens of degrees per hour; ring laser and fiber-optic gyroscopes cost tens of thousands of dollars, but drift at a rate of less than a degree per hour. The magnetometer within the AHRS assist in limiting this drift (which translates to a non-bounded drift in heading) by providing a reference against the Earth's magnetic field, but is subject to errors due to environmental magnetic anomalies, as well as electrical currents within the vehicle itself.

The DVL and the AHRS are combined into a single inertial navigation system (INS), known as a DVL-aided INS. The standard navigation technique for a conventional AUV is to localize itself using GPS on the surface, and when submerged, propagate its position using inertial navigation or *dead-reckoning*. Essentially, the speed-over-ground from the DVL and the attitude from the AHRS are used to estimate the current vehicle position by integration, usually through the use of an extended Kalman filter (EKF) [75]. However, since this technique has no absolute measure of vehicle position when underwater, the error from inertial navigation grows unbounded [74], and must be 'reset' periodically by surfacing to receive another GPS fix. A typical AUV using a high-grade MEMS IMU will typically have an inertial navigation drift of 1%-5% of distance traveled [149], while those with a fiber-optic gyro can achieve error rates of 0.1%-1% of distance traveled [76].

To bound the error growth of inertial navigation, underwater vehicles often make use of external localization techniques that provide them with some global measure of vehicle position. These approaches generally fall into two categories: acoustic or geophysical positioning [77] [78] [79].

Geophysical Localization Techniques

Geophysical techniques include terrain relative [80], visually-augmented [81], and sonar-aided navigation [82]. These approaches generally sense features in the environment, either through

measurements of altitude, or through visual or acoustic imaging, and use those features to place themselves within a known map. Alternatively, image based approaches can estimate the vehicle's change in position by the relative offset between consecutive images. Although terrain relative and visually-augmented navigation can be performed with inexpensive, low-power sensors (altimeter and camera respectively), both have drawbacks; the former requires prior maps of the seafloor, which often lacks features and is vulnerable to environmental change, and the latter can only be performed in clear water, or using high-power lighting. Sonar-aided navigation requires costly sidescan [84] or imaging [83] sonars.

Acoustic Localization Techniques

In contrast, environmental conditions have comparatively less effect on acoustic positioning approaches, and localization is possible with the use of cheap hydrophones. Classical acoustic systems use multiple distributed transponders in fixed positions which respond to vehicle ranging requests, allowing AUV self-localization via trilateration and two-way travel-time ranging [75]. One limitation of these so-called long baseline (LBL) or short baseline (SBL) systems is the time and expense associated with setting up the network of beacons. Early work by Newman and Leonard [85], Vaganay [86] and Curcio [87] sought to improve ease-of-use of LBL systems, by removing the need to manually geolocate the transponders either through simultaneous localization and mapping (SLAM) techniques, or by using surface vehicles with high GPS navigation accuracy as mobile transponders in a moving LBL approach. However, even with these improvements, these approaches always retain the disadvantage of requiring multiple transponders, which adds to cost and carries a penalty in terms of setup time or communications.

Ultra-short baseline (USBL) and inverted USBL (iUSBL) systems also exist, which are able to provide a full navigation fix using a single transponder. This transponder contains an acoustic receiver array that is used to interrogate the vehicle; the relative range and bearing from the transponder to the vehicle is then determined from the response using two-way travel-time and phase differencing, and vehicle position is estimated via triangulation. Recent work by Hodgkinson [89] and Morgado [90] have verified the feasibility of the iUSBL approach, enabling an AUV to self-localize through the fusion of triangulated acoustic measurements and odometry.

Unfortunately, the use of two-way travel-time in these approaches means that each AUV must carry an active acoustic system to transmit ranging requests, increasing price and communications cost. In addition, this severely restricts scalability, since the acoustic channel must be time or frequency shared across multiple AUVs and transponders. These limitations have recently been overcome with the advent of chip-scale atomic clocks (CSACs), which enable one-way travel-time (OWTT) ranging by synchronization of the clocks on both the

vehicle and transponder/s. These CSACs can enable OWTT LBL systems, as demonstrated in recent work by Melo and Matos [88], and have also enabled single-beacon OWTT navigation approaches such as those first outlined by Eustice [91], and improved upon by Webster [92]; these methods accrue OWTT range measurements over time and fuse them with dead-reckoning to bound positional error growth. Recently, work performed by Claus and Kepper have demonstrated closed-loop results of this single-beacon, one-way ranging approach using an EKF and a particle filter [93]. Unfortunately, these approaches require the vehicle to receive multiple range measurements from a variety of relative bearings in order to attain a suitably unambiguous positional fix.

The work detailed in this thesis overcomes many of the limitations of classical acoustic positioning systems, including price, communications cost, ease of use, and scalability, through an approach first suggested by Jakuba [95]. The approach suggested combines the advantages of OWTT passive reception, and the use of an inverted USBL receiver to semi-passively localize the vehicle relative to an acoustic source, providing an analysis of simulation results to demonstrate its feasibility.

1.2.2 Related Above-Water Localization Techniques

Although the vast majority of above-water robotic vehicles primarily make use of GPS for localization, there is a handful of work in above-water robotic navigation that is conceptually closely related to the work detailed in this manuscript. We describe this literature here, with acoustic and radio-based techniques that determine the range and bearing of the robot from a source being of particular interest, since these approaches use similar operational principles.

Radio Localization

Although not a robotic navigation system, early work in aerial and marine navigation led to the development of the Tactical Air Navigation System (TACAN). This system was developed in the 1950s, stemming from initial radio transponder navigation systems developed during World War II. TACAN emphasized the need for both bearing and range measurements from the aircraft to a single ground-based radio transponder, working over ranges of around 400 km, and allowing up to 100 aircraft to obtain relative bearing and range measurements simultaneously with high accuracy. Range is obtained via a two-way travel time interrogate-respond scheme, while bearing was determined by either physically or digitally rotating the beampattern transmitted by the transponder, allowing the aircraft's receiver to detect the amplitude-modulated signal. With the transponder pointing in a known fixed direction, and the rate of rotation known, the aircraft would thus be able to determine its relative bearing [96]. Thus, transponder-based range and bearing localization systems have long been in use, and were common before the advent of GPS.

More recent work by Cliff [97] has seen the use of a directional radio antenna mounted on a multirotor Unmanned Aerial Vehicle (UAV); consecutive relative bearing measurements made by the antenna are fused with orientation information from the UAV in a grid-based Bayesian filter to localize and track a radio-tagged bird online. These results built on previous work by Posch [98] in which he used a similar Yagi directional antenna and a particle filter to track a radio tag in simulation. A similar approach was demonstrated indoors by Isaacs [99] to localize a radio-frequency (RF) beacon, and this line of work continues with the use of additional directional antennas in work by Dressel [100] which again uses a discrete Bayes filter for the localization of an RF tag.

Acoustic Localization

Some early work in which the topics of acoustic source localization and robotics were first combined include that of Valin [125], in which a mobile robot was equipped with an array of microphones to localize sounds of interest in a room via time difference of arrival (TDOA) direction of arrival (DOA) estimation. This work was extended to perform DOA estimation of multiple acoustic sources by a mobile robot using a cascaded beamforming and particle filtering processing stack [101]. In general, a large variety of approaches have been used to address this problem of sound localization in the context of human-robot interaction, using various combinations of DOA estimation and Bayesian filtering approaches to detect and track the source. DOA estimation has been performed by robots using TDOA approaches, most often using generalized cross-correlation (GCC) methods, in which the signals from pairs of elements are cross-correlated to obtain a set of TDOAs which form a system of equations that can be solved simultaneously [126] [128] – this approach has been demonstrated in numerous experiments [125] [102] [103]. Beamforming methods have also been used, including conventional beamforming [101] [104] and the steered-response power (SRP) method [104] [106] [105]. A number of Bayesian filtering techniques have been incorporated with these DOA methods to track sound, including Kalman filtering [102], particle filtering [101] [107], and SLAM approaches [108] [109] [110]. In general, robot audition and acoustic localization is an active field of study, and good reviews of the field can be found at [111] and [112]. The majority of approaches use a TDOA scheme due to its computational efficiency, and because the unknown structure of the source signal means that performing multiple cross-correlations via the GCC improves detection of the vocal audio that the system desires to track.

In the context of above-water robot localization using acoustics, the state of the field is immature, but a few studies have been performed. Early work by Wang [113] used a speaker mounted on a ground robot and an array of 24 microphones distributed around a room to localize the robot using the SRP method under a near-field assumption. More recently, an inversion of this approach was demonstrated by Ogiso [114], in which four speakers were

placed around a room, and self-localization of a robot equipped with a microphone array was performed via an EKF with DOA measurements. Lin [115] was one of the first to demonstrate the possibility of cooperative relative acoustic localization with a team of three ground robots – in this work, each robot was equipped with a pair of microphones and a speaker; relative range between robots was determined via an interrogate-response scheme, while bearing was calculated using TDOA with the GCC method, with measurements fused via a particle filter. A more recent set of work by Basiri has extended this approach to aerial vehicles; this includes a tetrahedral microphone array mounted on a fixed-wing micro air vehicle, which used TDOA DOA estimation with particle filtering to perform bearing-only localization of an emergency whistle [117]; this work was extended to multiple fixed-wing vehicles, each equipped with a piezoelectric speaker, to localize the group via bearing-only estimates with a particle filter [118] and an EKF [119]; the same system on board pocket sized and standard quadrotors equipped with microphone arrays in outdoor and indoor settings was demonstrated in [116], using both passive and active sources. This set of approaches all used TDOA estimation for bearing-only acoustic localization, but also demonstrated the concept of navigating vehicles relative to each other via acoustic sources.

The concept of relative navigation illustrated by the TACAN system and by audio-based navigation approaches of Lin and Basiri is particularly useful to our work, and demonstrates that the idea of localizing a vehicle against a static beacon or a moving beacon is feasible and has utility. The work of Basiri in particular provides inspiration in the idea of acoustically navigating a robotic vehicle relative to a mobile acoustic source, demonstrated in his leader-follower aerial vehicle experiments in [116]; although the use of TDOA approaches is less attractive to us than beamforming-based approaches due to reduced robustness, and the addition of range measurements would drastically improve accuracy, the use of particle filtering for Bayesian estimation in acoustic localization context provided some inspiration for our work.

1.2.3 Acoustic and Radar Tracking

The approaches used for acoustic localization in the previously discussed literature make use of array processing and source localization and tracking techniques developed over many decades, stemming from work in the fields of acoustic and radar tracking.

Direction of arrival (DOA) estimation using phased arrays has a long history and is a very mature field of research; as such, we do not provide the reader with individual references, and instead point the reader to excellent reviews of current techniques [120] [174] and provide a very brief overview of the major approaches. The earliest approaches for DOA estimation were based on beamforming techniques [121], the conventional approach of which is to use the positions of the elements in the array and the theorized direction of the incoming signal to delay in time (or shift in phase) the signals as received by all elements – if the direction is correct, then

this spatial filter aligns all signals constructively [158]. This conventional beamformer (CBF) has fairly poor performance in distinguishing sources spaced closely together, and so due to the desire to improve this angular resolution, techniques to reduce the beamwidth were developed, which are described as so-called adaptive or super-resolution (subspace-based) beamforming methods. In the presence of multiple sources, Capon's method, or the Minimum Variance Distortionless-Response (MVDR) beamforming method, seeks to minimize the average output power while passing a signal arriving from the theorized direction [122]. The Multiple Signal Classification (MUSIC) method was also developed to improve angular resolution to detect multiple sources, and works by exploiting the structure of the autocorrelation matrix – essentially, the autocorrelation matrix is eigen-decomposed into sub-matrices which contain only the source signals, and beamforming is applied by weighting these components [123]. Another high-resolution DOA approach is known as Estimation of Signal Parameters via Rotational Invariance Technique (ESPRIT), which also uses the structure of the autocorrelation matrix – in this approach, the array is assumed to contain at least two identical *subarrays*, which are separated by some distance; these two subarrays experience a delay in the signals received on their elements, due to the distance between them, which ESPIRIT uses to solve directly for the DOA via root-finding of the two (or more) simultaneous equations related by the displacement of the subarrays [124]. Alternative approaches for DOA estimation are based on time difference of arrival (TDOA), in which the difference in time at which the signal is received by multiple elements is estimated, typically using the generalized cross correlation (GCC) family of methods [157]; the output of this cross-correlation for every pair of elements in the array is used to estimate the TDOA of element pairs, and the DOA is usually solved via the optimization of a nonlinear set of simultaneous equations [126]. This method was extended into a hybrid TDOA/beamforming method known as steered response power (SRP), in which the estimates of the TDOA for element pairs are aligned via a spatial filter over a set of candidate source positions, either assuming a theorized near-field position or theorized far-field direction [127] [128] [129]. Rather than improving angular resolution algorithmically, improvements were also gained by simply increasing the number of elements in the array and developing techniques to process those elements more efficiently. One of the techniques developed is known as *subarray processing*, and effectively performs digital beamforming on sets of elements within subarrays, then uses these sets as 'elements' of the larger array for beamforming [130]. The design of how such subarrays are selected is a small but active area of research [131] [132].

It is important to note that the trend of DOA research toward larger arrays and algorithms for improved angular resolution has been driven by the desire to detect and localize multiple radio-frequency or acoustic sources; this has led to more computationally and architecturally complex systems which require a large amount of computing power. This direction of research is somewhat orthogonal to ours, since in our work we assume the presence of only a *single*

source, and we emphasize the need for efficiency in computation and memory.

The source tracking problem can be considered as a dualism for positioning, and has been studied extensively in the context of acoustics and radar. These approaches generally make use of Bayesian filtering methods, with books available detailing radar tracking using Kalman filtering [133], particle filtering [134] [135], as well as covering both approaches [136]. With the need to track multiple targets, more recent approaches include particle filters for multi-target tracking [137], Kalman filtering with Gaussian mixtures [138], Guassian sum filtering [139], and variations of the probability hypothesis density (PHD) filter [140] [141]. Numerous studies have also been performed on the accuracy of target tracking applications, including the work of Moura [142], which derives analytical bounds on accuracy using a linear array under certain assumptions on source movement, as well the sensitivity of the solution to parameters such as signal-to-noise ratio (SNR) and range. In general, our work does not provide any theoretical guarantees such as these.

Similar to the trend of developing DOA estimation algorithms with improved angular resolution in order to better track multiple targets, work in tracking and filtering methods has tended to focus toward the multi-target application. Again, these approaches are of less interest to us, since we have the luxury of tracking a single target.

1.3 Demonstrated Multi-AUV Deployments

Since one of the primary goals of this work was to enable multi-AUV deployments, we briefly discuss some of the relevant and latest literature in the field of multi-AUV systems, limiting ourselves specifically to work that contains experimental results. Earliest experimental work in multi-AUV deployments include the work of Fiorelli [38], which concentrated on the deployment of three gliders in formation as well as a propeller-driven AUV to perform gradient climbing and sample temperature gradients. In these experiments performed in Monterey Bay in August of 2003, the gliders surfaced every two hours, at which point they communicated to a centralized computer in order to update their mission based on a virtual potential formation control scheme; thus centralized command and control was performed iteratively by calculating desired waypoints at this central computer, and broadcasting them to the fleet when the vehicles surfaced. Additional results from these experiments were reported by Leonard [39], which she used to derive further control laws for coordinated optimal sampling for a follow-up experiment planned at the time for August 2006.

This system was expanded upon in the work of Paley [143] into a full centralized command and control architecture called the Glider Coordinated Control System (GCCS), in which ocean models were simulated to predict glider motion when underwater, and updated with glider GPS position when they surfaced, with communications between the vehicles and the land-based GCCS occurring over the Iridium satellite network. Predicted positions of the

gliders were used to update control laws in the GCCS to perform desired missions and complete scientific objectives, with these coordinated trajectories transmitted and uploaded to the gliders upon surfacing. The system was validated with experimental deployments in Buzzards Bay in March 2006, demonstrating successful coordination when tidal flow was weak, but performance degraded substantially during periods of strong tidal flow.

This line of work ultimately led to the largest and longest deployment of multiple underwater vehicles to date, in which Leonard and others deployed a fleet of 10 gliders, 6 of which operated continuously for 24 days, with the desire to sample intermittent upwelling events in Monterey Bay in August of 2006 [144]. Again, the GCCS was used to predict glider trajectories using ocean models, and updated plans were transmitted to each glider upon surfacing, demonstrating sustained and automated coordinated control of these gliders.

Other examples of multi-AUV deployments include work by Das [145], in which a propeller-driven AUV and a glider were simultaneously deployed to observe the evolution of phytoplankton blooms in Monterey Bay in 2010. These deployments were performed as independent stages in the overall experiment, where assets were deployed each day and retrieved, their collected data analyzed in a central command room where the following mission was planned and the assets redeployed. Another example of experimental work in which multiple AUVs and gliders were essentially programmed to operate independently via a central command was recently demonstrated by Branch in May of 2017 [146]. In these experiments, each vehicle operated independently using an adaptive behavior to detect temperature fronts; upon surfacing, data uploaded to the central command was used to update an estimate of the front location, which was then used to transmit commands to the vehicles to transect this new front estimate. Work by Claus [93] performed in the fall of 2016 also demonstrated a similar operating paradigm in which two AUVs were operated together as independent platforms, but whose positions were augmented via range measurements to a single acoustic beacon. The issue with approaches such as these, where vehicles are periodically coordinated via a central command and control structure upon surfacing, is that their utility is limited to vehicles that operate over large spatial and temporal length scales, such as gliders, where precise sampling is of less importance, or they require the use of vehicles with high-grade navigational sensors.

Alternatively, field experiments have also been demonstrated in which multiple AUVs operate in a *coordinated* fashion, without the need to surface and transmit data to a centralized command. This includes early work by Soares [148], in which two leader AUVs transmitted their heading information via their acoustic modems to a single follower AUV; these bidirectional acoustic messages enabled the follower vehicle to determine its range to both leaders and maintain its position between them, as experimentally demonstrated in a saltwater bay in Lisbon in June of 2012. Early work by Petillo [149] also experimentally demonstrated the use of two AUVs for the adaptive detection and characterization of internal waves in the Tyrrhenian Sea in August of 2010; during these experiments, a follower AUV adaptively trailed a

leading AUV, with both vehicles dynamically modifying their depths to maintain position within a thermocline in order to characterize the internal wave structure. Acoustic modems were used to communicate relevant information between the two vehicles as well as a topside ship-based command center. Other experiments include those performed by Walls [150], in which two AUVs demonstrated a cooperative localization scheme where OWTT range factors between the vehicles were communicated to each other acoustically, and used within a factor graph framework for localization. More recently, in 2013 and 2014, Lin [147] performed a two-AUV deployment, where each vehicle was equipped with a stereo hydrophone system to estimate the relative distance and bearing to an acoustic transmitter tag for the tracking of marine life; the AUVs each exchanged their estimate of the tag location via acoustic modems, allowing each to fuse the other's estimate of tag position into their own solution; this system was used to autonomously track and follow the tag as it moved. The issue with these in-situ coordinated approaches is that they require active bidirectional communications between vehicles, which limits their scalability for large groups of vehicles.

1.4 Motivation

The previous section has illustrated how autonomous underwater vehicles (AUVs) and remotely operated vehicles (ROVs) have become an invaluable asset for defense, industry, and oceanography, having expanded dramatically out of their original application areas of seafloor mapping, and inspection and intervention. Applications range from acoustic and visual surveying, to mine countermeasures and inspection, to chemical and biological sampling, and adaptive plume and target tracking. We have seen how this rise in their use has been driven by improvements in their reliability, in a large part thanks to the development of sophisticated navigational sensors, namely the Doppler velocity log (DVL) and the high-grade attitude and heading reference system (AHRS), which allow them to perform inertial navigation to a high degree of accuracy. This improvement is a double-edged sword, however – the high cost, power use, and size of these sensors mean that conventional vehicles are large and expensive. This cost has stifled the democratization of this technology, and hindered its widespread use. The size and complexity of these vehicles has also meant that their deployment is unwieldy and expensive, often requiring specialized personnel for their operation, and often with the need of a support vessel, driving up operational expenses. As a result, vehicle operators tend to be very risk averse.

The risk averse nature of operations, as well as the fundamental limitations of the underwater environment, has meant that multi-AUV operations are rare. The acoustic channel is highly limited in bandwidth and channel capacity, restricting communications between vehicles, as well as limiting scalability due to the need to time or frequency share the channel – as a result, multi-vehicle cooperative navigation schemes are limited in scale. Alternatively,

multi-AUV operations that limit inter-vehicle communications need significant operator input for planning and execution of individual vehicle missions, requiring several personnel at considerable expense.

The recent emergence of simpler, low-cost and miniature underwater vehicles in the commercial and consumer sector has the potential to upend the current operating paradigm of large and expensive vehicles deployed with the assistance of support infrastructure. With each vehicle in the tens of thousands rather than hundreds of thousands or millions of dollars, riskier autonomous behaviors and adaptive networks or formations of vehicles are a potential reality. However, these vehicles lack the strap-down sensors used on more conventional vehicles to improve navigation, including the DVL and high-end AHRS. The limited payload space and on-board power of these small vehicles mean that adding these devices would be impractical or impossible due to their size, the increase in per-vehicle cost, or the impact on vehicle runtime.

The primary motivation of this work is to enable the democratization of underwater vehicle technology by providing a low-cost navigation suite that is well suited to this new generation of inexpensive and miniature underwater vehicle. A secondary motivation is to enable multi-AUV deployments using the same navigation suite. In order to achieve this, the system must be scalable, low-power and low-cost, and ideally easy to deploy. By providing such a navigation suite, we hope to work towards the more ubiquitous use of underwater robots, both by making these vehicles more widely available and accessible, as well as enabling multi-vehicle deployments that are more user-friendly.

1.5 Contributions

The major contribution of this thesis is the design and development of a low-cost, low-power, self-contained acoustic navigation system ideally suited for the new generation of inexpensive and miniature underwater robot. The system is distinguished by the use of a single acoustic navigation beacon, which periodically broadcasts a signal, and a passive receiver array mounted on the vehicle, which is time-synchronized to the beacon and processes the broadcast signal to determine range and angle to the navigation beacon. The navigation system has three defining characteristics: (i) the passive nature of the receiver and the use of a single active beacon means that the system is scalable to an arbitrarily large number of vehicles, all of which can use the system to navigate concurrently; (ii) the passive receiver and single beacon allow the system to be low-cost and low-power, preserving vehicle run-time and maintaining the vehicle's base price; and (iii) use of a single beacon makes the system easy to deploy and intuitive to use. The passive nature of the ultra-short baseline (USBL) array mounted on each vehicle has led us to describe this method of acoustic navigation as Passive Inverted Ultra-Short Baseline (piUSBL) navigation.

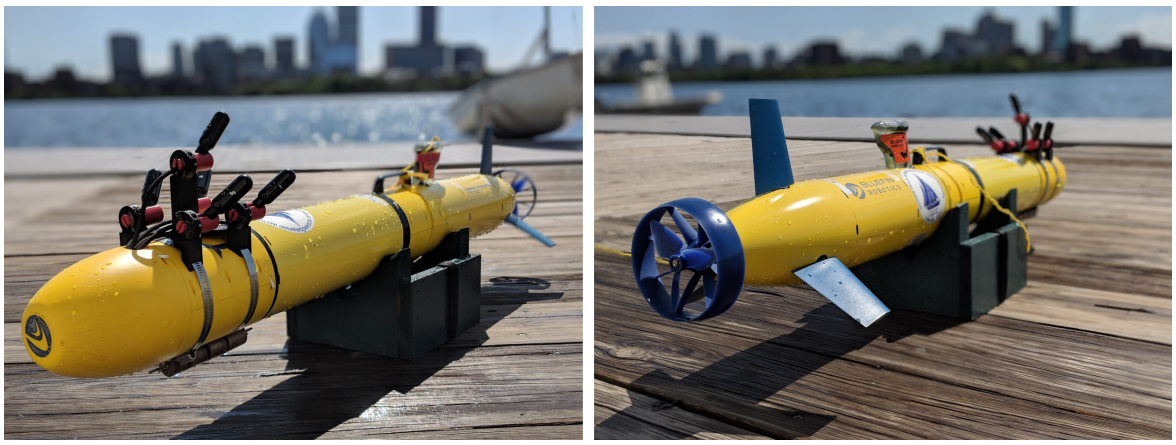

Figure 1.2: One of our commercial Bluefin SandShark autonomous underwater vehicles (AUVs), *Quokka*, outfitted with our low-cost navigation system.

The piUSBL navigation system has been implemented on a fleet of three small and low-cost Bluefin SandShark AUVs, as pictured in Fig. 1.2, each of which is 12.4 cm in diameter, about 105 cm long, and weighs around 12 kg. Closed-loop, online navigation with this system is performed fully on-board each vehicle, using an embedded, single-board computer. Repeated deployments of all three AUVs has demonstrated the system's ability and robustness in enabling simultaneous multi-AUV navigation, validated with over 36 hours of vehicle runtime over multiple experiments.

This thesis also presents secondary contributions in array processing and multi-AUV operations. An efficient method for direction of arrival (DOA) estimation has been developed via beamforming on pairs of array elements, which significantly reduces computation time and memory use for small arrays. A novel operational paradigm, which we term *relative navigation*, has also been developed, in which vehicle behaviors are designed within a *beacon-centric* frame of reference, and switching between these behaviors is made possible by the transmission of different signals from the beacon. Experiments using this operating paradigm have validated its utility as a means of intuitively and easily commanding and controlling a fleet of AUVs.

1.6 Thesis Organization

The remainder of this manuscript is divided into the following six chapters:

- **Chapter 2 – Passive Inverted Ultra-Short Baseline Positioning: *The Prototypical System*:** This chapter introduces the system design of Passive Inverted Ultra-Short Baseline (piUSBL) navigation, outlines the major technical concepts that underly its operation, and details the prototype implementation of the system on a low-cost,

miniature AUV. Initial results of the navigation approach are presented from experimental deployments of the prototype system, and reflections and lessons learned from these experiences are detailed, informing subsequent improvements to the system.

- **Chapter 3 – Improving Processing Speed: *The Sequential Monte-Carlo Beamformer*:** This chapter introduces an improvement to the Bayesian processing approach used to fuse acoustic and odometry information in the piUSBL navigation system. By tightly coupling the conventional beamformer (CBF) and the particle filter, real-time, closed-loop navigation is made possible. Experimental results of closed-loop navigation are provided using our prototype system, as well as with an implementation of the system on a Bluefin-21 AUV, demonstrating its utility for expensive and large conventional AUVs. The concept of relative navigation is introduced with a demonstration of the system enabling the Bluefin-21 to home-in on a moving beacon.

- **Chapter 4 – Improving Measurement Precision: *Element Pair Decomposition Beamforming*:** This chapter introduces a novel direction of arrival (DOA) estimation method called element pair decomposition (EPD) beamforming. By constructing the direction measurement via beamforming on pairs of elements in the array along coning angles, we are able to substantially reduce computation time and memory use, which consequently enables us to drastically improve the resolution of the acoustic direction measurement. The development of EPD beamforming is detailed, and its properties in terms of accuracy, speed and memory usage against the CBF are analyzed.

- **Chapter 5 – Testing the Pipeline: *System Verification and Evaluation*:** This chapter provides a comprehensive evaluation of the entire piUSBL system stack with an implementation of the system on a autonomous surface vehicle (ASV). Comparison of the positioning solution from piUSBL against differential GPS (DGPS) allows us to analyze and verify its accuracy, providing us with a quantitative measure of its ability to localize underwater vehicles. The implementation of the piUSBL navigation suite on our fleet of three commercial Bluefin SandShark AUVs is also provided, along with a description of the calibration procedures required.

- **Chapter 6 – Experiments with Multiple Autonomous Underwater Vehicles: *Relative Navigation using piUSBL*:** This chapter details the novel operational paradigm of relative navigation, including the method of commanding various vehicle behaviors via the transmission of different acoustic signals, and the method of fleet-wide control via the movement of the beacon. Extensive experimental results are provided of multi-AUV deployments, demonstrating the navigational ability of the piUSBL system. Proof-of-concept multi-AUV missions are also described.

- **Chapter 7 – A Vision of Ubiquity: *Closing Remarks and the Future*:** This chapter concludes this manuscript, providing the reader with a summary of the contributions of this thesis, as well as the direction of ongoing and future work.

Chapter 2

Passive Inverted Ultra-Short Baseline Positioning: *The Prototypical System*

2.1 Introduction

Passive Inverted Ultra-Short Baseline (piUSBL) positioning uses well-established technological elements and data processing techniques. In December of 2015, Bluefin Robotics in partnership with the Defense Advanced Research Projects Agency (DARPA) provided the Massachusetts Institute of Technology (MIT) Laboratory for Autonomous Marine Sensing Systems (LAMSS) with a prototype low-cost, miniature autonomous underwater vehicle (AUV) as part of the DARPA ADAPTable sensor system (ADAPT) program; (a modified design of this AUV would eventually go on to become the commercially available Bluefin SandShark micro–AUV). With the availability of this platform and necessary technological components, we designed a payload for this vehicle and built it in close cooperation with Bluefin; this prototypical system provided the first practical demonstration of piUSBL positioning, establishing the feasibility and utility of this approach for navigating this newly emerging class of low-cost, low-power, miniature underwater vehicle. In this chapter we detail the prototypical hardware and software elements that are fundamental toward enabling piUSBL positioning, provide results from experiments conducted using this prototype, and walk through the insights gained from these results – insights that inspired the necessary subsequent system modifications that allowed us to demonstrate robust and accurate navigation using piUSBL on a fleet of three commercial SandShark AUVs almost three years later.

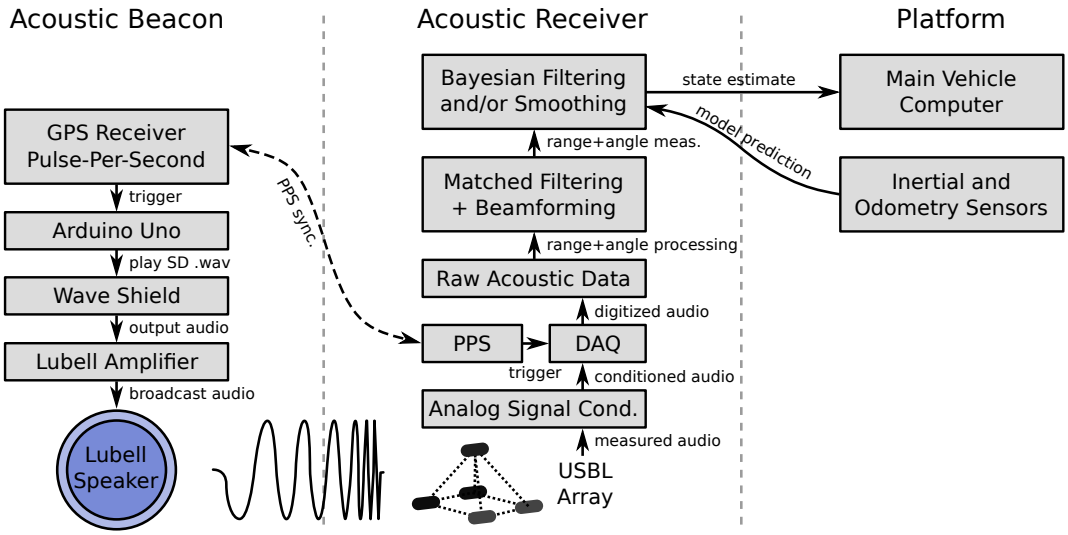

Figure 2.1: General system diagram of piUSBL positioning – the acoustic beacon periodically broadcasts a stored acoustic signal; the acoustic receiver has an accurate clock that enables the synchronous recording of this signal, which is then processed to obtain range and angle measurements that are ingested by a Bayesian filter and/or smoother.

2.2 System Design

The piUSBL system has two fundamental elements: the acoustic beacon, which periodically broadcasts an acoustic signal, and the acoustic piUSBL receiver, which *synchronously* measures and records this acoustic signal. The piUSBL receiver has three characteristics that distinguish piUSBL positioning from other approaches to acoustic positioning: first, the receiver must be *time synchronized* to the acoustic beacon; second, the receiver must contain an *ultra-short baseline (USBL) array* of *passive* acoustic sensors; and third, the receiver must be *vehicle-mounted*. The first two characteristics allow the vehicle to *passively* measure the one-way travel-time (OWTT) between the beacon and the AUV, and the third characteristic is the *inversion* of the prevailing USBL paradigm in which a USBL transceiver is mounted on the topside platform rather than subsea platform[1] – the name *Passive Inverted Ultra-Short Baseline* positioning is thus a reference to the passive and inverted nature of the acoustic receiver. As a result of its design and as previously mentioned, the piUSBL system has two substantial advantages over existing acoustic positioning systems: (i) the use of a single beacon reduces system cost and significantly improves ease of deployment; and (ii) the completely passive nature of the receiver reduces power use and cost, and enables the navigation of an arbitrarily large number of underwater vehicles using just one beacon.

[1]The terms 'topside' and 'subsea' are standard in underwater and marine operations; 'topside' refers to platforms above the water, such as a deployment ship, mooring or dock; 'subsea' refers to underwater assets, such as a towfish or underwater vehicle.

The components of our piUSBL system are outlined in Fig. 2.1, generalized for use on an arbitrary robotic platform. In brief, the system flow is as follows: the acoustic beacon broadcasts a user-defined acoustic signal into the water when triggered by the rising edge of the pulse-per-second (PPS) signal from a global positioning system (GPS) receiver; simultaneously, the piUSBL receiver is triggered to record acoustic measurements sensed by an array of hydrophones using a digital acquisition device (DAQ) – triggering of this DAQ is performed using a PPS signal that has been synchronized to the beacon prior to vehicle deployment, using either a highly-accurate clock, or, if a surface tether is available, another GPS receiver; digitized acoustic data is then processed to obtain range and angle measurements via matched filtering and beamforming; finally, acoustic range and angle is fused with vehicle odometry and inertial measurements using a recursive Bayesian filter or smoother to generate an estimate of vehicle state, which is made available to the main vehicle computer for feedback control. Note that the system is platform agnostic, and components may be substituted without altering the spirit of the piUSBL approach – as we mentioned, its defining characteristic is the use of OWTT range and angle from the platform to the beacon.

2.3 Hardware

Throughout this thesis, we make use of a number of different platforms, including miniature and large AUVs, and an autonomous surface vehicle (ASV), each of which are outfitted with a piUSBL receiver. The piUSBL system has slightly different specifications for each of these vehicles, but each specific implementation shares many commonalities; system configurations specific to each platform are introduced within the chapters in which they are used, but for the sake of convenience common system elements are outlined here.

2.3.1 Acoustic Beacon

The acoustic beacon used by piUSBL comprises of five components, shown on the left of Fig. 2.1. The first is a GPS receiver (our beacons use either a Garmin GPS 18x LVC puck[2], or an Adafruit Ultimate GPS Breakout Version 3[3]), which outputs a GPS PPS signal. This PPS signal is monitored by a digital pin on an Arduino Uno micro-controller[4], allowing it to detect the rising edge of the signal and command the playback of a user-defined WAV signal stored in an SD card on an attached Adafruit Wave Shield[3]. The output jack of the Wave Shield is connected to a Lubell 3400 60 W amplifier[5], which amplifies the acoustic signal and broadcasts it into the water via a Lubell LL916C underwater speaker[5] (alternatively, the less

[2]https://www.garmin.com/en-US/
[3]https://www.adafruit.com/
[4]https://www.arduino.cc/
[5]http://www.lubell.com/

powerful Lubell UW30PA 30 W amplifier and UW30 speaker can be used). The result is a custom-designed and built underwater acoustic beacon that periodically broadcasts a user-defined acoustic signal at a rate of 1 Hz with a jitter of less than 1 ms (qualitatively measured using an oscilloscope). Very consistent timing on the firing of the beacon is necessary in order to obtain precise range measurements – a 1 ms delay translates to an error in range of approximately 1.5 m; although this jitter can be improved through custom designed circuitry, the Arduino Uno and Wave Shield provide an accessible and low-cost electronics design for the beacon. The Arduino code for our beacon is publicly available[6].

2.3.2 Passive Inverted Ultra-Short Baseline Receiver

Although the operating principle of the piUSBL receiver is identical on each platform, the hardware that the passive acoustic receiver consists of differs slightly by vehicle. As illustrated in the center of Fig. 2.1, the piUSBL receiver passively detects the broadcast acoustic signal and enables the vehicle to instantaneously determine the range and angle to the beacon in the vehicle body-fixed frame (BFF). The receiver includes a USBL array, consisting of multiple High Tech Inc. HTI-96-Min hydrophones[7] with current-mode pre-amplifiers. This array is rigidly attached to the vehicle, and the acoustic energy it measures is converted into a voltage signal and filtered using a passive resistor-capacitor bandpass circuit ($10 \leq f_c \leq 160 \times 10^3$ Hz), before being converted into a digital signal using a Measurement Computing USB-1608FS-Plus DAQ[8]. In order to synchronously start the digital conversion in sync with the broadcasts of the beacon, the DAQ is triggered to record using the rising edge of a PPS signal on-board the vehicle that has been synchronized to the beacon PPS signal prior to launch. This synchronization is achieved in one of two ways: if the platform has access to the surface (in the case of an ASV or if there is a tether), then the DAQ is simply triggered using the PPS signal from a GPS receiver, as in the case for the beacon; if the platform is an untethered AUV, then a highly accurate and precise chip-scale atomic clock (CSAC) is employed to generate a synchronized PPS signal – we use the Microsemi Sa.45s CSAC[9] integrated on the Jackson Labs CSAC GPSDO module[10]. These CSACs will typically drift by less than 100 µs per 24 h in holdover, assuring accurate synchronization, with a jitter of approximately 80 ps. The DAQ is programmed to record a user-defined number of samples of acoustic data (allowing the user to dictate maximum sensing range) at a user-specified sampling rate from each element of the array after each triggering event, with this raw acoustic data made available to an on-board Raspberry Pi 3 computer[11]. When combined with vehicle attitude (orientation) from

[6] https://github.com/nicrip/Lubell/
[7] http://www.hightechincusa.com/
[8] https://www.mccdaq.com/
[9] https://www.microsemi.com/
[10] https://jackson-labs.com/
[11] https://www.raspberrypi.org/

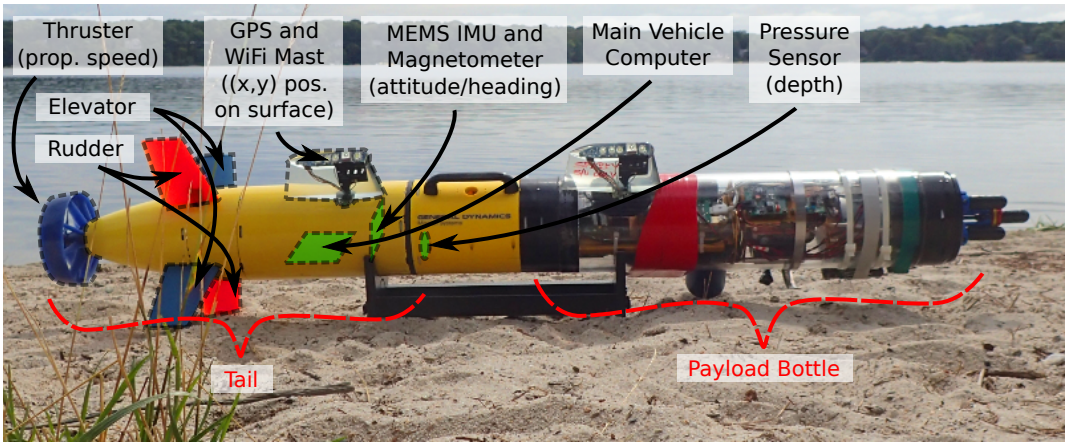

Figure 2.2: Prototype Bluefin SandShark AUV outfitted with our piUSBL payload – the rear third of the AUV is the standard SandShark platform (the tail), which consists of a single magnetically-coupled propeller and motor, two stepper motors for elevator and rudder fin control, a pressure sensor for depth, a GPS and WiFi receiver, and a low-grade 9-axis MEMS IMU with magnetometer.

compass and inertial sensors on the platform, acoustic range and angle measurements in the body-fixed frame (BFF) can be used to estimate the relative (x, y) position of the beacon in the vehicle-carried (vehicle-centered) East-North-Up (ENU) local-level frame; if the position of the beacon is known in the global frame, the vehicle can then be localized in the global frame of reference. Consequently, it should be noted that the accuracy of piUSBL localization has some dependence on the accuracy of the vehicle's inertial sensors and compass.

2.3.3 Prototype Bluefin SandShark Configuration

The prototypical piUSBL system was implemented on a prototype low-cost, miniature AUV called the SandShark from Bluefin Robotics[12], provided to MIT LAMSS as part of the DARPA ADAPT program. This vehicle is illustrated in Fig. 2.2, with the AUV sectioned into two distinct parts: the standardized tail section as the rear third of the vehicle, provided by Bluefin; and the front two-thirds of the vehicle which is the custom payload outfitted with our piUSBL receiver.

Platform

The rear third of the AUV is the prototype SandShark platform, which is equipped with a number of sensors and actuators as labeled in Fig. 2.2. It consists of: (i) a single magnetically-coupled propeller and motor, providing forward thrust to the vehicle and a measure of propeller rotations-per-minute (RPM); (ii) two stepper motors, each controlling a pair of elevator

[12] https://gdmissionsystems.com/underwater-vehicles/bluefin-robotics/

(blue in Fig. 2.2) and rudder (orange in Fig. 2.2) control surfaces, allowing the vehicle to be controlled in pitch and yaw; (iii) a GPS and WiFi receiver in the mast, providing a global position solution and communications when the vehicle is on the surface; (iv) a low-grade 9-axis micro-electro-mechanical systems (MEMS) inertial measurement unit (IMU) with magnetometer, which provides an estimate of vehicle attitude and heading; (v) a pressure sensor to estimate vehicle depth; and (vi) a Linux main vehicle computer.

AUVs typically measure speed-over-ground using a Doppler velocity log (DVL), which measures the Doppler shift from the sea-floor return of a transmitted acoustic signal; unfortunately, DVLs are expensive and power-hungry, leading Bluefin to omit using one for speed measurements on the SandShark AUV. Instead, a linear mapping of propeller RPM to speed, followed by pitch (β) compensation, is used to provide a direct estimate of AUV speed-over-ground:

$$v_{sog} = RPM \cdot 9.125 \times 10^{-4} \cdot \cos(\beta) \tag{2.1}$$

This linear transform was obtained by Bluefin from experimental measurements of the RPM-to-speed mapping from deployments using initial SandShark prototypes; as expected, this estimate of speed is highly inaccurate, especially when compared to DVL speed measurements, and contributes significantly to the inaccuracy of the internal dead-reckoning solution shown in the results section of this chapter.

Attitude and heading on conventional AUVs are typically measured using high-grade IMUs and gyroscopes, which are integrated into a single attitude and heading reference system (AHRS). Due to the high cost of a conventional AHRS, the prototype SandShark instead makes use of a low-grade 9-axis MEMS IMU and magnetometer, whose measurements are fused using the direction cosine matrix (DCM) algorithm [161] to provide a low-cost AHRS for the vehicle. The low level of accuracy of this magnetometer-based AHRS also contributes to the inaccuracy of the dead-reckoning solution, and is additionally affected by magnetic anomalies.

Attitude and speed data from the MEMS AHRS and propeller are used by the tail for feedback control on depth, pitch and heading set-points. The platform uses simple proportional-integral-derivative (PID) controllers to drive the AUV to these desired set-points, with gains tuned by Bluefin over a number of vehicle deployments (note that there is no PID controller for speed, since it is a direct mapping of propeller RPM). AUV speed, heading and depth set-points are commanded from the payload Raspberry Pi 3 computer using the MOOS-IvP [156] autonomy framework, which allows us to plan and execute autonomous vehicle missions and behaviors. Vehicle speed and attitude information is also made available to the payload computer. This separation of control and command between the main vehicle computer and

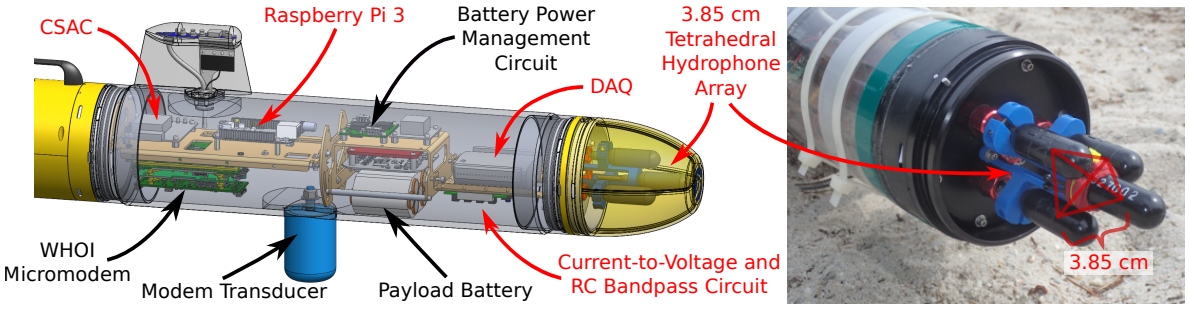

Figure 2.3: CAD diagram of prototype Bluefin SandShark payload with piUSBL receiver – the components of the receiver are labeled in red, while additional components that are not directly used for piUSBL positioning are labeled in black; a close-up photo of the 3.85 cm tetrahedral hydrophone array is shown on the right.

the payload computer is known as the *frontseat-backseat* paradigm, and enables the user to abstract away low-level vehicle control (handled by the frontseat, or main vehicle computer) from high-level behavior and mission commands (handled by the backseat, or payload computer). In the prototype SandShark AUV all sensor drivers and vehicle programs were implemented using the Robot Operating System (ROS) middleware [162]. Note that although the lack of a DVL and a high-grade AHRS significantly reduces the accuracy of dead-reckoning, it importantly allows for the vehicle to be inexpensive and small, with a 12.4 cm diameter and a total length of about 115 cm.

Payload

The payload for the prototype SandShark AUV makes up the front two-thirds of the vehicle and includes our prototypical piUSBL receiver, consisting of the five main hardware components outlined in subsection 2.3.2, and labeled in red in Fig. 2.3. Its USBL array consists of four hydrophones mounted on the nose of the vehicle (right of Fig. 2.3) in a tetrahedral configuration, with the distance between the centers of each pair of hydrophones being 3.85 cm. Acoustic energy captured by this array is converted into a voltage signal and bandpass filtered using analog circuitry, then digitized using a DAQ. The payload computer clock is synchronized to GPS PPS using an on-board CSAC, which is also used to trigger the DAQ. This digital acoustic data is then made available to the payload computer via a universal serial bus (USB) connection, where it can be processed. The DAQ is configured to record 8000 samples from each hydrophone every second, at a sampling rate of 37.5 kS/s – assuming a sound speed in freshwater of 1481 $\mathrm{m\,s^{-1}}$, this gives our system an effective range of about 316 m:

$$r_{max} = \frac{c}{Fs} \cdot n = \frac{1481\ \mathrm{m/s}}{37500\ \mathrm{S/s}} \cdot 8000\ \mathrm{S} \approx 316\ \mathrm{m} \tag{2.2}$$

Note that although the prototypical implementation of the piUSBL system is limited in its current configuration to a range of 316 m, it can be extended to a range of up to 1 km by configuring the DAQ to record a larger number of samples, and the system can be modified further to extend the range to multiple kilometers.

The payload includes additional components that are not part of the piUSBL system – a battery and power management circuit that provides power to the payload; and a WHOI Micromodem 2[13] underwater acoustic modem configured to operate with a BTech[14] BT-28UF 28 kHz transducer. This acoustic modem and transducer are used as part of a separate research project in underwater communications with low-cost AUVs; however, it can also be used in conjunction with Micromodem-based LBL beacons to provide an independent set of vehicle position estimates for validation of our piUSBL system. It is important to note that the Micromodem is not used for piUSBL positioning.

Receiver USBL Array and Source Signal

The design of the USBL array for the prototypical system took into consideration a number of factors. Firstly, the number of elements required for an array to obtain an *unambiguous* estimate of direction in 3D (azimuth and inclination): this requires a minimum of four elements, since any three elements will define a 3D plane which introduces an ambiguity similar to the left-right ambiguity experienced by a line array (i.e. given an acoustic source positioned in one of the half-spaces defined by the 3D plane, it is impossible to use the time delays experienced by each element of the array to disambiguate it from its 'mirrored', 'virtual' source in the other half-space). Secondly, selecting a suitable array geometry given a minimum of four elements: since we do not wish to bias its performance in specific directions, we decided upon a regular tetrahedron, a shape that is spherically symmetrical – this means that the beampattern of the array has a main lobe width that is nearly consistent regardless of the direction in which it is steered. An illustration of the theoretical 3D beampatterns of a regular tetrahedral array with an acoustic plane-wave incident from various directions is shown in Fig. 2.4. We can see from this illustration that the main lobe has a consistent shape and beamwidth regardless of the direction of the incoming acoustic energy (note that these beampatterns are generated with an acoustic source signal set to a frequency such that the inter-element spacing of the array is half the wavelength of the source signal).

This consistency in main lobe beamwidth is quantified in Fig. 2.5, which illustrate the horizontal and vertical 3dB (half-power) beamwidths of the main lobe as the azimuth and inclination of the incoming acoustic plane wave is varied (the wavelength of the incoming plane wave is set to double the array inter-element spacing). It is apparent that the horizontal and

[13] https://acomms.whoi.edu/micro-modem/
[14] http://btechacoustics.com/

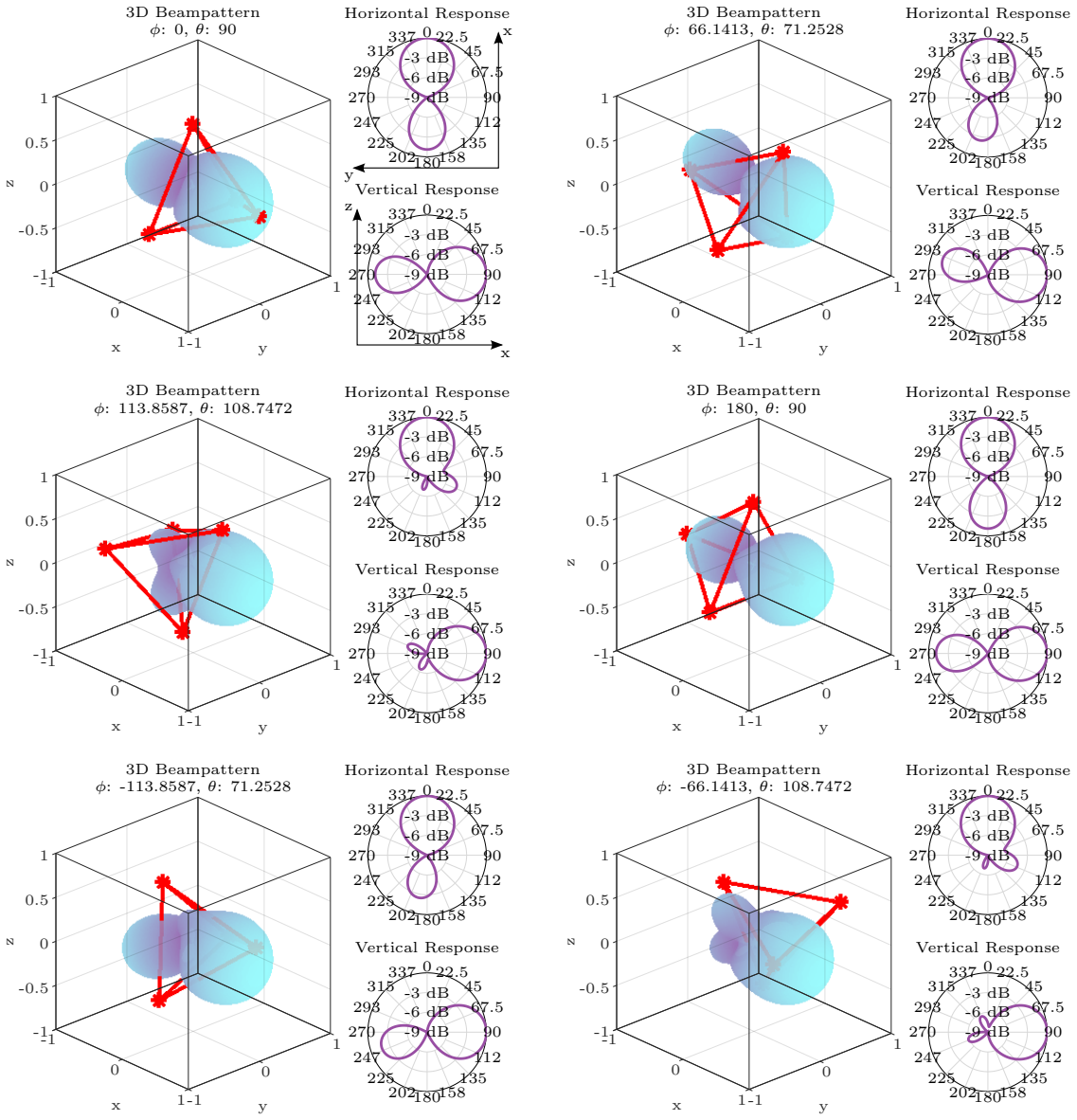

Figure 2.4: 3D beampatterns of a four element regular tetrahedral array – beampatterns are shown for an incoming plane wave whose wavelength is double the array inter-element spacing (i.e. half-wavelength array spacing); regardless of the direction (azimuth and inclination, ϕ and θ) of the incoming plane wave, the beampattern has a near-consistent main lobe width; it is apparent that the 3dB (half-power) beamwidth remains fairly consistent in both the horizontal and vertical directions, a property resulting from its spherical symmetry.

vertical beamwidths remain fairly consistent between 81° and 84° regardless of the direction of the incoming acoustic energy. Thus, this regular tetrahedral array design is not biased toward energy coming from any particular direction.

Once the geometry of the array was decided upon, the next decision to make was the size

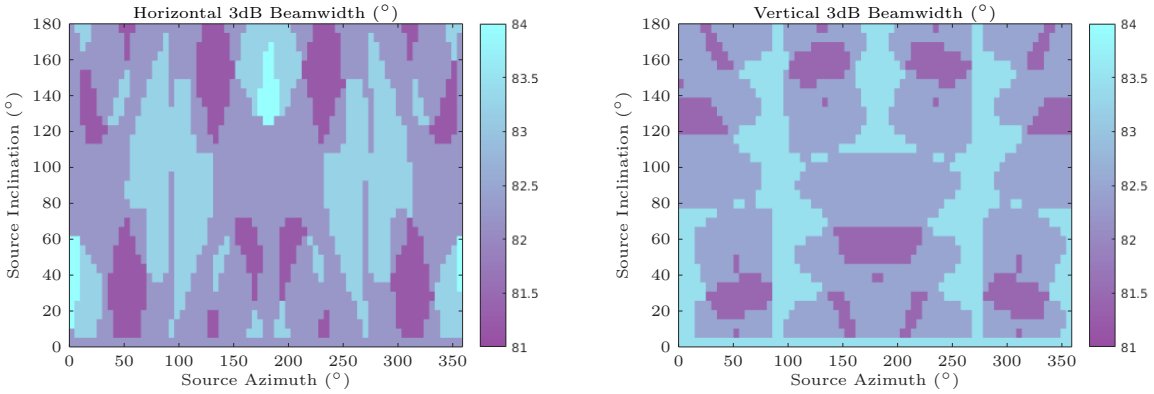

Figure 2.5: Horizontal and vertical 3dB beamwidths of a four element regular tetrahedral array – as the incoming acoustic plane wave is varied in azimuth and inclination (incoming direction), the shape of the main lobe changes slightly; the left plot shows the 3dB (half-power) beamwidth in the horizontal plane, while the right plot shows the 3dB (half-power) beamwidth in the vertical plane; the beamwidths remain fairly consistent (between 81° and 84°) regardless of the direction of the incoming acoustic energy. Note that the source wavelength is set to double the array inter-element spacing.

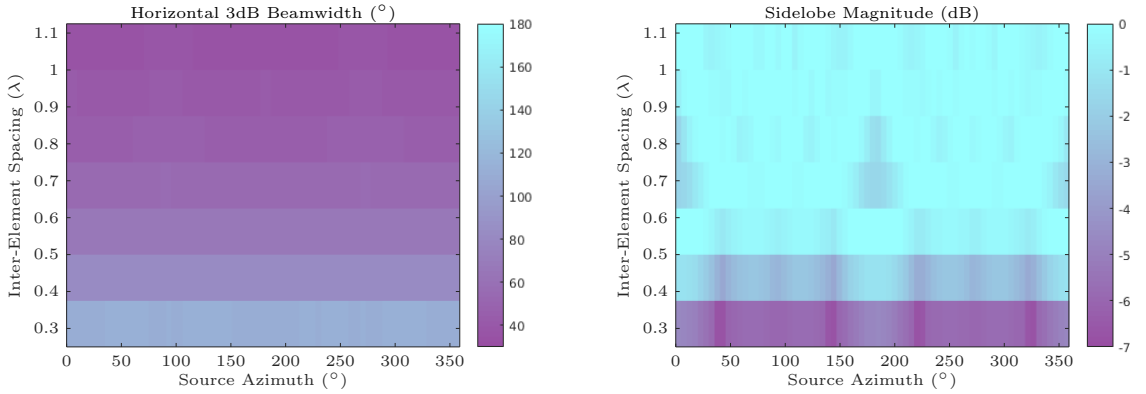

Figure 2.6: Horizontal 3dB beamwidths and sidelobe magnitudes of a four element regular tetrahedral array – as the wavelength (λ) of the incoming acoustic plane wave is reduced (frequency is increased), the 3dB (half-power) beamwidth of the main lobe is reduced, thereby improving the angular resolution of the array; however, doing so increases the magnitude of the sidelobe, thereby introducing acoustic noise from unwanted directions; thus, the choice of inter-element spacing (or source operating frequency) is a trade-off between these two characteristics. Note that the inclination of the source is set to a constant 90° in these plots.

of the array. Since the array was to be mounted at the nose of the vehicle in order to facilitate an easy mounting scheme and to reduce the effect of the array on vehicle hydrodynamics, the diameter of the array had to be less than the 10 cm diameter of the payload end-plate; in addition, since we wanted to minimize any effect on vehicle hydrodynamics, we wanted to be able to enclose the array using the pre-existing SandShark hollow nosecone, as illustrated in

Fig. 2.3. These constraints resulted in the selection of a tetrahedral edge length of 3.85 cm.

The size of an array and the frequency of the acoustic signal incident upon it have a close coupling, and determine the array's overall response. To understand these effects on the array response, we can combine these two parameters and talk about the response characteristics by discussing it in terms of the array inter-element spacing in wavelengths (λ). Thus, if we increase the inter-element spacing, this is effectively identical to either: (i) increasing the physical size of the array, or (ii) increasing the frequency (decreasing the wavelength) of the acoustic signal. Varying the inter-element spacing has two major effects on the response of the array: (i) it changes the beamwidth of the main lobe, and (ii) it changes the magnitude of the sidelobes[15], with these two effects occurring simultaneously. In Fig. 2.6 we illustrate how the mainlobe 3dB beamwidth (in the horizontal plane) varies with changes to the inter-element spacing, as well as how the sidelobe magnitude varies. It can be seen that there is a trade-off between the two – as inter-element spacing increases (or frequency increases), the main lobe beamwidth is reduced, thereby improving the angular resolution of the array response; however, at the same time the sidelobe magnitude increases, thereby introducing acoustic noise from directions other than the main lobe, with the potential of reducing accuracy in the array response.

Since we already selected the size of the tetrahedral array, in order to balance this trade-off we had to carefully select the operating frequency of the acoustic signal broadcast by our beacon. Possible frequencies were limited by the rated frequency response of Lubell underwater speaker, which is 0.5–21 kHz, as well as the range of frequencies that the Wave Shield can play, which include audio files up to a sampling rate of 22 kHz. Examining Fig. 2.6, it is apparent that at inter-element spacings greater than 0.5λ, the magnitude of the sidelobe comes within 2dB of the main lobe, an effect which we wish to avoid; in addition, the maximum operating frequency of our beacon is about 20 kHz, which, for our 3.85 cm array, corresponds to a maximum inter-element spacing of:

$$0.0385\ \text{m} := x \cdot \lambda \qquad \text{AND} \qquad \lambda := \frac{c}{f} \tag{2.3}$$

$$0.0385\ \text{m} \cdot \frac{1}{x} = \frac{c}{f} \tag{2.4}$$

$$x = \frac{f}{c} \cdot 0.0385\ \text{m} \tag{2.5}$$

$$x = \frac{20000\ \text{Hz}}{1481\ \text{m/s}} \cdot 0.0385\ \text{m} = 0.52\lambda \tag{2.6}$$

[15]The main lobe is the lobe in the beampattern containing the highest power, while sidelobes are other local maxima – in Fig. 2.4, the main lobe in the first plot can be clearly seen pointing directly along the positive x–axis, while a sidelobe can be seen pointing directly opposite along the negative x–axis. When a sidelobe is the same magnitude as the main lobe, it is called a grating lobe.

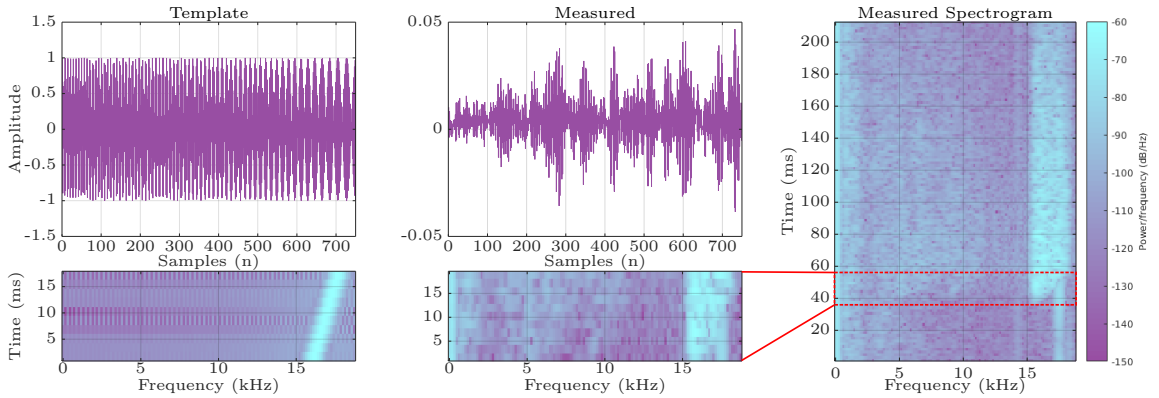

Figure 2.7: Ideal and measured 20 ms, 16–18 kHz LFM up-chirp – the ideal up-chirp is shown on the top-left over time (750 samples or 20 ms), and as a spectrogram in the bottom-left; the corresponding up-chirp as measured by the first element of the USBL array is shown in the top-center over time, and as a spectrogram in the lower-center; the spectrogram of the full sample of measured acoustic data (8000 samples or 213 ms) is shown on the right.

We therefore selected an inter-element spacing of 0.4λ–0.5λ, settling upon an operating frequency of 17 kHz, or about a 0.44λ inter-element spacing.

The final design choice that had to be made in terms of the array and the source signal was the actual design of the broadcast acoustic signal. An obvious choice was a linear frequency modulation (LFM) chirp, which, from radar research, has long been well known to have advantageous pulse compression properties, with improved performance in terms of signal-to-noise ratio (SNR) and range resolution [163]. For simplicity, we selected a LFM up-chirp with a bandwidth (Δf) of 2 kHz centered around 17 kHz and a duration (T) of 20 ms, resulting in a 20 ms, 16–18 kHz LFM up-chirp. The pulse compression ratio (PCR) is calculated as:

$$PCR = T \cdot \Delta f \tag{2.7}$$

$$= 20 \times 10^{-3}\ \text{s} \cdot 2000\ \text{Hz} = 40 \tag{2.8}$$

This PCR can be understood as an amplification or gain of the SNR; thus an increase in either the duration or bandwidth of the broadcast signal has the potential to improve the SNR of our system – however, these selected values lie well within the known capabilities of our acoustic beacon and so were selected for our experiments. The theoretical range resolution (S_r) of our up-chirp can be calculated according to:

$$S_r = \frac{c}{2 \cdot \Delta f} \tag{2.9}$$

$$= \frac{1481 \text{ m/s}}{2 \cdot 2000 \text{ Hz}} = 0.37 \ m \tag{2.10}$$

Recall in subsection 2.3.1 that our acoustic beacon has a jitter on its transmission bounded by 1 ms, which corresponds to an error in range of approximately 1.5 m; as such, this selected LFM signal design is more than adequate for our purposes, and does not represent a limitation on the accuracy of our system. The designed ideal LFM up-chirp that is transmitted by the beacon in our prototypical system is shown on the left in Fig. 2.7, and the in-water up-chirp signal measured by the first element of the SandShark USBL array during a deployment is shown in the center and the right of Fig. 2.7.

2.3.4 The Importance of Accurate Timing

Accurate timing is a linchpin component that enables piUSBL positioning. An accurate time source on-board the vehicle enables OWTT *passive* ranging between the acoustic beacon and the piUSBL receiver. This is crucial because it enables an arbitrary number of vehicles, each equipped with a piUSBL receiver, to self-compute an instantaneous localization fix of itself relative to the acoustic beacon, at every timestep. This makes piUSBL positioning inherently scalable, easily deployable, and introduces the possibility of a novel operational paradigm: instantaneous position fixes enable a *relative* navigation paradigm, in which AUV control, planning and navigation are executed in a beacon-relative frame of reference (i.e. a reference frame whose origin is defined by the acoustic beacon); using relative navigation, the beacon itself can be moved in order to alter the motion of the vehicles, effectively providing a centralized means of fleet-wide control. This is in contrast to global navigation, where vehicles must plan and navigate within a static, fixed frame of reference. Further work along this direction of relative navigation can be found in chapter 6 of this thesis.

Although accurate on-board timing is critical for piUSBL positioning, it carries with it a secondary (but no less important) advantage. With a vision of a future where underwater robots are ubiquitous (in terms of both greater access and numbers), new and previously unattainable operational concepts that make use of a large number of vehicles could become a reality. For example, multi-AUV deployments could be undertaken to sample mid-water oceanographic phenomena that are highly dynamic in both space and time; each AUV in a fleet would be able to sample the phenomena from a specific point in space over a duration of time – by combining the temporally varying measurements from all AUVs, we would be able to infer the spatiotemporal dynamics of the oceanographic phenomena, allowing us to model how it varies spatially over time. However, in order to do this, the data streams measured

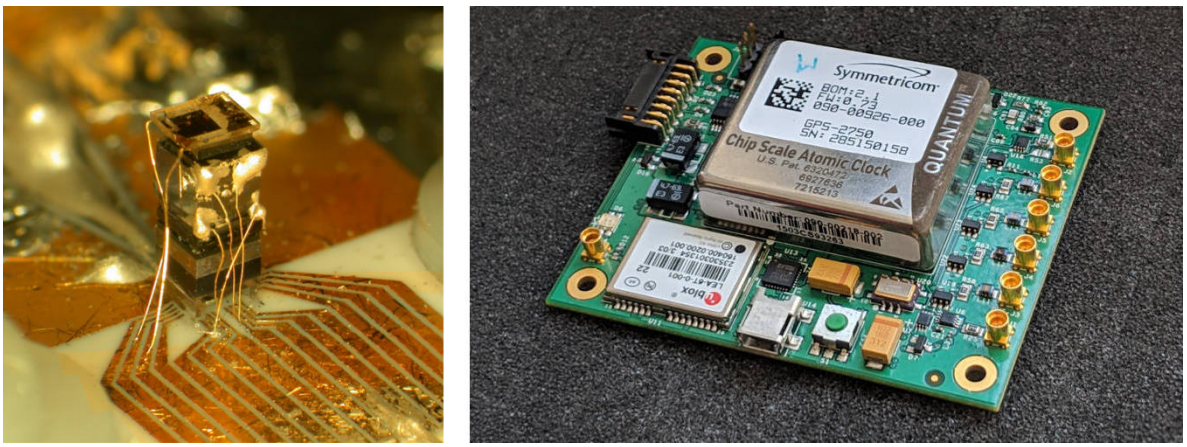

Figure 2.8: *Left*: Photograph of the original prototype miniature atomic clock from NIST (courtesy of NIST). *Right*: a commercial Jackson Labs chip-scale atomic clock module with Symmetricom (Microsemi) Sa.45s CSAC.

by multiple independent vehicles must be *synchronized or time-aligned* to a shared starting point – without doing so, the ability to infer how the phenomena as a whole changes with time becomes significantly more challenging. Strategies can be used, such as constraining the spatiotemporal dynamics to an assumed prior model, or estimating clock drift between two endpoints (clock offsets measured against GPS during surfacing events [165]), but these strategies are fraught with trade-offs, since these processes are non-stationary. On the other hand, time synchronization of all AUVs in a fleet explicitly enables independent measurements to be combined, since all vehicles share a common time reference. This highlights the coupled and critical importance of accurate timing for multi-AUV operations: accurate timing enables piUSBL positioning, which in turn drastically improves the feasibility of multi-AUV deployments (through cost-effectiveness and scalability); and once multi-AUV deployments are feasible, accurate timing plays an equally important role in enabling novel multi-AUV operations by allowing fleet-wide independent measurements to be merged into a common time reference.

Accurate timing in GPS-denied environments has only recently been made possible with the emergence of an enabling technology in the form of the chip-scale atomic clock (CSAC). The development of CSAC technology grew out of the desire to improve satellite ranging measurements for GPS and global navigation satellite system (GNSS) [164]. Position fixes produced by these systems are directly informed from the trilateration (or multilateration) of range from satellite signals, which is derived by accurately measuring their propagation time. Since GPS and GNSS use OWTT ranging, their performance is directly tied to clock performance of the transmitter and receiver, both of which must be synchronized. GNSS receivers typically use a temperature compensated crystal oscillator (TCXO), which exhibit

noisy short-term stability and poor long-term stability performance, and which are regularly time-disciplined against the satellites atomic clocks to prevent drift. As such, when these TCXOs lose satellite signal, they drift at a rate of 1 s/day; these make them especially inadequate for underwater time synchronization and ranging, since a drift rate of this magnitude corresponds to a drift in range of about 1500 m/day, or about 1 m/minute. The initial technological development of the CSAC grew out of the DARPA CSAC program [166], from an initial feasibility demonstration of a rudimentary miniature-scale atomic clock physics package developed at the National Institute of Standards and Technology (NIST) in 2002 (Fig. 2.8, left). The operating principle of the CSAC is based on that of classical atomic clocks, but it achieves enormous reductions in size and power consumption through the use of modern MEMS technologies. Atomic clock oscillators make use of the electronic transition frequency in the microwave region of the electromagnetic spectrum of atoms, and they typically use rubidium, hydrogen or cesium. The first commercially available CSAC, known as the Sa.45s, was released in 2011 by Microsemi (formerly Symmetricom), and is based on cesium (Fig. 2.8, right). A feedback loop locks onto the electronic transition frequency, allowing it to become a stable clock reference. Excellent overviews on the operating principles of MEMS clocks and CSACs are provided by Knappe in [167], as well as in [164], and by Kitching in [168].

These CSACs have many useful characteristics which make them especially well-suited to underwater or other GPS-denied applications: they have very low power consumption[16] (on the order of 200 mW), they are small and lightweight (around 16 cm^3 and 35 g), they have excellent accuracy (typically on the order of 10^{-10} s), and, most importantly, they have excellent holdover stability with a drift that is typically less then 100 µs/day – this translates to a drift in range of less than 15 cm/day (four orders of magnitude improvement over a typical TCXO). An additional characteristic of CSACs that is especially pertinent to our application is its holdover capability after a cold start (power-on after having been unpowered for multiple days): disciplining of the clock for less than 10 minutes allows it to achieve a drift rate of less than 100 µs/day [170] (no TCXO or quartz-resonator-based clock is able to achieve anywhere near this level of cold start holdover stability). Unfortunately the Sa.45s CSAC currently suffers from a fairly high cost (in the range of $5000) due to production and quality-assurance issues, but their performance over many years of deployments in ocean applications has been well studied and documented, demonstrating that no other solution exists in this niche of low-power and high accuracy timing sources [165]. Additionally, issues of reliability and high cost are expected to be addressed as CSAC technology is incorporated into many critical timing applications, especially within the military. Between 2010 and 2014, commercial CSAC production benefited from the transition into the U.S. Army Manufacturing Technology program, which had the aim of increasing production lots to 40000 units per year

[16] As compared to traditional atomic clocks; crystal oscillators have an even lower power draw, which may be a requirement for long-duration systems that must be deployed for months or even years.

at a manufacturing cost of $200 per unit [169] [170]. For our application, issues of reliability and power consumption are acceptable, since vehicles are intended to be deployed for hours rather than for months or years at a time.

Accurate timing is critical toward enabling accurate, low-cost and low-power underwater navigation for the next generation of inexpensive and miniature AUV and ROV, as well as for the fusion of data measurements from multiple sensing streams in multi-AUV applications. The emergence of commercially available CSACs is an enabling technology that has made this possible, and their continued development ensures that positioning approaches such as piUSBL are likely to become a widespread solution for low-cost underwater navigation.

2.4 Acoustic Processing

With the piUSBL system elements defined and the hardware design choices finalized, we now walk through the prototypical software stack that converts raw digitized acoustic data into a filtered position solution. The first step in this stack is to generate acoustic range and angle measurements from the raw acoustic data – how this is performed is detailed in this section.

2.4.1 Matched Filtering for Range Estimation

Estimating the OWTT range between the beacon and receiver is essentially a signal detection problem – we wish to determine whether or not the acoustic signal broadcast by the beacon is present in the receiver measurements, and if so, its offset within the measurement [171]. The optimal linear filter to perform this detection is known as the *matched filter* – this filter maximizes the SNR for signal detection in the presence of additive white Gaussian noise. A full derivation of the matched filter demonstrating this optimality is given in Appendix A.

Generating the Range Measurement

To generate a range measurement between the AUV and the acoustic beacon, we perform matched filtering of the acoustic data captured by each element of the USBL array. The acoustic data recorded by each element $x_i[n]$ is firstly pre-whitened and magnitude-normalized using the phase transform (PHAT) [157], since all relevant information exists within the phase of these signals:

$$\hat{X}_i[\omega] = \frac{X_i[\omega]}{|X_i[\omega]|} \tag{2.11}$$

The PHAT has been empirically shown to improve robustness to noise and reverberation in real-world environments [157]. The matched filter output for each element is then:

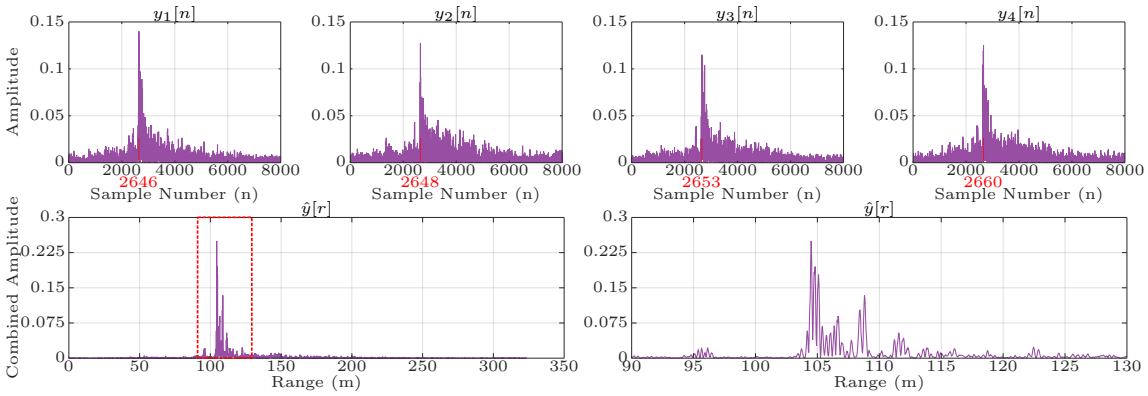

Figure 2.9: Output of the matched filter applied to experimental data – the matched filter output of the USBL array using acoustic data measured by each of the four elements is shown at the top, with their maximum sample numbers labeled in red; the matched filter output from all elements is combined to produce the range measurement shown below, with the highlighted red portion containing the maximum range shown zoomed in on the bottom right.

$$y_i[n] = \sum_{k=-\infty}^{\infty} \hat{x}_i[k]s[k-n] = \hat{x}_i[n] * s[-n] \quad \Leftrightarrow \quad Y_i[\omega] = \hat{X}_i[\omega]S^*[\omega] \tag{2.12}$$

The matched filter output for each element in the USBL array is illustrated at the top of Fig. 2.9, using 8000 samples of measured acoustic data from a deployment of the prototype AUV. Notice that the maximum (labeled in red) is consistent across elements and fairly close in time – this indicates that the broadcast signal is present in the measurements of all elements, and was 'cleanly' received by the array (since each array element received the signal at the same time). Because the USBL array is nose-mounted, certain configurations of the AUV and the beacon will cause some or all of the elements of the array to be occluded from the broadcast signal by the body of the vehicle (e.g. when the beacon is directly behind the AUV). Excluding measured acoustic data that occur during these configurations is not straightforward, but we can use the standard deviation of these arg-maxima as an initial validity check:

$$\sigma_{max\_sample} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\underset{n}{\text{argmax}}(y_i[n]) - \frac{1}{N}\sum_{i=1}^{N}\underset{n}{\text{argmax}}(y_i[n])\right)^2} \tag{2.13}$$

where $N = 4$ for the prototype system, since its USBL array has four elements. We set a threshold such that measured acoustic data is deemed valid if $\sigma_{max\_sample} < 5$, and discard the acoustic data otherwise; this threshold was set empirically from experimental testing.

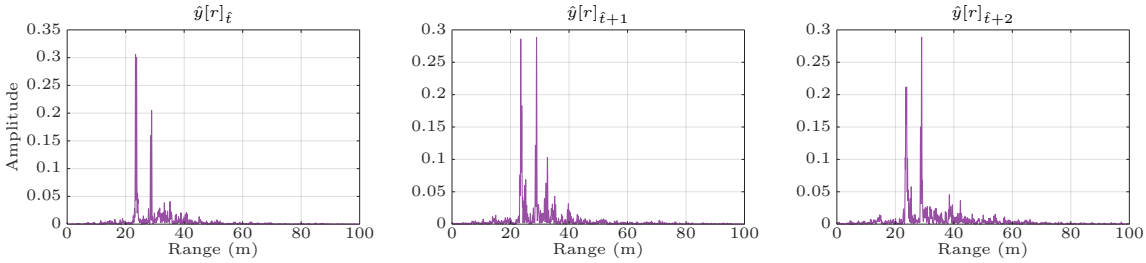

Figure 2.10: sequential range measurement distributions over 3 s at $\hat{t}$, $\hat{t}+1$ and $\hat{t}+2$ – notice that range measurements exhibit multiple modes; in this case the true ranges over time are 23.38 m, 23.50 m and 23.66 m; however, at times $\hat{t}+1$ and $\hat{t}+2$ the dominant modes occur at false ranges of 28.91 m and 28.99 m due to real-world effects such as multipath fading.

The matched filter output from all elements now need to be combined to obtain a single range measurement that balances the maxima of all the outputs. If the measured acoustic data is deemed valid, we use the empirical formula to combine the matched filter outputs:

$$\hat{y}[n] = \sum_{i=1,j=i+1}^{N} |y_i[n]|\,|y_j[n]| \qquad i \neq j \tag{2.14}$$

Finally, to obtain a 'pseudo probability distribution' for range, we normalize this combined output so that it has unit energy, and we transform this pseudo-distribution into the range domain by converting sample numbers n into ranges r:

$$\hat{y}[r] = \frac{\hat{y}[r]}{\sqrt{\sum_{\infty}^{\infty} |\hat{y}[r]|^2}} \quad \text{WHERE} \quad r = \frac{c}{F_s} \cdot n = \frac{1481\ \text{m/s}}{37500\ \text{S/s}} \cdot n \tag{2.15}$$

This results in a *range measurement distribution* that we use to estimate the distance between the AUV and the acoustic beacon. An example range distribution is shown at the bottom of Fig. 2.9, as well as in the sequential plots of Fig. 2.10.

2.4.2 Conventional Beamforming for Angle Estimation

Estimating the angle between the receiver and the beacon is essentially a direction of arrival (DOA) estimation problem – we wish to obtain an angular measurement distribution that reflects (i.e. has a maximum) in the most likely direction of the incoming acoustic signal. A number of approaches exist to perform DOA estimation, including classical beamforming methods, adaptive beamforming methods, subspace-based methods, and time difference of arrival (TDOA) methods [172] [173] [174]. Classical and adaptive beamforming approaches (i.e. Bartlett and Capon beamformers) fall under the category of nonparametric DOA methods,

in that they make no assumption about the structure of the received data – these approaches are generally highly robust, but provide low resolution in that they struggle to distinguish between multiple acoustic sources close in angular space. Subspace-based methods (such as MUSIC) fall under the category of parametric methods, in that they assume that the structure of the measurement covariance matrix can be decomposed into signal and noise covariance matrices through spectral (eigen) decomposition – separation of the noise contribution to the measurement then enables a high angular resolution output, but these methods can suffer from degraded robustness. In contrast to these approaches that operate in the frequency domain (i.e. spectral-based), TDOA methods solve a non-linear system of equations using the time-difference between receivers to estimate the direction to the source – these approaches are computationally efficient, but they require fast sampling rates to obtain high angular resolution and they are highly sensitive to timing errors.

In our application, we have the advantage of assuming that only a single source is transmitting a tracking signal within the frequency band of interest – as a consequence we do not require a high angular resolution DOA estimate, since we are not concerned with being able to distinguish between closely-spaced sources. Due to limited computational resources on-board the AUV, computational efficiency and robustness is of paramount importance, motivating the use of conventional beamforming for DOA estimation [158].

The Conventional Beamformer

A conventional beamformer (CBF) (also known as a delay-and-sum or Bartlett beamformer) is conceptually very simple – given some array of receivers that are arbitrarily arranged in some spatial geometry, and given a plane wave signal incident onto this array from some azimuth and inclination, a geometric relationship exists that maps these angles to the time at which the signal reaches each receiver. Under a far field assumption, we can assume that the acoustic wavefront that is incident onto the array is planar, which allows us to formulate the relevant spatial filtering equations that define the CBF [158].

Let us begin by defining the relevant spherical and Cartesian coordinate systems – we can convert between the two using the following equations:

$$\begin{aligned} x &= r \cdot \sin(\theta) \cdot \cos(\phi) & r &= \sqrt{x^2 + y^2 + z^2} \\ y &= r \cdot \sin(\theta) \cdot \sin(\phi) & \theta &= \arccos(\frac{z}{r}) \\ z &= r \cdot \cos(\theta) & \phi &= \arctan(\frac{y}{x}) \end{aligned} \tag{2.16}$$

where we call r the *range*, θ the *inclination* (where $\theta = 0$ points along the positive z-axis) and ϕ the *azimuth* (where $\phi = 0$ points along the positive x-axis, and positive rotation is

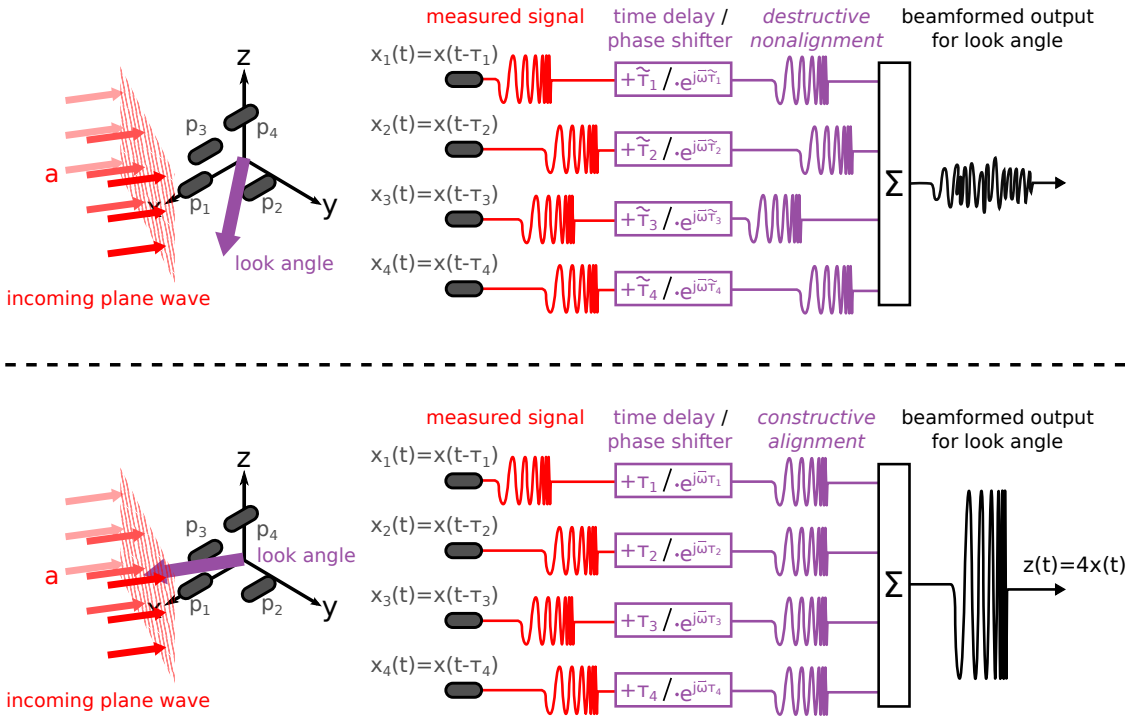

Figure 2.11: Conceptual illustration of the conventional beamformer (CBF) – the signals measured by each receiver are time-delayed or phase-shifted according to the geometry of the array, the angle of the incoming plane-wave, and the frequency of operation; *Top*: when the look-angle is not pointing in the opposite direction of the incoming plane wave, nonalignment of the filtered signals result in destructive interference and a low CBF power output. *Bottom*: when the look-angle is pointing exactly in the opposite direction of the incoming plane wave, the filtered signals are aligned, resulting in a scaled reconstruction of the incoming signal and a maximum CBF power output.

around the z-axis following the right-hand rule). Now consider an arbitrary array of *isotropic* receivers (possessing an omni-directional response) that are subject to an incoming plane wave signal, as shown in Fig. 2.11. These N receivers spatially sample the signal field at their positions $\boldsymbol{p}_i : i = 1, 2, ..., N$, yielding a collection of measurements:

$$\boldsymbol{x}(t;\boldsymbol{p}) = \begin{pmatrix} x_1(t;\boldsymbol{p_1}) \\ \vdots \\ x_N(t;\boldsymbol{p_N}) \end{pmatrix} \tag{2.17}$$

The measurement recorded by each receiver is processed using a linear time-invariant (LTI) filter with impulse response $h_i(t)$, and the outputs are summed to obtain the array output $z(t)$:

$$z(t) = \sum_{i=1}^{N} [x_i(t;\boldsymbol{p_i}) * h_i(t)] = \sum_{n=1}^{N} \int_{-\infty}^{\infty} x_i(t;\boldsymbol{p_i}) h_i(t-\tau) d\tau \tag{2.18}$$

$$= \int_{-\infty}^{\infty} \boldsymbol{h}^T(t-\tau)\boldsymbol{x}(t;\boldsymbol{p}) d\tau \tag{2.19}$$

WHERE

$$\boldsymbol{h}(\tau) = \begin{pmatrix} h_1(\tau) \\ h_2(\tau) \\ \vdots \\ h_N(\tau) \end{pmatrix} \tag{2.20}$$

Using the Fourier transform of Eq. A.7 along with the convolution–multiplication relation of Eqs. A.9 and A.10, we can write this in the frequency domain as:

$$Z(\omega) = \int_{-\infty}^{\infty} z(t) e^{-j\omega t} dt = \boldsymbol{H}^T(\omega)\boldsymbol{X}(\omega) \tag{2.21}$$

Consider again Fig. 2.11 – the plane wave is propagating in the direction $\boldsymbol{a}$ with radian frequency ω. The signal arriving at each receiver, whose positions are defined with respect to some virtual array origin, can be written in terms of the signal received by this array origin:

$$\boldsymbol{x}(t;\boldsymbol{p}) = \begin{pmatrix} x(t-\tau_1) \\ x(t-\tau_2) \\ \vdots \\ x(t-\tau_N) \end{pmatrix} \tag{2.22}$$

WHERE

$$\tau_i = \frac{\boldsymbol{a}^T \boldsymbol{p}_i}{c} \tag{2.23}$$

where c is speed of sound, and $\boldsymbol{a}$ is a unit vector in the direction of propagation:

$$\boldsymbol{a} = \begin{pmatrix} -\sin(\theta)\cdot\cos(\phi) \\ -\sin(\theta)\cdot\sin(\phi) \\ -\cos(\theta) \end{pmatrix} \tag{2.24}$$

The time difference for receiver i with respect to the array origin is thus given by:

$$\tau_i = -\frac{1}{c}[\sin(\theta)\cdot\cos(\phi)\cdot p_{x_i} + \sin(\theta)\cdot\sin(\phi)\cdot p_{y_i} + \cos(\theta)\cdot p_{z_i}] \tag{2.25}$$

$$= -\frac{1}{c}[u_x\cdot p_{x_i} + u_y\cdot p_{y_i} + u_z\cdot p_{z_i}] \tag{2.26}$$

$$= -\frac{\boldsymbol{u}^T\boldsymbol{p}_i}{c} \tag{2.27}$$

where $\boldsymbol{u} := -\boldsymbol{a}$. The i^{th} component of $\boldsymbol{x(t;p)}$ is given by $x(t-\tau_i)$, and so the i^{th} component of its Fourier transform $\boldsymbol{X}(\omega)$ can be written as:

$$X_i(\omega) = \int_{-\infty}^{\infty} x(t-\tau_i)e^{-j\omega t}dt \tag{2.28}$$

$$= \int_{-\infty}^{\infty} x(t)e^{-j\omega\tau_i}e^{-j\omega t}dt \tag{2.29}$$

$$= X(\omega)e^{-j\omega\tau_i} \tag{2.30}$$

WHERE

$$\omega\tau_i = \frac{\omega}{c}\boldsymbol{a}^T\boldsymbol{p}_i = -\frac{\omega}{c}\boldsymbol{u}^T\boldsymbol{p}_i \tag{2.31}$$

For plane waves propagating in a locally homogeneous medium, we define the wavenumber $\boldsymbol{k}$ as:

$$\boldsymbol{k} = \frac{\omega}{c}\boldsymbol{a} = \frac{2\pi}{\lambda}\boldsymbol{a} = -\frac{2\pi}{\lambda}\boldsymbol{u} \tag{2.32}$$

Using Eq. 2.23 and Eq. 2.32 we see that:

$$\omega\tau_i = \boldsymbol{k}^T\boldsymbol{p}_i \tag{2.33}$$

If we define the vector:

$$\boldsymbol{v(k)} = \begin{pmatrix} e^{-j\boldsymbol{k}^T\boldsymbol{p}_1} \\ e^{-j\boldsymbol{k}^T\boldsymbol{p}_2} \\ \vdots \\ e^{-j\boldsymbol{k}^T\boldsymbol{p}_N} \end{pmatrix} \quad \Leftrightarrow \quad \boldsymbol{v}(\omega) = \begin{pmatrix} e^{-j\omega\tau_1} \\ e^{-j\omega\tau_2} \\ \vdots \\ e^{-j\omega\tau_N} \end{pmatrix} \tag{2.34}$$

We can finally rewrite $\boldsymbol{X(\omega)}$ as:

$$\boldsymbol{X}(\omega) = X(\omega)\boldsymbol{v}(\boldsymbol{k}) = X(\omega)\boldsymbol{v}(\omega) \tag{2.35}$$

The vector $\boldsymbol{v}(\boldsymbol{k})$ or $\boldsymbol{v}(\omega)$ is referred to as the array manifold vector, and contains all the relevant characteristics of the array (i.e. the positions of the receivers, and how the array responds to a planar wave of a certain frequency incident from a given inclination and azimuth).

Compare the measurements of the array given by Eq. 2.30, $X_i(\omega) = X(\omega)e^{-j\omega\tau_i}$, collated into the *full array measurement* given by Eq. 2.35, $\boldsymbol{X}(\omega) = X(\omega)\boldsymbol{v}(\boldsymbol{k})$, to the LTI filter output applied to the full array measurement that is given by Eq. 2.21, $Z(\omega) = \boldsymbol{H}^T(\omega)\boldsymbol{X}(\omega)$. From these equations, it is readily apparent that the natural response of the array due to its geometry is a collection of LTI filters with transfer functions $H_i(\omega) = e^{-j\omega\tau_i}$. Thus, the array geometry results in shifting in time of the signal received by element i by τ_i, a value corresponding to the time difference in signal reception between element i and the array origin.

If we align the signals from all elements by their respective delays τ_i and then sum and normalize them, we get the illustration shown at the bottom of Fig. 2.11 (shown without the normalization); this acts as a collection of LTI filters that add τ_i delays to the signals recorded by the elements, essentially 'undoing' the time differences that each receiver experiences due to the incident angle of the incoming planar waveform. After normalization, the output $z(t)$ is equivalent to the original transmitted waveform. To achieve this, it is obvious that the LTI filter for each element must have impulse response:

$$h_i(\tau) = \frac{1}{N}\delta(\tau + \tau_i) \tag{2.36}$$

And as a result, the sum of the filter outputs is a reconstruction of the incident planar waveform:

$$z(t) = x(t) \tag{2.37}$$

This collection of LTI filters (which we call a *spatial filter*) is referred to as the *conventional beamformer (CBF)* or as the delay-and-sum beamformer. We can rewrite the spatial filter in compact form as:

$$\boldsymbol{H}^T(\omega) = \frac{1}{N}\boldsymbol{v}^H(\boldsymbol{k}) \tag{2.38}$$

The Hermitian operator converts $-j$ to j, and $\boldsymbol{v(k)}$ is defined in terms of a single *look-angle* (θ, ϕ) and the frequency or wavelength λ of the incident signal (as well as the array geometry). As a final note, an arbitrary spatial filter applied to an array can therefore be written in terms of the desired filter and the natural response of the array due to its geometry:

$$\mathcal{Y}(\omega; \boldsymbol{k}) := \boldsymbol{H}^T(\omega)\boldsymbol{v}(\boldsymbol{k}) \tag{2.39}$$

$\mathcal{Y}(\omega; \boldsymbol{k})$ can be interpreted as the array transfer function, and allows us to observe the output response of an array by abstracting away its geometrical properties.

The CBF is defined by Eq. 2.38 with computations typically done in the frequency domain for reasons of efficiency. For a continuous wave narrowband (single frequency) signal, the time delays experienced by array elements correspond to phase shifts in the Fourier domain:

$$x(t - \tau_i) \leftrightarrow e^{-j\omega\tau_i} X(\omega) \tag{2.40}$$

Therefore, in the Fourier domain, conventional beamforming is simply performed by applying opposing phase shifts rather than time delays, using Eq. 2.38 as illustrated in Fig. 2.11. Finally, for practical purposes, for a digitized, discrete wideband signal that contains multiple frequencies (as is the case for our LFM signal), a single time delay corresponds to increasing phase shifts with frequency, resulting in a vector of frequencies $\boldsymbol{\omega}$:

$$x[n - \tau_i] \leftrightarrow e^{-j\boldsymbol{\omega}\tau_i} \boldsymbol{X}[\boldsymbol{\omega}] \tag{2.41}$$

In this case, the array manifold vector actually becomes a matrix, and we rewrite Eq. 2.38 of the CBF as:

$$\boldsymbol{H}^T[\boldsymbol{\omega}] = \frac{1}{N}\boldsymbol{V}^H[\boldsymbol{\omega}] \tag{2.42}$$

where $\boldsymbol{V}(\omega)$ contains the M frequency columns from an M–point DFT:

$$\boldsymbol{V}[\boldsymbol{\omega}] = \begin{pmatrix} e^{-j\boldsymbol{\omega}\tau_1} \\ e^{-j\boldsymbol{\omega}\tau_2} \\ \vdots \\ e^{-j\boldsymbol{\omega}\tau_N} \end{pmatrix} = \begin{bmatrix} e^{-j\omega_1\tau_1} & e^{-j\omega_2\tau_1} & \dots & e^{-j\omega_M\tau_1} \\ e^{-j\omega_1\tau_2} & e^{-j\omega_2\tau_2} & \dots & e^{-j\omega_M\tau_2} \\ \vdots & & \ddots & \vdots \\ e^{-j\omega_1\tau_N} & e^{-j\omega_2\tau_N} & \dots & e^{-j\omega_M\tau_N} \end{bmatrix} \tag{2.43}$$

Generating the Angle Measurement

To generate a robust angle measurement between the AUV and the acoustic beacon, we first pre-filter the acoustic data captured by each element of the array using the matched filter detailed in section 2.4.1; the allows us to avoid computationally intensive beamforming if the measurement is not valid, and it also improves SNR which in turn improves CBF accuracy – since both the matched filter and CBF are LTI filters, this sequential structure is sound. However, in order to exploit our knowledge of the structure of the broadcast signal, we use the chirp Z-transform (CZT) [159] instead of the discrete Fourier transform (DFT) to transform the signal into the frequency domain for beamforming. Using the CZT essentially allows us to dictate the frequency interval of interest while ignoring irrelevant frequencies; since our broadcast signal has energy in the range of 16–18 kHz, the CZT allows us to efficiently extract these frequency components with high resolution, improving processing speed and accuracy of beamforming.

Note that Eq. 2.42 for beamforming is only for a single look-angle defined by the azimuth ϕ and inclination θ. However, since we do not know the angle of the incoming acoustic wave, we must perform an exhaustive search over a set of candidate look-angles in order to find the likeliest direction of the incoming acoustic signal and thus the likeliest direction to the beacon. First, we must rewrite the output of the beamformer for a given look-angle (Eq. 2.21) as an element-wise multiplication and sum, since the CBF spatial filter $\boldsymbol{H}(\omega)$ and the CZT of the array measurement $\boldsymbol{X}(\omega)$ are both matrices rather than vectors:

$$\boldsymbol{Z}[\boldsymbol{\omega};\theta,\phi] = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{H}_i[\boldsymbol{\omega};\theta,\phi] \odot \boldsymbol{X}_i[\boldsymbol{\omega}] \tag{2.44}$$

WHERE

$$\boldsymbol{H}_i[\boldsymbol{\omega};\theta,\phi] = \boldsymbol{V}_i^*[\boldsymbol{\omega}] = e^{j\boldsymbol{\omega}\tau_i} \tag{2.45}$$

$$= \begin{bmatrix} e^{j\omega_1\tau_i} & e^{j\omega_2\tau_i} & \dots & e^{j\omega_M\tau_i} \end{bmatrix} \tag{2.46}$$

The output power of the beamformer averaged over all M frequency components is then:

$$|\hat{Z}[\theta,\phi]|^2 = \frac{1}{M}\sum_{i=1}^{M} |\boldsymbol{Z}[\omega_i;\theta,\phi]|^2 \tag{2.47}$$

Now given a set of candidate look-angles to perform a search over, we can calculate the beamformed output power for each look-angle. The maximum over these set of candidates provides the maximum likelihood estimate (MLE) for the angle between the array and the beacon, in the reference frame of the array:

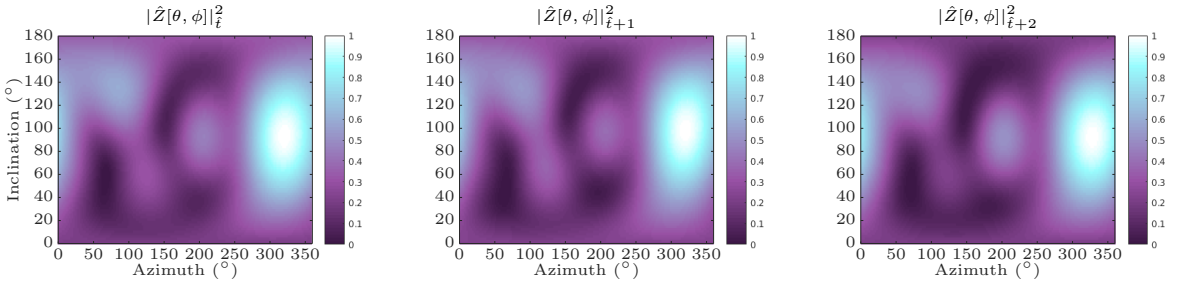

Figure 2.12: sequential angle measurement distributions over 3 s at $\hat{t}$, $\hat{t}+1$ and $\hat{t}+2$ – notice that as with range measurements, angle measurements are multimodal as a result of beamforming; in this case beamforming is performed over a regular grid of 180 azimuths (2° resolution) and 90 inclinations (2° resolution); in this sequence, the maximum of each distribution indicates that the MLE angle to the beacon (in the reference frame of the array) is at approximately $\theta = 94°$ and $\phi = 321°$.

$$(\theta^{MLE}, \phi^{MLE}) = \underset{\theta,\phi}{\text{argmax}} |\hat{Z}[\theta,\phi]|^2 \tag{2.48}$$

To obtain a pseudo-distribution for angle, we simply calculate the beamformed output power for a regular grid of azimuth and inclination look-angles using Eq. 2.47. This results in a 'heatmap' that we call our *angle measurement distribution*, which we use to estimate the angle between the AUV and the acoustic beacon. Example angle distributions are illustrated in the sequential plots of Fig. 2.12 – note that because we are not interested in absolute power levels, for simplicity we unit-normalize $|\hat{Z}[\theta,\phi]|^2$. The angle distributions of Fig. 2.12 were generated using in-water acoustic data from the piUSBL system on the prototype SandShark AUV – in this case the geometry of the 3.85 cm nose-mounted array is given by:

$$\boldsymbol{P_{prototype}} = \begin{bmatrix} \boldsymbol{p_1^T} \\ \boldsymbol{p_2^T} \\ \boldsymbol{p_3^T} \\ \boldsymbol{p_4^T} \end{bmatrix} = \begin{bmatrix} 0.03143 & 0 & 0 \\ 0 & 0 & 0.02223 \\ 0 & -0.01925 & -0.01111 \\ 0 & 0.01925 & -0.01111 \end{bmatrix} \tag{2.49}$$

The process of matched filtering outlined in section 2.4.1, and of beamforming outlined in section 2.4.2, provide us with range and angle measurement distributions and encompass the acoustic processing step of our prototypical system. Since the acoustic beacon and the receiver array broadcasts and records at a rate of 1 Hz, these measurements can be used to generate an instantaneous estimate of the relative position between the beacon and the AUV once every second. The acoustic processing performed by the prototypical system is summarized in algorithm 1.

Algorithm 1 Acoustic processing: matched filtering followed by conventional beamforming

procedure GENERATE_RANGE_MF($c, f_s, s, x_i \ : \ i = 1, ..., N$)
▷ Generates the range measurement using matched filtering
▷ Inputs: c: speed-of-sound, f_s: sampling frequency, s: broadcast template, x_i: measured signals
▷ Outputs: $\hat{y}$: range measurement distribution, y_i: matched filter output for each signal, σ_{max_sample}: standard deviation for validity check
    $S = DFT(s)$ ▷ Frequency domain template
    $X_i = DFT(x_i)$ ▷ Frequency domain measurement on element i
    $\hat{X}_i = \frac{X_i}{|X_i|}$ ▷ PHAT transform, Eq. 2.11
    $Y_i = \hat{X}S^*$ ▷ Matched filter convolution, Eq. 2.12
    $y_i = IDFT(Y_i)$ ▷ Output of matched filter for element i
    $\hat{y} = \sum_{i,j=1}^{N} |y_i||y_j| \ : \ i \neq j$ ▷ Combined matched filter outputs, Eq. 2.14
    $\hat{y} = \frac{\hat{y}}{\sqrt{\sum |\hat{y}|^2}}$ ▷ Range measurement, Eq. 2.15
    $\sigma_{max\_sample} = STD\_DEV(y_i \ : \ i = 1, ..., N)$ ▷ Validity check, Eq. 2.13
    return $\hat{y}, y_i, \sigma_{max\_sample}$
end procedure

procedure GENERATE_ANGLE_CBF($\boldsymbol{\phi}, \boldsymbol{\theta}, f_{lower}, f_{upper}, f_s, \boldsymbol{H}, y_i \ : \ i = 1, ..., N$)
▷ Generates the angle measurement using the Conventional Beamformer
▷ Inputs: $\boldsymbol{\phi}$: azimuth of look-angles, $\boldsymbol{\theta}$: inclination of look-angles, f_{lower}: lower frequency cutoff, f_{upper}: upper frequency cutoff, f_s: sampling frequency, $\boldsymbol{H}$: CBF spatial filter, y_i: matched filter outputs
▷ Outputs: $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|$: angle measurement distribution 'heatmap'
    $Y_i = CZT(y_i, f_{lower}, f_{upper}, f_s, M)$ ▷ Chirp Z-Transform of matched filter signals with M frequencies in desired frequency range
    $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}| = zeros(len(\boldsymbol{\theta}), len(\boldsymbol{\phi}))$ ▷ Storage for angle measurement
    for θ_k *in* $\boldsymbol{\theta}$ **do** ▷ Loop through all look-angles and calculate CBF power output
        for ϕ_l *in* $\boldsymbol{\phi}$ **do**
            $Z = \frac{1}{N} \sum_i \boldsymbol{H}[k, l, :, i] \odot Y_i$ ▷ Applying phase shifts for this look-angle, Eq. 2.44
            $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|[k, l] = \frac{1}{M} \sum |Z|^2$ ▷ Sum over all frequency components, Eq. 2.47
        end for
    end for
    return $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|$
end procedure

procedure PROCESS_ACOUSTICS($\boldsymbol{P}, N_{azim}, N_{incl}, f_{lower}, f_{upper}, c, f_s, s, x_i \ : \ i = 1, ..., N$)
▷ Continuously processes acoustic data measured by the array
▷ Inputs: $\boldsymbol{P}$: element positions, N_{azim}: no. of azimuths, N_{incl}: no. of inclinations, f_{lower}: lower frequency cutoff, f_{upper}: upper frequency cutoff, c: speed-of-sound, f_s: sampling frequency, s: broadcast template, x_i: measured signals
▷ Calculates: ***range***: range measurement distribution, ***angle***: angle measurement distribution
    $\boldsymbol{\phi} = 0 : \frac{2\pi}{N_{azim}} : 2\pi - \frac{2\pi}{N_{azim}}$ ▷ Equally-spaced azimuths to beamform

$\boldsymbol{\theta} = 0 : \frac{\pi}{N_{incls}} : \pi - \frac{\pi}{N_{incl}}$ ▷ Equally-spaced inclinations to beamform
$\boldsymbol{f} = f_{lower} : f_{upper}$ ▷ M equally spaced frequencies (e.g. 16–18 kHz) for CZT
▷ Loop through all azimuth/inclination combinations and pre-compute and store phase shifts for angle measurement distribution 'heatmap'
$\boldsymbol{H} = zeros(N_{incl}, N_{azim}, M, N)$ ▷ CBF spatial filter: phase-shift storage
for θ_k *in* $\boldsymbol{\theta}$ **do**
  for ϕ_l *in* $\boldsymbol{\phi}$ **do**
    $$\boldsymbol{a} = \begin{pmatrix} -\sin(\theta_k) \cdot \cos(\phi_l) \\ -\sin(\theta_k) \cdot \sin(\phi_l) \\ -\cos(\theta_k) \end{pmatrix}$$ ▷ Eq. 2.24
    $\tau = \frac{\boldsymbol{P} \cdot \boldsymbol{a}}{c}$ ▷ Time delays for this look-angle, Eq. 2.27
    $\boldsymbol{H}[k, l, :, :] = e^{-2j\pi\tau \boldsymbol{f}}$ ▷ Phase shifts for this look-angle, Eqs. 2.42 & 2.43
  end for
end for
loop
  $\hat{y}, y_i, \sigma_{max\_sample} = Generate\_Range\_MF(c, f_s, s, x_i \ : \ i = 1, ..., N)$
  if $\sigma_{max\_sample} < 5$ **then** ▷ Validity check
    $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}| = Generate\_Angle\_CBF(\boldsymbol{\phi}, \boldsymbol{\theta}, f_{lower}, f_{upper}, f_s, \boldsymbol{H}, y_i \ i = 1, ..., N)$
    $\boldsymbol{range} = \hat{y}$ ▷ Valid range measurement distribution
    $\boldsymbol{angle} = |\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|$ ▷ Valid angle measurement distribution
  else
    $\boldsymbol{range} = None$
    $\boldsymbol{angle} = None$
  end if
end loop
end procedure

2.5 Bayesian Filtering

In the previous section we described how range and angle measurement distributions are generated from the raw acoustic data captured by the piUSBL array. From these distributions we may be tempted to simply extract their MLE values and project them via triangulation to instantaneously estimate the relative position of the beacon in the reference frame of the array. However, this approach discards the majority of the information present within our measurements, which can lead to drastically incorrect estimates of vehicle position. Consider the sequence of range measurements in Fig. 2.10 – selecting the MLE from these measurements would lead us to estimate the vehicle's range at a false maxima of around 29 m rather than the true maxima at 23 m. Instead, since we know that the position of the acoustic beacon and the position of the AUV varies smoothly with time, we fuse our range and angle measurements with vehicle odometry and inertial measurements in order to generate a *temporally filtered*, continuous estimate of the position of the beacon relative to the vehicle. This fusion is performed using *Bayesian filtering* [175].

2.5.1 Bayesian Filtering in General

State Representation

In general, the state of a vehicle operating in 3D space at some time t can be represented by a vector of random variables containing its position and attitude in a global frame of reference:

$$\boldsymbol{x}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \gamma(t) \\ \beta(t) \\ \alpha(t) \end{bmatrix} \quad \text{WHERE} \quad \begin{bmatrix} x(t) & : & \text{global x–position} \\ y(t) & : & \text{global y–position} \\ z(t) & : & \text{global z–position} \\ \gamma(t) & : & \text{roll, rotation around body–fixed x–axis} \\ \beta(t) & : & \text{pitch, rotation around body–fixed y–axis} \\ \alpha(t) & : & \text{yaw, rotation around body–fixed z–axis} \end{bmatrix} \tag{2.50}$$

This is known as the state vector, where attitude is given in Euler angles.

Motion Model and State Prediction

The vehicle's state vector evolves smoothly over time based on control inputs $\boldsymbol{u}(t)$. Given a sufficiently accurate motion model of the vehicle, this enables us to predict the future state of the vehicle given the current state and the control inputs. Practically speaking, this motion model can be expressed in a discrete time space with a discretized time-step Δt using an arbitrarily complex function of current state and control input:

$$\boldsymbol{x}(t + \Delta t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t + \Delta t)) \tag{2.51}$$

In general, this function is nonlinear and can be made as complex as desired to capture the evolution of vehicle state over time – it can incorporate factors such as the physical properties of the vehicle (size, shape, mass, inertia), its interaction with the environment (lift, drag, thrust), as well as environmental forces acting upon it (currents, buoyancy). As expected, deriving a complex equation of motion that accounts for a large number of parameters is exceedingly difficult, but can provide better predictive power [176], [177], [178], [74] [179] [180]. Regardless of its complexity, no model can perfectly capture the dynamics of the vehicle operating in a real-world environment, and so an additional noise term is added to account for these uncertainties:

$$\boldsymbol{x}(t + \Delta t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t + \Delta t), \boldsymbol{\sigma}_{\boldsymbol{x}}(t)) \tag{2.52}$$

This equation of motion can instead be formulated in the language of probabilities. If we consider the state $\boldsymbol{x}(t)$ and the control inputs $\boldsymbol{u}(t)$ to be modeled as random variables, we

can rewrite the motion model in a *probabilistic representation* as the *state transition probability* [175]:

$$p(\boldsymbol{x}_{t+\Delta t}|\boldsymbol{x}_t, \boldsymbol{u}_{t+\Delta t}) \tag{2.53}$$

Simply put, the distribution of the predicted state $\boldsymbol{x}_{t+\Delta t}$ is dependent on the distributions of the current state $\boldsymbol{x}_t$ and the control input $\boldsymbol{u}_{t+\Delta t}$ – it is stochastically generated from these distributions, and any state is conditionally independent of all other states if given the previous state. In effect, the evolution of vehicle state is modeled using a Markovian process as a hidden Markov model (HMM), where the next state at any time $t + \Delta t$ is determined by the current state at t and the control $u_{t+\Delta t}$. Sensors on board the vehicle provide information that enable it to measure its state – equivalent to the observation in the HMM. We describe this part of the model in the next section.

Measurement Model and State Update

If the vehicle had no way of observing its own state, the HMM would be a Markov chain in which the uncertainty in the state would grow unbounded over time according to the motion model and its associated uncertainties. Thus, the vehicle will typically be equipped with various proprioceptive and exteroceptive sensors to measure its state and the environment around it. The measurements provided by these sensors will vary depending on the state of the vehicle (i.e. its position in the environment and its attitude), as well as the state of the environment or map [175]:

$$\boldsymbol{z}(t) = g(\boldsymbol{x}(t), \boldsymbol{m}(t)) \tag{2.54}$$

where $\boldsymbol{z}(t)$ is the collection of measurements from all sensors and $\boldsymbol{m}(t)$ is the map[17]. As with the case with the motion model, uncertainties in these measurements in the real-world are modeled using an additional noise term:

$$\boldsymbol{z}(t) = g(\boldsymbol{x}(t), \boldsymbol{m}(t), \boldsymbol{\sigma}_{\boldsymbol{z}}(t)) \tag{2.55}$$

As previously mentioned, these measurements can be modeled as the observations in a HMM – we can reformulate this measurement model in probabilistic terms as the *measurement probability* [175]:

[17] Note that the term *map* is used in the general sense, and encompasses all possible measurements that depend on environmental factors. For example, our acoustic measurements of range and angle to the beacon, as well as the state of the beacon, can be included in this term.

$$p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{m}_t) \tag{2.56}$$

where the measurement distribution is dependent on the distributions of the current state $\boldsymbol{x}_t$ and the map $\boldsymbol{m}_t$, and is conditionally independent of all other measurements and states given the current state.

The General Bayes Filter

The Bayes filter is a principled framework for fusing predictions generated by the motion model with the set of sensor measurements (or observations). The filter continuously tracks the new state $\boldsymbol{x}_t$ over time using only the posterior distribution of the previous state $\boldsymbol{x}_{t-\Delta t}$, the control input $\boldsymbol{u}_t$ and the observations $\boldsymbol{z}_t$; this framework ensures that the state maintains consistency with both the motion model and the measurements as it evolves:

$$p(\boldsymbol{x}_{t-\Delta t}|\boldsymbol{z}_{t-\Delta t}) \overset{\boldsymbol{u}_t, \boldsymbol{z}_t}{\Longrightarrow} p(\boldsymbol{x}_t|\boldsymbol{z}_t) \tag{2.57}$$

It does this using a recursive process involving a *predict step* and an *update step*, making use of the Bayes rule:

$$p(A|B, C) = \frac{p(B|A, C)p(A|C)}{p(B|C)} \tag{2.58}$$

The Bayes filter is summarized in algorithm 2 below [175].

Algorithm 2 The general Bayes filter

1: **procedure** BAYES_FILTER($p(\boldsymbol{x}_0), \boldsymbol{u}_t, \boldsymbol{z}_t$)
   ▷ Continuously estimates the current state using the motion model and measurements
   ▷ Inputs: $p(\boldsymbol{x}_0)$: initial state distribution, $\boldsymbol{u}_t$: control inputs, $\boldsymbol{z}_t$: measurements
   ▷ Calculates: $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$: current state distribution
2: **loop**
3: $p(\boldsymbol{x}_t|\boldsymbol{z}_{t-\Delta t}) = \int p(\boldsymbol{x}_t|\boldsymbol{u}_t\boldsymbol{x}_{t-\Delta t})p(\boldsymbol{x}_{t-\Delta t}|\boldsymbol{z}_{t-\Delta t})d\boldsymbol{x}_{t-\Delta t}$ ▷ Predict step using $\boldsymbol{u}_t$
4: $p(\boldsymbol{x}_t|\boldsymbol{z}_t) = \eta \cdot p(\boldsymbol{z}_t|\boldsymbol{x}_t, \boldsymbol{z}_{t-\Delta t})p(\boldsymbol{x}_t|\boldsymbol{z}_{t-\Delta t})$ ▷ Update step using $\boldsymbol{z}_t$
5: **end loop**
6: **end procedure**

The predict step in the Bayes filter uses the control input $\boldsymbol{u}_t$ in order to transition the state distribution from the previous time step $p(\boldsymbol{x}_{t-\Delta t}|\boldsymbol{z}_{t-\Delta t})$ to the new state distribution. The distribution over the new state $p(\boldsymbol{x}_t|\boldsymbol{z}_{t-\Delta t})$ is calculated by integrating (summing) over the

product of two distributions: the prior distribution of the state and the probability that the control $\boldsymbol{u}_t$ applied to the motion model causes a transition from $\boldsymbol{x}_{t-\Delta t}$ to $\boldsymbol{x}_t$.

The update step in the Bayes filter incorporates sensor measurements of the vehicle $\boldsymbol{z}_t$ to update the distribution over the current state. The distribution over the current state after the predict step, $p(\boldsymbol{x}_t|\boldsymbol{z}_{t-\Delta t})$, is multiplied by the probability that the sensor measurements are observed by this current state distribution. Notice that it makes use of the Bayes rule of Eq. 2.58; however, the denominator in Eq. 2.58 is not explicitly calculated – instead, we ensure that $p(\boldsymbol{x}_t|\boldsymbol{z}_t)$ is a probability distribution by normalizing its integral to 1 using the normalization factor η.

The Bayes filter requires an initial state distribution $p(\boldsymbol{x}_0)$ from which the state evolves under a HMM assumption using these predict and update steps. However, this general form of the Bayes filter is computationally intractable, since it requires an integration and product over arbitrary probability distributions. In general, it can only be performed for very simple state estimation problems where the state space is finitely discretized, so that the integral becomes a sum, and the product can be performed as an element-wise multiplication. Otherwise, assumptions can be made about the form of the state and measurement distributions – for example, by assuming that they are unimodal Gaussians, closed-form integrals and products become available, resulting in the well-known Kalman filter and its variants.

2.5.2 Bayesian Filtering for piUSBL

In our specific application of piUSBL positioning, we would like to fuse our acoustic range and angle measurement distributions with a simple motion model of the AUV, in order to generate a state estimate that is consistent with how the vehicle can move, as well as with the instantaneous position implied by the acoustic measurements from the piUSBL array.

State Representation and Motion Model

Our state representation follows a rather unorthodox approach, which requires us to go through the various frames of reference used by our system. In our case we wish to track the position of the beacon relative to the AUV, so we formulate the state vector that is partially tracked by our Bayesian filter as:

$$\boldsymbol{x}(t) = \begin{bmatrix} x^{vcf}_{beacon}(t) \\ y^{vcf}_{beacon}(t) \\ x^{llf}_{beacon} \\ y^{llf}_{beacon} \\ z^{llf}_{beacon} \\ x^{llf}_{auv}(t) \\ y^{llf}_{auv}(t) \\ z^{llf}_{auv}(t) \\ \gamma_{auv}(t) \\ \beta_{auv}(t) \\ \alpha_{auv}(t) \\ v_{auv}(t) \end{bmatrix} \text{ WHERE } \begin{bmatrix} x^{vcf}_{beacon}(t) & : & \text{x–position of beacon relative to the AUV} \\ y^{vcf}_{beacon}(t) & : & \text{y–position of beacon relative to the AUV} \\ x^{llf}_{beacon} & : & \text{x–position of beacon in the local-level frame} \\ y^{llf}_{beacon} & : & \text{y–position of beacon in the local-level frame} \\ z^{llf}_{beacon} & : & \text{z–position of beacon in the local-level frame} \\ x^{llf}_{auv}(t) & : & \text{x–position of AUV in the local-level frame} \\ y^{llf}_{auv}(t) & : & \text{y–position of AUV in the local-level frame} \\ z^{llf}_{auv}(t) & : & \text{z–position of AUV in the local-level frame} \\ \gamma_{auv}(t) & : & \text{roll of AUV, rot}^{\text{n}} \text{ around body–fixed x–axis} \\ \beta_{auv}(t) & : & \text{pitch of AUV, rot}^{\text{n}} \text{ around body–fixed y–axis} \\ \alpha_{auv}(t) & : & \text{yaw of AUV, rot}^{\text{n}} \text{ around body–fixed z–axis} \\ v_{auv}(t) & : & \text{speed-over-ground of AUV in forward direction} \end{bmatrix} \tag{2.59}$$

The first two elements of the state vector are tracked using our Bayesian filter, and are dependent on other elements of the state vector, the motion model, and our acoustic measurements. Other elements of the state vector are either constants provided to the vehicle before deployment (e.g. the position of the beacon in the local-level frame (LLF)), or time-varying values supplied to the vehicle by proprioceptive sensors (e.g. vehicle attitude, depth and speed-over-ground), or are calculated from other elements of the state vector (e.g. AUV position in the LLF).

The various reference frames used by our prototypical piUSBL system are illustrated in Fig. 2.13. There are three major frames of reference: the local-level frame (LLF), which is locally tangential to the Earth's surface, and whose axes are ordered according East-North-Up (ENU) convention[18]; the ENU vehicle-carried frame (VCF) whose axes also follow the ENU convention, but whose origin is at the center of gravity of the vehicle – as such, these axes always aligned with the axes of the LLF, but it is 'carried' around by the vehicle; and the body-fixed frame (BFF) which is rigidly 'fixed' to the AUV body so that it rotates with it, and whose origin is the vehicle's center of gravity. The CBF operates in the BFF, and so for visual purposes we have placed these axes in Fig. 2.13 at the nose of the vehicle where the USBL array is located; however, for the sake of simplicity, in our work we ignore any lever-arm effects between the vehicle's center of gravity and the position of the array, and we assume that the origins of the BFF and the VCF are collocated. In the BFF of our system, positive yaw is around the z–axis following the right-hand rule and causes the AUV to turn left; positive pitch is around the y–axis following the left-hand rule and causes the AUV to

[18] In the East-North-Up (ENU) convention, the x–axis points East, the y–axis points North, and the z–axis points up.

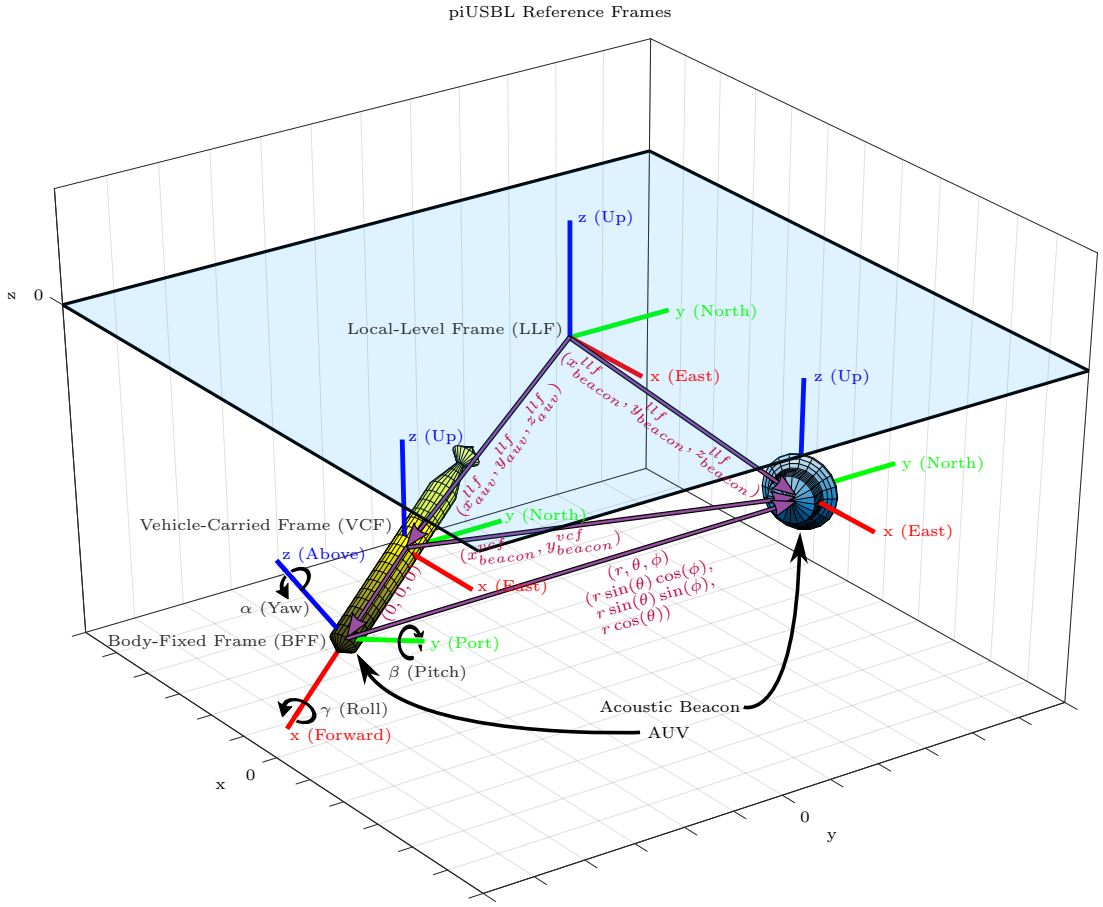

Figure 2.13: Illustration of the reference frames used by the prototypical piUSBL system – there are three main reference frames: the local-level frame (LLF), a frame of reference that is locally tangential to the Earth's surface, and whose axes follow the East-North-Up (ENU) convention; the ENU vehicle-carried frame (VCF) whose axes are aligned with the LLF, but whose origin is 'carried' by the vehicle at its center of gravity; and the body-fixed frame (BFF) which is rigidly attached to the AUV so that it rotates with it, and is also centered at its center of gravity. Note that for clarity, we have visually separated the origins of the VCF and the BFF, but in our system we assume co-locality for simplicity. Note also that BFF rotations are non-standard: roll γ and yaw α follow the right-hand rule, but pitch β follows the left-hand rule.

pitch up; and finally, positive roll is around the x–axis following the right-hand rule and causes the AUV to roll right. When the x, y, and z axes of the BFF are parallel with the x, y, and z axes of the VCF or the LLF, then pitch, roll and yaw of the vehicle are all equal to zero.

Let us now work through each of the elements of the state vector besides x^{vcf}_{beacon} and y^{vcf}_{beacon} which we wish to track via our filter. The next three elements, x^{llf}_{beacon}, y^{llf}_{beacon}, and z^{llf}_{beacon}, are set arbitrarily based on the beacon's depth in the water and the origin of the LLF – for convenience, we set $x^{llf}_{beacon} = 0$, $y^{llf}_{beacon} = 0$ and $z^{llf}_{beacon} = -1$ (the beacon is always deployed to a depth of 1 m). The next two elements, x^{llf}_{auv} and y^{llf}_{auv} can then be calculated using x^{vcf}_{beacon}, y^{vcf}_{beacon}, x^{llf}_{beacon}, and y^{llf}_{beacon}. The AUVs depth sensor directly provides an estimate of z^{llf}_{auv} via a depth value z^{llf}_{depth}. The next three elements which pertain to AUV attitude are directly provided to the vehicle via its IMU and magnetometer, giving values for roll γ, pitch β and yaw α that have been pre-filtered by the vehicle's AHRS. Finally, speed-over-ground v_{auv} is directly provided using the AUVs propeller RPM via the transformation in Eq. 2.1.

Rather than having a motion model for the AUV, our prototypical piUSBL system uses a very simple forward velocity motion model in order to perform state prediction to update the first two elements of the state vector, which pertain to the relative position of the beacon:

$$x^{vcf}_{beacon}(t + \Delta t) = x^{vcf}_{beacon}(t) - v_{auv}(t) \cdot \Delta t \cdot \cos(\alpha_{auv}(t)) \tag{2.60}$$

$$y^{vcf}_{beacon}(t + \Delta t) = y^{vcf}_{beacon}(t) - v_{auv}(t) \cdot \Delta t \cdot \sin(\alpha_{auv}(t)) \tag{2.61}$$

This motion model is used within the prediction step of a Bayesian filtering approach known as the *particle filter*, whose configuration for our specific application is described next.

The Particle Filter

Bayesian filtering can be performed in a computationally tractable way using a wide variety of methods, each of which have their own strengths and weaknesses. In general, these methods fall under two families: parametric and non-parametric approaches [175]. Parametric approaches are dominated by the Kalman filter and its variants, including the extended Kalman filter and the unscented Kalman filter. The principle assumption that these variants make is that the state and measurement distributions can be accurately represented using a unimodal Gaussian, and under this assumption, they derive closed-form equations for the state prediction and state update steps of Bayesian filtering. As such, they are extremely computationally efficient, but lack the ability to accurately model multimodal measurements and states. The Gaussian sum filter is a parametric approach that attempts to overcome this assumption by representing multimodal distributions using a weighted sum of Gaussians [181] [182]. Non parametric approaches include the histogram filter and the discrete Bayes filter, where the state space is discretized into finite elements (e.g. a static or dynamic grid), and

each element stores the cumulative posterior as a single probability value; these discretized filters are able to model multimodal distributions to an arbitrary precision (based on the resolution of the discretization), but are computationally expensive with computational cost increasing exponentially with the dimension of the state space. The family of non-parametric filters also includes the particle filter[19] and its many variants – the key idea with these approaches is to represent the posterior distribution using a set of randomly sampled values drawn from the state distribution. These sampled values are referred to as 'particles', and represent possible instances of the true state according to its distribution – as a result, the more particles that are grouped within a region of the state space, the more likely that the true state falls within this region. To accurately represent the underlying distribution, often a large number of particles must be used – this represents a trade-off between accuracy of the representation and computational cost [175]. Since the literature surrounding Bayesian filtering is vast and spans many fields, we will not go into further detail about specific filters, and defer the interested reader to the good overviews provided by [175] [182] and [181].

Our piUSBL system has a number of practical considerations that limit our choice of which Bayesian filtering approach is most appropriate. First, note that our acoustic range and angle measurement distributions are clearly multimodal, as seen in Figs. 2.10 and 2.12; this multimodality is due to the undesirable effects of underwater acoustic propagation such as multipath, reverberation, reflections and interference. Second, our motion model is non-linear and whichever representation our filter uses for the underlying distribution must be easily transformed between our various measurement spaces. Thirdly, the filter we use must be fast enough to run on a Raspberry Pi 3, and would ideally be straightforward to implement. These considerations led us to select the particle filter.

Particle Filter Initialization

The prototypical piUSBL system maintains a set of R particles, each of which contains a realization of the state vector (Eq. 2.59) and an associated *particle weight* w_i:

$$\boldsymbol{S}(t) = \{\boldsymbol{s}_1(t), \boldsymbol{s}_2(t), ..., \boldsymbol{s}_i(t), ..., \boldsymbol{s}_R(t)\} \tag{2.62}$$

WHERE

$$\boldsymbol{s}_i(t) = \begin{bmatrix} \boldsymbol{x}_i(t) \\ w_i \end{bmatrix} \quad i = 1, ..., R \tag{2.63}$$

The particle filter is initialized by initializing this set of particles using some initial state distribution. In our case, before vehicle deployment we know the position of the AUV in the

[19] The particle filter is also known as sequential Monte-Carlo filtering.

LLF from GPS, and we also know the position of the beacon in the LLF (since we arbitrarily set it), which remains static over the duration of the mission. Using this information we can initialize the state vector of each particle according to:

$$\boldsymbol{x}_i(0) = \begin{bmatrix} x^{vcf}_{i,beacon}(0) \\ y^{vcf}_{i,beacon}(0) \\ x^{llf}_{i,beacon} \\ y^{llf}_{i,beacon} \\ z^{llf}_{i,beacon} \\ x^{llf}_{i,auv}(0) \\ y^{llf}_{i,auv}(0) \\ z^{llf}_{i,auv}(0) \\ \vdots \end{bmatrix} = \begin{bmatrix} x^{llf}_{i,beacon} - x^{llf}_{i,auv}(0) = -x^{llf}_{gps}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ y^{llf}_{i,beacon} - y^{llf}_{i,auv}(0) = -y^{llf}_{gps}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ 0 \\ 0 \\ -1 \\ x^{llf}_{gps}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ y^{llf}_{gps}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ -z^{llf}_{depth}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ \vdots \end{bmatrix} \tag{2.64}$$

Each particle thus initializes its state vector by drawing a sample from a Gaussian distribution centered around the GPS position of the vehicle relative to the beacon, with standard deviation σ^2_{gps}. We omit the elements of the state vector that pertain to vehicle attitude and speed, since these values are irrelevant for the purposes of initialization. Importantly, note that whenever the AUV receives a valid GPS fix[20], the particles are re-initialized using the scheme in Eq. 2.64. Note also that the depth value provided by the vehicle's pressure sensor z^{llf}_{depth} increases with depth, which is opposed to our LLF, resulting in the negation. The particle weights are assigned an equal value $w_i = \frac{1}{R}$ at initialization.

Particle Filter Predict Step

After initialization and the AUV has dived and is no longer receiving GPS, each particle in the set is propagated using our motion model defined by Eqs. 2.60 and 2.61 – this is the predict step of the Bayesian filter. This is performed for each particle by first drawing a sample from each variable of the control space (heading α_{ahrs} and forward speed-over-ground v_{sog}) in order to update the corresponding relevant elements of the state space ($\alpha_{i,auv}$ and $v_{i,auv}$) – these same values are then used to propagate each particle:

[20]The AUV will only receive a valid GPS fix when it has surfaced, which generally only occurs if the vehicle stops and floats to the surface – GPS signals are not available underwater.

$$\boldsymbol{x}_i(t) = \begin{bmatrix} x^{vcf}_{i,beacon}(t) \\ y^{vcf}_{i,beacon}(t) \\ \vdots \\ x^{llf}_{i,auv}(t) \\ y^{llf}_{i,auv}(t) \\ z^{llf}_{i,auv}(t) \\ \gamma_{i,auv}(t) \\ \beta_{i,auv}(t) \\ \alpha_{i,auv}(t) \\ \\ v_{i,auv}(t) \\ \\ \end{bmatrix} = \begin{bmatrix} x^{vcf}_{i,beacon}(t-\Delta t) - v_{i,auv}(t-\Delta t)\cdot\Delta t\cdot\cos(\alpha_{i,auv}(t-\Delta t)) \\ y^{vcf}_{i,beacon}(t-\Delta t) - v_{i,auv}(t-\Delta t)\cdot\Delta t\cdot\sin(\alpha_{i,auv}(t-\Delta t)) \\ \vdots \\ x^{llf}_{i,beacon} - x^{vcf}_{i,beacon}(t) \\ y^{llf}_{i,beacon} - y^{vcf}_{i,beacon}(t) \\ -z^{llf}_{depth}(t) \\ \gamma_{ahrs}(t) \\ \beta_{ahrs}(t) \\ \alpha_{ahrs}(t) + \mathcal{N}(0,\sigma^2_\alpha) \\ v_{sog}(t) + \mathcal{N}(0,\sigma^2_v) = \\ RPM(t)\cdot 9.125\times 10^{-4}\cdot\cos(\beta_{ahrs}(t)) + \mathcal{N}(0,\sigma^2_v) \end{bmatrix} \tag{2.65}$$

As previously mentioned, the first two elements of particle's state vector are propagated using our motion model – the position of the AUV itself in the LLF can then be estimated using this relative beacon position and the LLF beacon position, as indicated by the reference frames in Fig. 2.13. Depth $z^{llf}_{i,auv}$, roll $\gamma_{i,auv}$ and pitch $\beta_{i,auv}$ are sensed directly by the AUVs depth pressure sensor[21] and AHRS, which are taken to be the 'true' values. The yaw $\alpha_{i,auv}$ is modeled as a Gaussian centered around the value provided by the AHRS (which is highly dependent on the magnetometer) with some standard deviation σ_α. Finally, the vehicle forward speed-over-ground is calculated using the pitch-compensated RPM-to-speed mapping of Eq. 2.1 – like yaw, it is modeled as a Gaussian centered around this value with some standard deviation σ_v. We omit elements of the state vector which do not evolve with time.

Particle Filter Update Step

The particle filter then continuously recurses between this predict step and an update step – this update step serves to correct the prediction using our acoustic range and angle measurement distributions generated via matched filtering and beamforming. This requires us to transform the particles into the domains in which these measurements are performed. We recognize an issue here: we have two independent measurements, range and angle, each within their own domains – how best do we incorporate this information? We have a few possible options:

- Combined domain: since range and angle are independent, we could produce a combined 3D range/angle domain, with axes of range, azimuth and inclination. This approach

[21] The depth sensor on a typical AUV is extremely accurate, with an error on the order of ±10 cm – depth is calculated from pressure using the Fofonoff equation [183].

is the most technically correct one, however, it results in a very large measurement domain, only very small portions of which would contain areas of high probability. Such a representation is inefficient, and especially problematic for a particle filter which suffers from the particle deprivation problem[22] – a large number of particles would be required to avoid degeneracy.

- Sequential: we could first transform particles into the range domain, incorporate the range measurement, then transform these partially-updated particles into the angle domain, then incorporate the angle measurement (or vice-versa). Unfortunately, this approach has the tendency to bias our state estimate. As we will explain later, the particle weights are multiplied against the measurement distribution at their position within the measurement domain. Consider a particle transformed into the range domain, where it lands into an area of high probability causing its weight to become large – there is no guarantee that this particle will fall within a similarly high probability area once transformed into the angle domain. Again, we would need a large number of particles to avoid biasing our estimate.

- Parallel (factored): we could duplicate the set of particles, transform one into the range domain and the other into the angle domain, incorporate measurements, and finally recombine them in some way. We use a variant of this approach, which, while not strictly correct, is fast and provides good results in practice.

To incorporate range and angle measurements in parallel at every filter update step, we first duplicate the set of particles, so that we have a range domain set $\underline{\overset{\text{rng}}{\boldsymbol{S}}}$ and an angle domain set $\underline{\overset{\text{agl}}{\boldsymbol{S}}}$, both of which are identical at the start of the update step:

$$\underline{\overset{\text{rng}}{\boldsymbol{S}}}(t) = \boldsymbol{S}(t) = \{\underline{\overset{\text{rng}}{\boldsymbol{s}}}_i(t), ..., \underline{\overset{\text{rng}}{\boldsymbol{s}}}_R(t)\} \quad : \quad \underline{\overset{\text{rng}}{\boldsymbol{s}}}_i(t) = \begin{bmatrix} \underline{\overset{\text{rng}}{\boldsymbol{x}}}_i(t) \\ \underline{\overset{\text{rng}}{w}}_i \end{bmatrix} \quad i = 1, ..., R \tag{2.66}$$

$$\underline{\overset{\text{agl}}{\boldsymbol{S}}}(t) = \boldsymbol{S}(t) = \{\underline{\overset{\text{agl}}{\boldsymbol{s}}}_i(t), ..., \underline{\overset{\text{agl}}{\boldsymbol{s}}}_R(t)\} \quad : \quad \underline{\overset{\text{agl}}{\boldsymbol{s}}}_i(t) = \begin{bmatrix} \underline{\overset{\text{agl}}{\boldsymbol{x}}}_i(t) \\ \underline{\overset{\text{agl}}{w}}_i \end{bmatrix} \quad i = 1, ..., R \tag{2.67}$$

Both of these duplicated particle sets contain the set of state estimates and their associated weights after performing the filter prediction step. The state estimate held by each particle in the range set $\underline{\overset{\text{rng}}{\boldsymbol{S}}}$ can be transformed into the range domain via:

[22] Particles in the particle filter tend to 'cluster' around areas of high probability – if the posterior estimate of the current state is a tightly clustered set of particles, and none of these particles exist within the high-probability portion of a new measurement distribution, the filter will fail to capture new measurement information and the state estimate will be incorrect. This is especially problematic if the measurement space is high-dimensional and only small portions of this domain contain areas of high probability.

$$\underline{\overline{\text{rng}}}r_i(t) = \sqrt{(\underline{\overline{\text{rng}}}x^{vcf}_{i,beacon}(t))^2 + (\underline{\overline{\text{rng}}}y^{vcf}_{i,beacon}(t))^2 + (\underline{\overline{\text{rng}}}z^{llf}_{i,beacon}(t) - \underline{\overline{\text{rng}}}z^{llf}_{i,auv}(t))^2} \tag{2.68}$$

$$= \sqrt{(\underline{\overline{\text{rng}}}x^{vcf}_{i,beacon}(t))^2 + (\underline{\overline{\text{rng}}}y^{vcf}_{i,beacon}(t))^2 + (-1 - \underline{\overline{\text{rng}}}z^{llf}_{i,auv}(t))^2} \tag{2.69}$$

where we use the $\underline{\overline{\text{rng}}}$ pre-superscript to indicate that this particle is a member of the duplicate set used to incorporate the range measurement. After this transformation, the particles represent a distribution in the range domain that must be multiplied against our range measurement distribution. For each particle, the range measurement is evaluated at the particle position, and the particle's weight is multiplied by the result of this evaluation:

$$\underline{\overline{\text{rng}}}w_i = \underline{\overline{\text{rng}}}w_i \cdot \hat{y}[\underline{\overline{\text{rng}}}r_i(t)] \tag{2.70}$$

where $\hat{y}[r]$ is a valid range measurement from acoustic processing[23] (algorithm 1). As with the range set, we can transform the state estimates of the particles in the angle set $\underline{\overline{\text{agl}}}\boldsymbol{S}$ into the BFF to incorporate the beamformed angle measurement. Referencing Fig. 2.13, we see that the transformation from the BFF to the VCF for a given particle is performed by:

$$T^{vcf}_{i,bff}(t) = R_z(\alpha_{i,auv}(t))R_y(\beta_{i,auv}(t))R_x(\gamma_{i,auv}(t)) \tag{2.71}$$

$$\text{WHERE} \quad (\text{using } s(\cdot) := \sin(\cdot) \text{ and } c(\cdot) := \cos(\cdot))$$

$$R_z(\theta) = \begin{bmatrix} c(\theta) & -s(\theta) & 0 \\ s(\theta) & c(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} c(\theta) & 0 & -s(\theta) \\ 0 & 1 & 0 \\ s(\theta) & 0 & c(\theta) \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\theta) & -s(\theta) \\ 0 & s(\theta) & c(\theta) \end{bmatrix} \tag{2.72}$$

The position of the beacon relative to the AUV as estimated by the particle in the BFF is thus given by:

$$\begin{bmatrix} \underline{\overline{\text{agl}}}x^{bff}_{i,beacon}(t) \\ \underline{\overline{\text{agl}}}y^{bff}_{i,beacon}(t) \\ \underline{\overline{\text{agl}}}z^{bff}_{i,beacon}(t) \end{bmatrix} = (T^{vcf}_{i,bff}(t))^T \begin{bmatrix} \underline{\overline{\text{agl}}}x^{vcf}_{i,beacon}(t) \\ \underline{\overline{\text{agl}}}y^{vcf}_{i,beacon}(t) \\ \underline{\overline{\text{agl}}}z^{llf}_{i,beacon}(t) - \underline{\overline{\text{agl}}}z^{llf}_{i,auv}(t) \end{bmatrix} \tag{2.73}$$

and the particle's position in the angle domain is finally given by the transformation from Cartesian to spherical coordinates:

[23] Of course, since the range measurement has a discretized domain, evaluating it at an arbitrary range will return the measurement at the range closest to the input.

$$\underline{\overset{\text{agl}}{\theta}}_i(t) = \arccos\left(\frac{\underline{\overset{\text{agl}}{z}}^{bff}_{i,beacon}(t)}{\sqrt{(\underline{\overset{\text{agl}}{x}}^{bff}_{i,beacon}(t))^2 + (\underline{\overset{\text{agl}}{y}}^{bff}_{i,beacon}(t))^2 + (\underline{\overset{\text{agl}}{z}}^{bff}_{i,beacon}(t))^2}}\right) \tag{2.74}$$

$$\underline{\overset{\text{agl}}{\phi}}_i(t) = \arctan\left(\frac{\underline{\overset{\text{agl}}{y}}^{bff}_{i,beacon}(t)}{\underline{\overset{\text{agl}}{x}}^{bff}_{i,beacon}(t)}\right) \tag{2.75}$$

where we use the $\underline{\overset{\text{agl}}{\cdot}}$ pre-superscript to indicate that this particle is a member of the duplicate set used to incorporate the angle measurement. As with the case for incorporating range, each particle's weight in this set is updated by multiplying it against the result of the evaluation of the angle measurement at the particle's position:

$$\underline{\overset{\text{agl}}{w}}_i = \underline{\overset{\text{agl}}{w}}_i \cdot |\hat{Z}[\underline{\overset{\text{agl}}{\theta}}_i(t), \underline{\overset{\text{agl}}{\phi}}_i(t)]|^2 \tag{2.76}$$

where $|\hat{Z}[\theta, \phi]|^2$ is a valid angle measurement distribution from acoustic processing[24] (algorithm 1). After incorporating the range and angle measurement distributions, the weights of their respective particle sets are renormalized such that $\sum_{i=1}^{R} \underline{\overset{\text{rng}}{w}}_i = 1$ and $\sum_{i=1}^{R} \underline{\overset{\text{agl}}{w}}_i = 1$, since these particles as a whole represent a probability distribution. These duplicated particle sets must now be recombined into a single set of particles in order to perform the predict step at the next filter cycle:

$$\text{LET} \quad \underline{\overset{\text{rng}}{\boldsymbol{S}}}(t)^{\uparrow} = \underline{\overset{\text{rng}}{\boldsymbol{S}}}(t) \quad \text{SUCH THAT} \quad \underline{\overset{\text{rng}}{w}}_{i-1} \leq \underline{\overset{\text{rng}}{w}}_i \leq \underline{\overset{\text{rng}}{w}}_{i+1} \tag{2.77}$$

$$\text{LET} \quad \underline{\overset{\text{agl}}{\boldsymbol{S}}}(t)^{\uparrow} = \underline{\overset{\text{agl}}{\boldsymbol{S}}}(t) \quad \text{SUCH THAT} \quad \underline{\overset{\text{agl}}{w}}_{i-1} \leq \underline{\overset{\text{agl}}{w}}_i \leq \underline{\overset{\text{agl}}{w}}_{i+1} \tag{2.78}$$

The duplicated sets are first reordered in terms of ascending weights. The corresponding reordered range and angle particles are then element-wise multiplied and transformed back into the VCF to produce the new set of particles for the next motion model update:

[24] Given the discretized nature of the beamformed angle measurement, evaluating it at a desired azimuth ϕ and inclination θ will return the beamformed power measurement at the angle closest to this input.

$$\begin{bmatrix} x^{vcf}_{i,beacon}(t) \\ y^{vcf}_{i,beacon}(t) \\ -- \end{bmatrix} = T^{\ vcf}_{i,bff}(t) \begin{bmatrix} \overline{\overline{\text{rng}}}r_i(t)^{\uparrow} \cdot \sin(\overline{\overline{\text{agl}}}\theta_i(t)^{\uparrow}) \cdot \cos(\overline{\overline{\text{agl}}}\phi_i(t)^{\uparrow}) \\ \overline{\overline{\text{rng}}}r_i(t)^{\uparrow} \cdot \sin(\overline{\overline{\text{agl}}}\theta_i(t)^{\uparrow}) \cdot \sin(\overline{\overline{\text{agl}}}\phi_i(t)^{\uparrow}) \\ \overline{\overline{\text{rng}}}r_i(t)^{\uparrow} \cdot \cos(\overline{\overline{\text{agl}}}\theta_i(t)^{\uparrow}) \end{bmatrix} \tag{2.79}$$

$$w_i = \frac{\overline{\overline{\text{rng}}}w_i^{\uparrow} \cdot \overline{\overline{\text{agl}}}w_i^{\uparrow}}{\sum_{i=1}^{R} \overline{\overline{\text{rng}}}w_i^{\uparrow} \cdot \overline{\overline{\text{agl}}}w_i^{\uparrow}} \tag{2.80}$$

where we use the $^{\uparrow}$ symbol to indicate that the particles and weights have ascending order. Notice that the particle z coordinate in the LLF remains unchanged – it is not updated using the acoustic measurement, because calculating it using the AUV depth sensor is much more accurate. This element-wise multiplication of ordered range and angle particles produces a new set of particles in the VCF, with the particles more suitably weighted by the combined range and angle measurement; while this approach is not strictly correct, it allows us to maintain a constant R particles efficiently during duplication and recombination and performs well in practice. These series of calculations incorporates our acoustic measurements, and represent the complete particle filter update step.

Particle Filter Systematic Resampling

At every update step particle weights are reevaluated and multiplied against the measurement distributions – the weight of a particle w_i represents the likelihood of the particle's state hypothesis $\boldsymbol{x}_i$. As such, after a series of updates, the distribution of weights across all particles can become very uneven, with a few particles of high weight (strong hypotheses) and a majority with very low weight (weak hypotheses). This is a critical issue with particle filtering, as it can drastically bias our state estimate and reduce filter performance. We prevent degradation of filter performance by carrying out a resampling step after the update step and before the next predict step – the goal of particle resampling is to represent a single highly weighted particle by a number of equally weighted particles at the original particle position [184]. By resampling, we maintain a set of particles that have a more even distribution of weights. In our case, we make use of the *systematic resampling* scheme, for its computational simplicity and because it has been empirically demonstrated to have good performance [184].

Particle Filter Likelihood Estimation

The set of particles that represent the probability distribution of the position of the beacon relative to the vehicle in the VCF must be converted into a single estimate for the purposes of vehicle control. One possible approach is to simply use the state estimate of the most highly weighted particle, but this approach tends to result in trajectories that are qualitatively

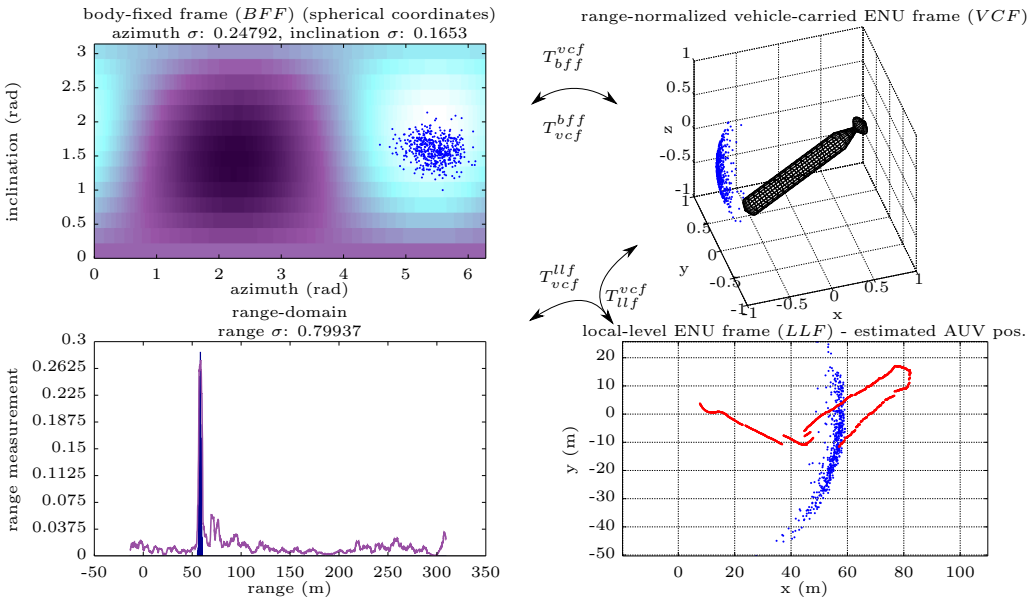

Figure 2.14: Visualization of the piUSBL particle filter in action – *Top Left*: the power output from CBF is shown, along with the duplicated set of particles $\overset{\text{agl}}{\boldsymbol{S}}$ that has been transformed into the angle domain to incorporate the angle measurement. *Bottom Left*: the range measurement distribution from matched filtering is shown in purple, along with a histogram of the duplicated set of particles $\overset{\text{rng}}{\boldsymbol{S}}$ used to incorporate the range measurement in blue. *Top Right*: a visualization of the range-normalized position of the beacon relative to the AUV in the VCF from particle state estimates. *Bottom Right*: a visualization of the relative position of the beacon in the LLF from particle state estimates, as well as their weighted mean over time shown in red.

'rough' or inconsistent. We instead use the weighted mean of the particles as the estimated relative position of the beacon, and use the weighted covariance as a measure of uncertainty in our filter estimate:

$$\boldsymbol{\mu}^{vcf}_{beacon}(t) = \begin{bmatrix} \sum_{i=1}^{R} w_i \cdot x^{vcf}_{i,beacon}(t) \\ \sum_{i=1}^{R} w_i \cdot y^{vcf}_{i,beacon}(t) \end{bmatrix} \tag{2.81}$$

$$\boldsymbol{\Sigma}^{vcf}_{beacon}(t) = \frac{1}{R-1} \sum_{i=1}^{R} w_i \left(\begin{bmatrix} x^{vcf}_{i,beacon}(t) \\ y^{vcf}_{i,beacon}(t) \end{bmatrix} - \boldsymbol{\mu}^{vcf}_{beacon}(t) \right) \left(\begin{bmatrix} x^{vcf}_{i,beacon}(t) \\ y^{vcf}_{i,beacon}(t) \end{bmatrix} - \boldsymbol{\mu}^{vcf}_{beacon}(t) \right)^{T} \tag{2.82}$$

This concludes our explanation of the structure of the particle filter specifically designed for our prototypical piUSBL system. The filter is summarized in algorithm 3, which, for simplicity, references many of the previous equations. The particle filter is also visually represented in Fig. 2.14.

Algorithm 3 The piUSBL particle filter

procedure PARTICLE_FILTER($p(\boldsymbol{x}_{gps}^{llf}), p(\boldsymbol{z}_{depth}^{llf}), \gamma_{ahrs}, \beta_{ahrs}, p(\alpha_{ahrs}), p(v_{sog}), x_i \;:\; i = 1, ..., N$)
▷ Continuously estimates the relative position of the beacon in the VCF using our AUV motion model and acoustic range and angle measurements
▷ Inputs: $p(\boldsymbol{x}_{gps}^{llf})$: GPS distribution, $p(\boldsymbol{z}_{depth}^{llf})$: depth measurement distribution, γ_{ahrs}: roll measurement, β_{ahrs}: pitch measurement, $p(\alpha_{ahrs})$: yaw measurement distribution, $p(v_{sog})$: forward speed-over-ground distribution, x_i: measured acoustic signals
▷ Calculates: $\boldsymbol{\mu}_{beacon}^{vcf}$: relative beacon position estimate, $\boldsymbol{\Sigma}_{beacon}^{vcf}$: state estimate covariance
  for $i = 1 : R$ **do** ▷ Initialize particles
    Initialize particle $\boldsymbol{s}_i$ state $\boldsymbol{x}_i$ using Eq. 2.64 and weight $w_i = \frac{1}{R}$
  end for
  loop
    for $i = 1 : R$ **do** ▷ Predict step
      Propagate particle $\boldsymbol{s}_i$ state $\boldsymbol{x}_i$ using Eq. 2.65
    end for
    $\boldsymbol{range}, \boldsymbol{angle}$ = PROCESS_ACOUSTICS(...) ▷ Algorithm 1 (loop portion only)
    if valid acoustic measurements returned from algorithm 1 **then**
      Duplicate particle set $\boldsymbol{S}$ into range particle set $\overset{\text{rng}}{\frown}\boldsymbol{S}$ using Eq. 2.66
      Duplicate particle set $\boldsymbol{S}$ into angle particle set $\overset{\text{agl}}{\frown}\boldsymbol{S}$ using Eq. 2.67
      for $i = 1 : R$ **do** ▷ Update step
        Incorporate acoustic range measurement distribution for this particle using Eqs. 2.69 & 2.70
        Incorporate acoustic angle measurement distribution for this particle using Eqs. 2.73, 2.74, 2.75 & 2.76
      end for
      Reorder $\overset{\text{rng}}{\frown}\boldsymbol{S}$ using ascending weights with Eq. 2.77
      Reorder $\overset{\text{agl}}{\frown}\boldsymbol{S}$ using ascending weights with Eq. 2.78
      for $i = 1 : R$ **do** ▷ Recombine $\overset{\text{rng}}{\frown}\boldsymbol{S}$ and $\overset{\text{agl}}{\frown}\boldsymbol{S}$ to produce new particles for $\boldsymbol{S}$
        Project particle range $\overset{\text{rng}}{\frown}r_i^{\uparrow}$ and angle $\overset{\text{agl}}{\frown}\theta_i^{\uparrow}, \overset{\text{agl}}{\frown}\phi_i^{\uparrow}$ into new $\begin{bmatrix} x_{i,beacon}^{vcf} \\ y_{i,beacon}^{vcf} \end{bmatrix}$ using Eq. 2.79
        Update new particle weight w_i using Eq. 2.80
      end for
      Perform systematic resampling of particles ▷ Resample step
    end if
    $\boldsymbol{\mu}_{beacon}^{vcf} = \begin{bmatrix} \sum_{i=1}^{R} w_i \cdot x_{i,beacon}^{vcf} \\ \sum_{i=1}^{R} w_i \cdot y_{i,beacon}^{vcf} \end{bmatrix}$ ▷ Estimate step
    $\boldsymbol{\Sigma}_{beacon}^{vcf} = \frac{1}{R-1} \sum_{i=1}^{R} w_i \left(\begin{bmatrix} x_{i,beacon}^{vcf} \\ y_{i,beacon}^{vcf} \end{bmatrix} - \boldsymbol{\mu}_{beacon}^{vcf} \right) \left(\begin{bmatrix} x_{i,beacon}^{vcf} \\ y_{i,beacon}^{vcf} \end{bmatrix} - \boldsymbol{\mu}_{beacon}^{vcf} \right)^T$
  end loop
end procedure

Considerations on the Factored Nature of the piUSBL Particle Filter

The separation of the piUSBL particle filter into two parallel update steps, one for the incorporation of range measurements and the other for the incorporation of angle measurements to the acoustic beacon, essentially represents a *factorization* of the state space; thus, the piUSBL particle filter can be described as a *factored* particle filter. This approach is strictly an *approximation* of the more conventional particle filter, which would operate using a single set of particles within the full state space that combines both the range and angle domain.

The decision to make use of a factored particle filter was motivated by the need to achieve real-time processing rates on-board the vehicle, by reducing the number of particles in the filter. In fact, the structure of update step in the piUSBL particle filter in some sense allows us to achieve the same dynamic range in particle weights that would have been present if we were to use a larger number of particles within the conventional particle filter. Consider a particle that sits at the global maximum within the combined range and angle domain; this particle can be represented by two separate particles in the range domain and the angle domain, both sitting at the respective maxima of those domains – because our approach rearranges the parallel sets of particles into ascending order, when element-wise multiplying the two sets of particle weights together during the update step, the resulting *combined* particle would have the same weight as the theoretical particle in the combined domain of the conventional particle filter. Thus, in a sense our factored particle filter uses a subset of particles that would be present in the conventional particle filter – with the advantage that this subset represents a significant reduction in the number of particles required to sample the full range of values present in the combined measurement space.

Although it is an approximation of the conventional particle filter, our piUSBL particle filter works very well in practice, as will be apparent from results shown later in this chapter as well as subsequent chapters. However, because we do not provide results using the same system with a conventional particle filter, it is difficult to determine whether the factored nature of the filter represents a loss in accuracy in comparison to a conventional filter – conceptually, it appears intuitive that sampling the range and angle measurement spaces separately and combining the parallel particles in an ordered manner would maintain the dynamic range of a conventional particle filter, further investigation is necessary to confirm this intuition definitively. This can be performed in the future by simply reprocessing experimental data using a conventional particle filter; for this work, we simply note that the current factored implementation performs adequately for our purposes.

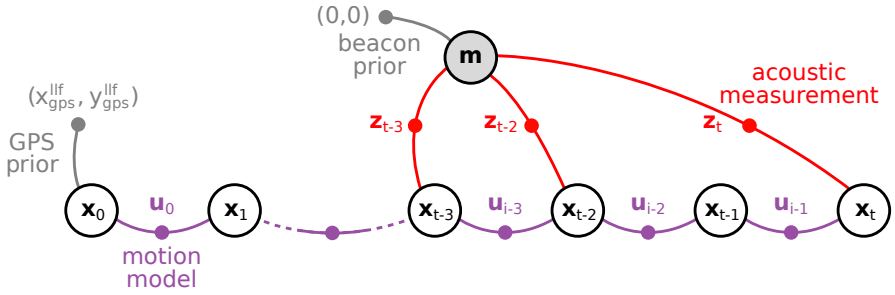

Figure 2.15: Illustration of factor graph used for smoothing in the prototypical piUSBL system – $\boldsymbol{x}_i$ are AUV states (poses) in the LLF connected by motion model factors using control inputs $\boldsymbol{u}_i$; $\boldsymbol{m}$ is the map, which in the case for piUSBL is simply the state of the beacon in the LLF, which is connected by a prior factor to $(0, 0)$; vehicle states are connected to the beacon via acoustic measurement factors $\boldsymbol{z}_i$, which can either be range-only, or range-plus-bearing measurements; the initial vehicle pose has a prior factor from GPS $(x^{llf}_{gps}, y^{llf}_{gps})$. Note that for the smoothing problem we are solving for the state of the AUV in the 2D local-level frame (i.e. no depth solving); in addition, not all vehicle states are connected to the map, since valid acoustic measurements are not guaranteed at every timestep.

2.6 Smoothing (Factor Graph SLAM)

Particle filtering provides the vehicle with a current estimate of the relative position of the beacon, as well as the uncertainty of this estimate; however, it does so by utilizing a recursive Bayesian filtering structure, in which the vehicle trajectory is modeled as a HMM with the Markov assumption, meaning that the current state is dependent only on the previous state, and the current measurement is only dependent on the current state. In effect, it operates by marginalizing out all previous measurements – the current state incorporates all information from the measurement, and is then used to inform the next state. Because the state at each time step marginalizes out the measurement, the resulting trajectory is often discontinuous.

Smoothing on the other hand utilizes all previous measurements to optimize over the entire history of states, so that the resulting state trajectory over all time $\boldsymbol{x}_{0:t}$ optimally 'explains' its history of control inputs $\boldsymbol{u}_{0:t}$ and observed measurements $\boldsymbol{z}_{0:t}$. As a result, the optimized full vehicle trajectory is smooth. Smoothing is also often referred to as *full simultaneous localization and mapping (SLAM)*, since the measurements come from a common map $\boldsymbol{m}$ – thus, optimizing over the trajectory, which includes map measurements that are necessarily connected over time (and as such are constrained), results in a trajectory that not only provides the best explanation of the vehicle state over time given the measurements, but simultaneously provides the best explanation of the measurements (and thus the map) given the vehicle state over time. We wish to calculate the posterior distribution over the entire vehicle trajectory $\boldsymbol{x}_{0:t}$, as well as the map $\boldsymbol{m}_{0:t}$ [175]:

$$p(\boldsymbol{x}_{0:t}, \boldsymbol{m}_{0:t} | \boldsymbol{z}_{0:t}, \boldsymbol{u}_{0:t}) = p(\boldsymbol{x}_{0:t}, \boldsymbol{m} | \boldsymbol{z}_{0:t}, \boldsymbol{u}_{0:t}) \tag{2.83}$$

where we remove the map's dependence on time, since it is usually assumed to be static. To visualize this optimization problem for our particular case, we formulate it as a *factor graph* in the LLF, as shown in Fig. 2.15 [185].

Unlike the Bayesian filtering case, where we are estimating the *relative* position of the beacon in the vehicle-carried frame, for smoothing we formulate the problem differently: we estimate the *absolute* vehicle pose, $\boldsymbol{x}_t = [x_t, y_t, \alpha_t]^T$, in the 2-dimensional local-level frame using a factor graph smoothing framework to represent the collection of poses over the entire trajectory. Each node $\boldsymbol{x}_t$ in the graph corresponds to the pose estimate at time t, and is linked to preceding and subsequent pose nodes by factors calculated using our motion model with input $\boldsymbol{u}_t$ – this motion model differs slightly to the one used for particle filtering:

$$x_t = x_{t-1} + v_{t-1} \cdot \Delta t \cdot \cos(\alpha_{t-1}) \tag{2.84}$$

$$y_t = y_{t-1} + v_{t-1} \cdot \Delta t \cdot \sin(\alpha_{t-1}) \tag{2.85}$$

The initial pose $\boldsymbol{x}_0$ is constrained by a prior factor, which represents surface GPS measurements in the LLF, and the acoustic beacon node (our map in this case) is also constrained by a prior factor that places it at the arbitrary location of our choosing in the LLF, $(0, 0)$. When a valid acoustic measurement is outputted by acoustic processing, we use the result of the particle filter to link the pose at which the measurement occurred to the beacon node, by calculating the weighted mean range and bearing from the particles, as well as the corresponding variances:

$$\boldsymbol{z}_t = \begin{bmatrix} r_t & : & \text{2D range between AUV and beacon} \\ \zeta_t & : & \text{bearing angle from AUV to beacon} \end{bmatrix} \qquad \boldsymbol{\sigma}_{t,\boldsymbol{z}} = \begin{bmatrix} \sigma_{t,r} \\ \sigma_{t,\zeta} \end{bmatrix} \tag{2.86}$$

$$r_t = \sum_{i=1}^{R} w_i \sqrt{(x^{vcf}_{i,beacon}(t))^2 + (y^{vcf}_{i,beacon}(t))^2} \tag{2.87}$$

$$\zeta_t = \sum_{i=1}^{R} \left(w_i \cdot \arctan\left(\frac{y^{vcf}_{i,beacon}(t)}{x^{vcf}_{i,beacon}(t)} \right) \right) - \alpha_t \tag{2.88}$$

$$\sigma_{t,r} = \frac{1}{R-1} \sum_{i=1}^{R} w_i \left(\sqrt{(x^{vcf}_{i,beacon}(t))^2 + (y^{vcf}_{i,beacon}(t))^2} - r_t \right)^2 \tag{2.89}$$

$$\sigma_{t,\zeta} = \frac{1}{R-1} \sum_{i=1}^{R} w_i \left(\left(\arctan\left(\frac{y^{vcf}_{i,beacon}(t)}{x^{vcf}_{i,beacon}(t)} \right) - \alpha_t \right) \ominus \zeta_t \right)^2 \tag{2.90}$$

where we define the $\ominus$ symbol as the circular subtraction. These values are used to inform the factor between the relevant pose node and the beacon node. Formulating the optimization like this over the AUV states in the 2D LLF is simpler and more intuitive than smoothing over the relative beacon position, especially because in this case the beacon is static – this

means that the relative beacon position and the absolute LLF AUV position are equivalent except for negation.

We use the GTSAM[25] library [186], and specifically the iSAM2[26] algorithm [187] to incrementally perform maximum a-posteriori inference over our factor graph as it is constructed. Motion and measurement noise are independent and assumed to be Gaussian, with the standard deviation of measurement noise estimated directly using the particles from our filter, as in Eq. 2.86. In terms of GTSAM nomenclature, AUV pose nodes are added to the factor graph as *Pose2*, and the beacon node is added as a *Point2*; the priors to the beacon and initial pose are added each as a *PriorFactor*. Motion model factors are added as *BetweenFactor*, and measurement factors are added as *BearingRangeFactor*. The noise values for the priors and motion model are set as:

$$\sigma_{\boldsymbol{x}} = \begin{bmatrix} 2\text{ m} & 0 & 0 \\ 0 & 2\text{ m} & 0 \\ 0 & 0 & 5^\circ \end{bmatrix} \quad \sigma_{\boldsymbol{m}} = \begin{bmatrix} 0.1\text{ m} & 0 \\ 0 & 0.1\text{ m} \end{bmatrix} \quad \sigma_{t,\boldsymbol{u}} = \begin{bmatrix} 0.5\text{ m s}^{-1} & 0 & 0 \\ 0 & 0.1\text{ m s}^{-1} & 0 \\ 0 & 0 & 0.2^\circ\text{s}^{-1} \end{bmatrix} \tag{2.91}$$

where $\sigma_{\boldsymbol{x}}$ is the noise on the initial pose prior, $\sigma_{\boldsymbol{m}}$ is the noise on the beacon prior, and $\sigma_{t,\boldsymbol{u}}$ is motion model noise in surge, sway and yaw. GTSAM and the ISAM2 algorithm are state-of-the-art in factor graph SLAM. However, note that it only supports unimodal Gaussian distributions – as a result, we are forced to approximate our state and measurement distributions as Gaussians using the output of our particle filter. This is not strictly correct, since we are essentially removing vital independent assumptions between states and measurements via the filter, but it serves our purposes of investigating the utility of smoothing in our application in a straightforward and simple manner.

2.7 Experimental Results

To evaluate the system stack of our prototypical piUSBL system, we performed experiments using the prototype SandShark AUV with piUSBL payload (subsection 2.3.3) on a portion of the Charles River by the Massachusetts Institute of Technology (MIT) Sailing Pavilion (Fig. 2.16) in the Fall of 2016. Our acoustic beacon (subsection 2.3.1) was submerged to about 1 m depth and fastened to the pavilion dock, at a known GPS position. The AUV was pre-programmed with a mission where it was instructed to travel back-and-forth along the dock for 70 m at 2 m depth and a speed of 1.4 m s^{-1}. The mission duration was set to 1200 s, and the AUV was instructed to surface for a 120 s GPS fix whenever it was at the end of a

[25] https://bitbucket.org/gtborg/gtsam/
[26] https://github.com/nicrip/pygtsam/

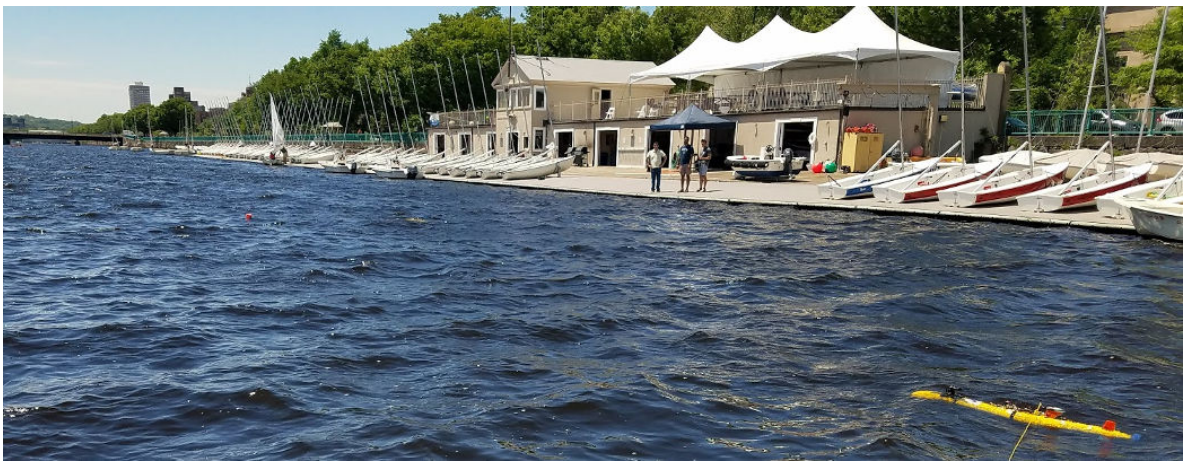

Figure 2.16: Photograph of the Massachusetts Institute of Technology (MIT) Sailing Pavilion on the Charles River, with a commercial SandShark AUV in the water.

70 m run and the time since the last fix was greater than 300 s for runs 1 and 4, 480 s for runs 2 and 5, and 660 s for run 3.

Acoustic processing (section 2.4), including both matched filtering for range and beamforming for angle were performed on-board the Raspberry Pi 3 in real-time at approximately 1.25 Hz. For the CBF (subsection 2.4.2), we selected the number of azimuths $N_{azim} = 270$ and the number of inclinations $N_{incl} = 15$ (for a total of 4050 look-angles), since a higher azimuthal angular resolution was more critical. We placed an additional check of validity on acoustic measurements due to vehicle self-occlusion: if the MLE value of azimuth from beamforming was $90° \leq \phi \leq 270°$, then the angle acoustic measurement was discarded[27]. This data was recorded by the payload along with pre-filtered navigation data from the vehicle, which was received by the payload at a rate of about 10 Hz. The payload and AUV system clocks were synchronized using an NTP server running on the payload.

Particle filtering (subsection 2.5.2) with 1500 particles and factor graph smoothing (section 2.6) were performed offline. iSAM2 was used to build and solve the factor graph with vehicle poses added at a rate of 5 Hz and using ranges and bearings output by the particle filter for measurement factors between vehicle poses and the beacon pose using Eq. 2.86. A new graph was initialized each time the AUV received a GPS fix, allowing us to monitor the difference in estimated and true position during the underwater to surface transition.

The AUV was deployed for a total of five runs, with the vehicle surfacing for two GPS fixes for runs 1, 2, 3 and 5, and for three GPS fixes for run 4. For the five runs we use a simple metric to assess the inter-GPS-fix navigation performance of dead reckoning, particle filtering,

[27]This check basically deems an angle acoustic measurement to be valid if the acoustic energy incident onto the array came from in front of the AUV – this is necessary, since the array is 'blind' to acoustic energy coming from behind due to its position at the nose of the vehicle.

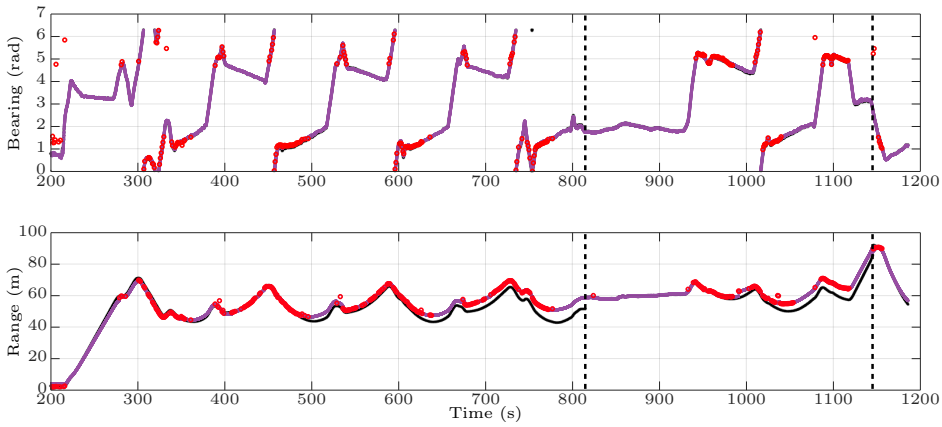

Figure 2.17: Range and bearing between the AUV and the acoustic beacon as output by dead reckoning and particle filtering using the prototypical piUSBL system for run 5 – bearing and range estimates from the AUV to the beacon in the LLF are shown for dead reckoning (black), particle filtering (purple), and raw acoustic measurement MLE values (red), with dashed vertical lines indicating AUV surfacing times for GPS fixes.

and factor graph smoothing over all five runs: during the underwater to surface transition a discontinuity in position occurs when the AUV gets a GPS fix, which is caused by localization error during underwater navigation; smaller jumps would indicate better performance. Unfortunately, this metric is subject to GPS positional error.

For the fifth and final run, two commercial long baseline (LBL) beacons from Hydroid[28] were deployed, one at each end of the dock at known GPS positions and a depth of 1 m. These LBL beacons respond to acoustic queries transmitted by the AUV via its WHOI micro-modem, allowing the vehicle to calculate two-way travel-time to each LBL beacon. These times are stored by the vehicle and post-processed using trilateration to calculate vehicle position as a means of validating our prototypical piUSBL system. The micro-modem is not used for any other purpose than to query the commercial LBL beacons. Unfortunately, the LBL beacons are subject to the same undesirable acoustic effects that cause measurement outliers.

Plots of bearing and range from the AUV to the beacon for run 5 are shown in Fig. 2.17, which plot these values as estimated from dead reckoning (black) and particle filtering (red), along with MLE measurements from our range and angle measurement distributions as purple circles. We see that the particle filter successfully fuses the observed acoustic measurements with the dead reckoned motion model, pulling the estimate towards observations. Our validity checks are apparent when looking at the measurements – no bearing measurements exist between 90° and 270°, which are invalid due to self-occlusion. Even though outliers exist in the measurements, these are successfully filtered out by the particle filter.

[28] https://www.hydroid.com/

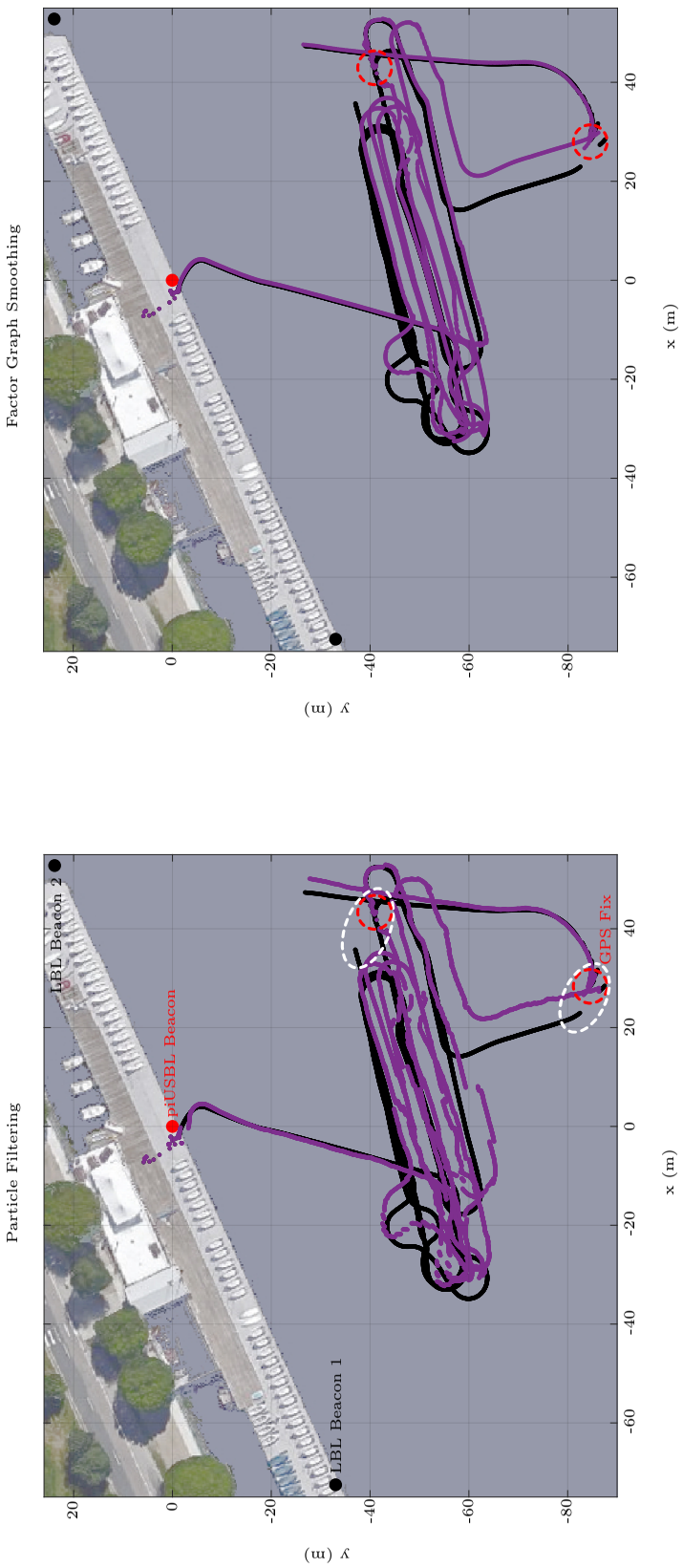

Figure 2.18: AUV trajectories as estimated by dead-reckoning, particle filtering and factor graph smoothing in the LLF for run 5 – the piUSBL beacon is shown as a red dot, and the Hydroid LBL beacons as the black dots affixed to the dock; locations of GPS surfacing events are shown as dashed red circles, and the jump in position for dead-reckoning during these events are highlighted by dashed white ellipses; the trajectory from dead-reckoning is shown in black in both plots. *Left:* the trajectory from particle filtering is shown in purple. *Right:* the trajectory from factor graph smoothing is shown in purple.

	GPS Jumps (m)										
	Run 1		**Run 2**		**Run 3**		**Run 4**			**Run 5**	
AUV Surface Time (s)	**550**	**1020**	**675**	**1250**	**840**	**1240**	**360**	**785**	**1210**	**920**	**1240**
Dead Reckon	13.8	7.0	17.0	10.2	18.4	17.6	2.7	2.7	4.4	7.4	6.1
Particle Filter	3.5	13.4	1.6	8.8	1.0	4.7	5.8	2.2	5.6	0.4	1.5
Factor Graph	1.3	3.1	1.5	11.5	0.3	4.9	1.4	3.8	2.5	1.0	3.8

	μ **GPS Jumps (m)**	σ **GPS Jumps (m)**
Dead Reckon	9.8	6.0
Particle Filter	4.4	3.9
Factor Graph	3.2	3.1

Table 2.1: Localization performance of the prototypical SandShark piUSBL system as measured using GPS discontinuities – the jump in GPS position when the AUV makes the underwater to surface transition is indicative of the performance of underwater navigation, with smaller jumps indicating better performance; a comparison of dead reckoning, particle filtering and factor graph smoothing indicates that filtering and smoothing improve underwater positioning.

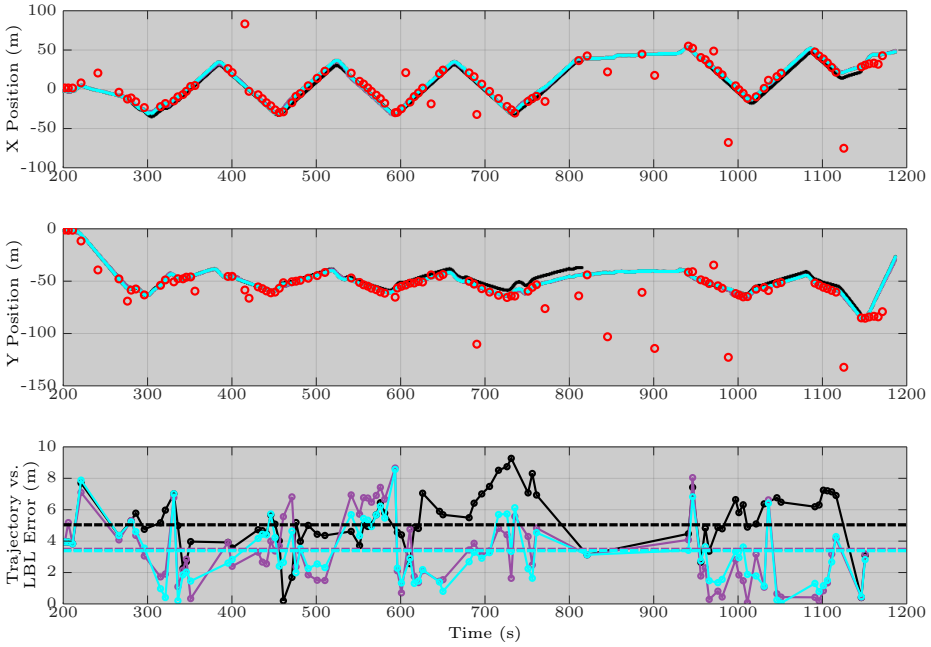

Figure 2.19: Trajectories resulting from dead reckoning, particle filtering and factor graph smoothing vs. commercial LBL position estimates – *Top*: the x-position estimate as calculated by dead-reckoning (black), particle filtering (purple) and factor graph smoothing (cyan), vs. LBL x-position estimates from trilateration as red circles. *Middle*: Similarly for y-position estimates. *Bottom*: Euclidean distance between the trajectories from dead-reckoning (black), particle filtering (purple) and factor graph smoothing (cyan) against the corresponding LBL estimates; the mean of this distance is shown for each approach as the horizontal dashed line.

The resulting trajectories of dead-reckoning, particle filtering and factor graph smoothing are shown for run 5 in Fig. 2.18 and is illustrative of typical trajectories resulting from our approach. During this run, the commercial LBL beacons were deployed. Dead reckoning is shown in black, and particle filtering and factor graph smoothing are shown in purple. The two GPS fixes for this run occur at (39 m, −42 m) (at approximately 814 s into the mission), and (−28 m, −84 m) (at approximately 1145 s into the mission). The positional jumps that occur during these GPS fixes are listed in table 2.1, along with the jumps that occur during the GPS fixes in runs 1–4, as well as the average jump distance and corresponding standard deviation. Qualitative examination of the trajectories indicate that the dead reckoned estimates are the least self-consistent, with large discontinuities when the vehicle surfaces for a fix (as highlighted by the dashed white ellipses). The particle filter trajectories are better in this respect, but they suffer from non-continuity caused by incorporation of latest observations in the filter's recursive estimate. In contrast, the trajectories resulting from iSAM2 factor graph smoothing are both self-consistent and maintain a smooth, continuous trajectory between GPS fixes; this is a result of optimizing over the entire vehicle history, incorporating all

acoustic measurements. These observations are supported by the discontinuity distances in table 2.1 – the iSAM2 approach has both the smallest average discontinuity, and the lowest standard deviation.

Besides the inherent positional uncertainty associated with GPS measurements, there are other possible sources for the observed differences between GPS and our localization approach. These include motion due to river currents, which cannot be accounted for, and, more importantly, the jitter in the acoustic beacon broadcast; recall in subsection 2.3.1 that the beacon has a 1 ms jitter, which, at a sound speed of 1481 m/s, corresponds to a 1.5 m error, which is on the same order of magnitude as the differences observed.

Finally, Fig. 2.19 displays x-position and y-position plots for the trajectories resulting from the three methods for run 5, with dead-reckoning in black, particle filtering in red, and factor graph smoothing in cyan, along with an estimate of the vehicle position as calculated by LBL trilateration in purple. Firstly, we note that the LBL estimates are highly susceptible to undesirable acoustic effects caused by the challenging acoustic environment of the Charles River, resulting in a number of outliers. Secondly, we notice that both particle filtering and factor graph smoothing more consistently match the LBL estimates, thereby providing strong validation for our piUSBL localization approach. This is supported by the bottom plot in Fig. 2.19, which shows the error between the trajectories outputted by the three methods and the LBL estimates over time - the average error is 5.04 m, 3.48 m, and 3.39 m for dead-reckoning, particle filtering, and factor graph factor graph smoothing respectively. These results are consistent with the GPS jump metrics.

2.8 Lessons Learned

The initial results obtained from experiments with the prototype SandShark AUV outfitted with the piUSBL system in the Fall of 2016 proved to be very promising, and provided us with a raft of insights into how piUSBL could be improved. The basic processing pipeline was demonstrated to operate effectively, resulting in a positioning system that is small, low-cost, easy to deploy, and scalable to many vehicles, making it an ideal solution for inexpensive and miniature autonomous underwater vehicles. In addition, the efficacy of the piUSBL approach in terms of positioning accuracy was established, with the system having shown that it could improve localization accuracy substantially. Four main lessons were learned from these initial experimental experiences:

- The mounting scheme of the USBL array on the prototype SandShark AUV is highly non-ideal, and limits performance of piUSBL positioning due to one simple fact: the vehicle is simply unable to receive acoustic energy from the half-space behind the vehicle, because acoustic signals from that direction are occluded by the body of the vehicle. As

a result, the filter *loses track* of the acoustic beacon whenever the AUV is traveling away from it – in effect, during these periods no acoustic measurements are incorporated by the filter or smoother, since they are deemed invalid, causing the state estimate to drift. The reduction in the number of valid acoustic measurements is a significant limitation for filter performance, and should be avoided as much as possible since performance is now tied to the behavior of the vehicle (i.e. performance improves when the AUV is pointing toward the beacon, and reduces when it is pointing away). The loss of acoustic measurements in these regimes is clearly visible in Fig. 2.17. An improvement in performance can be easily obtained by using a non self-occluding mounting scheme, where the line-of-sight between the array and the beacon in all situations is maximized.

- In terms of acoustic processing, generating the range measurement distribution is fast and carries little computational cost, since it is just a series of convolutions; on the other hand, generating the angle measurement distribution using the CBF is extremely computationally intensive, since the beamformer output power (Eq. 2.47) has to be computed over a search grid of azimuth/inclination combinations. The resulting angle measurement is accurate and robust to acoustic effects such as reverb and interference, but generating it uses the majority of our processing budget – a performance of approximately 1.25 Hz to beamform 4050 look-angles is at the edge of the acceptable range, since the beacon broadcasts at a rate of 1 Hz; in addition, this produces a fairly low resolution angle distribution, with steps of 1.33° in azimuth and 12.00° in inclination. On top of this, there is no remaining computational budget to incorporate our particle filter into the on-board online processing stack. In order to achieve online closed-loop navigation using piUSBL, computational performance must be improved, and ideally without the loss of robustness and accuracy that beamforming provides and without requiring a more powerful (and more power-hungry) on-board computer.

- As mentioned in the previous point, the resolution achievable by the CBF running on the on-board Raspberry Pi 3 is fairly low; this low resolution directly effects the accuracy of the piUSBL system, since the 'step-size' of resolution directly equates to the positional 'resolution' via triangulation. This resolution is not only tied to the speed at which the on-board computer can beamform at each look-angle, but also to the amount of memory that the computer possesses. A necessary step to speed up beamforming is to precompute and store the phase-shifts associated with each look-angle – if the number of look-angles is too large, there is simply not enough memory to store their associated phase shifts. Each look-angle requires $16 \times NFFT$ bytes of storage, where $NFFT$ is the size of the DFT, each bin of which requires a 16 byte double to store a complex number. Using $NFFT = 8192$, 4050 look-angles requires $4050 \cdot 16 \cdot 8192 = 530.84$ MB,

which is more than half the memory available on the Raspberry Pi 3[29]. In order to improve angle measurement accuracy (and thus piUSBL accuracy), it would be ideal to find some method of improving the resolution of the angle measurement distribution, without having to resort to using a computer with more memory.

- These initial results suggest that particle filtering, and filtering with the addition of factor graph smoothing, both provide good positional performance with factor graph smoothing improving performance slightly. Qualitatively, the trajectory generated by factor graph smoothing is 'better', in that it is smoother and more continuous than that of particle filtering alone; however, the increased system complexity and computational cost introduced with smoothing do not appear to be justified, especially since the smoothing solution is not necessary for closed-loop control. Since the ultimate goal of this work is to perform closed-loop multi-AUV deployments, factor graph smoothing appears to have little additional benefit over particle filtering alone, since the state estimate of the filter is adequate for feedback control. Batch optimization and smoothing can instead be applied in post-processing after mission completion, to generate more accurate and aesthetically pleasing trajectories, if required. As a result of this insight, we decided not to incorporate online factor graph smoothing in our final piUSBL system.

These four insights were instrumental in informing the subsequent development of our piUSBL approach. In the next three chapters, we address these issues by formulating an approach to enable online, closed-loop navigation; by developing a novel beamforming method that requires less memory and drastically improves precision and resolution of the angle measurement distribution; and by comprehensively evaluating and validating the accuracy of our final piUSBL system.

[29] We set the DFT size to 8192 since we collect 8000 samples per array element, and 8192 is the first power of two above 8000.

Chapter 3

Improving Processing Speed: *The Sequential Monte-Carlo Beamformer*

3.1 Introduction

The prototypical Passive Inverted Ultra-Short Baseline (piUSBL) system detailed in the previous chapter has provided us with enough data to demonstrate that the concept is sound, and that the processing pipeline can effectively provide a low-cost localization suite. The lessons learned from experiments with the prototypical piUSBL system have also provided us with an understanding of its limitations, and guidance on which aspects of the system should be improved in order for the system to provide robust and accurate positioning in real-time. In this chapter we focus on the necessary processing improvements needed to obtain online, real-time performance for closed-loop piUSBL positioning.

One of the insights we highlighted from the results of the preceding chapter is that a large majority of our computational budget is devoted to generating the acoustic angle measurement distribution via the use of the conventional beamformer (CBF), whose output is evaluated over a grid of look-angles. The resulting angle measurement distribution, or 'heatmap', provides us with a distribution that indicates the likelihood that the incoming acoustic signal comes from a particular direction, and covers the entire sphere (or 360° in azimuth and 180° in inclination). Generating this 'heatmap' is computationally intensive, and this computation scales with its resolution – this 'heatmap' is then sampled by the piUSBL particle filter in order to update particle weights in accordance to the likeliest direction to the beacon. Thus, we are caught in an ugly trade-off between the resolution of the angle measurement distribution, and the number of particles in our filter – increasing one necessitates a reduction in the other, since

increasing either requires more computation and our computational budget is limited. On the other hand, we would prefer not to reduce either resolution or the number of particles, since a reduction in either will degrade the accuracy of the piUSBL positioning solution.

Thankfully, the piUSBL particle filter provides us with a key insight that enables us to elegantly avoid having to make this trade-off, and by doing so, allows us to perform closed-loop, online navigation using piUSBL. We term the resulting *coupled* beamformer with particle filter the *sequential Monte-Carlo beamformer (SMCB)*. In this chapter we describe this integrated Bayesian filter, and provide experimental results using the resulting closed-loop piUSBL navigation system as fielded on the prototype SandShark autonomous underwater vehicle (AUV), as well as on a conventional Bluefin-21 AUV.

3.2 Beamforming *trade-off with* Particle Filtering

For simplicity, let us consider a set of N_{angles} look-angles that somehow cover the entire sphere in all directions, $S_{angles} = \{(\phi_1, \theta_1), (\phi_2, \theta_2), ..., (\phi_{N_{angles}}, \theta_{N_{angles}})\}$. To generate our 'heatmap' of angles, or *angle measurement distribution*, that indicates the likelihood that a given look-angle (ϕ_i, θ_i) is pointing in the direction of the acoustic beacon, we have to beamform at each look-angle, as shown in algorithm 4 (where we have rewritten and generalized the relevant procedure from algorithm 1 for an arbitrary beamformer).

Algorithm 4 Angle measurement distribution: iterating an arbitrary beamformer over a set of look-angles

procedure GENERATE_ANGLE($S_{angles}, f_{lower}, f_{upper}, f_s, \mathcal{H}(\phi, \theta, \boldsymbol{Y}), y_i \ : \ i = 1, ..., N$)
▷ Generates the angle measurement distribution using an arbitrary beamformer $\mathcal{H}$
▷ Inputs: $S_{angles} = \{(\phi_1, \theta_1), ..., (\phi_{N_{angles}}, \theta_{N_{angles}})\}$: set of look-angles, f_{lower}: lower frequency cutoff, f_{upper}: upper frequency cutoff, f_s: sampling frequency, $\mathcal{H}(\phi, \theta, \boldsymbol{Y})$: arbitrary beamforming function defined for some array, y_i: matched filter outputs for each array element i
▷ Outputs: $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|$: angle measurement distribution 'heatmap'
    $Y_i = CZT(y_i, f_{lower}, f_{upper}, f_s, M)$ ▷ Chirp Z-Transform of matched filter signals with M frequencies in desired frequency range
    $\boldsymbol{Y} = \left[\ Y_1 \ Y_2 \ \ldots \ Y_N \ \right]^T$ ▷ Collection of CZT-transformed signals
    $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}| = zeros(len(S_{angles}))$ ▷ Storage for angle measurement
    for (ϕ_k, θ_k) *in* S_{angles} **do** ▷ Loop through set of look-angles and get arbitrary beamformer output power
        $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|[k] = \mathcal{H}(\phi_k, \theta_k, \boldsymbol{Y})$ ▷ Apply the arbitrary frequency-domain beamformer on the CZT-transformed signals at the given look-angle
    end for
    return $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|$
end procedure

This procedure is used within our acoustic processing pipeline detailed in algorithm 1 in order to generate the range and angle measurement distributions incorporated by the piUSBL particle filter, as listed on line 9 of algorithm 3. Consequently, given N_{angles} look-angles and R particles, the naive particle filter implementation must perform beamforming N_{angles} times, and perform the predict, update and systematic resampling steps R times – if we naively assume that the processing time fore each beamformer loop is n ms, and that of the three particle filter steps is r ms, the total processing time for a single filtering loop incorporating acoustic measurements is approximately:

$$T_{cycle\_naive} = N_{angles} \times n + R \times r \quad \text{ms} \tag{3.1}$$

If we would like to achieve a fixed processing time per cycle, then it is obvious that this requires a trade-off between the number of look-angles, N_{angles}, and the number of particles, R. Reducing the number of look-angles, however, reduces the raw accuracy and precision of the angle measurement distribution; similarly, reducing the number of particles reduces the tracking accuracy of the filter, and causes its solution to become less smooth. Both options are poor.

3.3 Beamforming *coupled with* Particle Filtering

To overcome this computational bottleneck, we make use of a key insight. Notice that during the update step of the particle filter, the weights within the duplicated angle particle set, $\overline{\overline{S}}^{\text{agl}}$, are updated by evaluating the angle measurement distribution at the azimuth and inclination represented by the corresponding particle; this is performed by Eq. 2.76, which we restate here:

$$\overline{\overline{w}}^{\text{agl}}_i = \overline{\overline{w}}^{\text{agl}}_i \cdot |\hat{Z}[\overline{\overline{\theta}}^{\text{agl}}_i(t), \overline{\overline{\phi}}^{\text{agl}}_i(t)]|^2 \tag{3.2}$$

Thus the key insight is as follows: rather than beamforming on a static grid of look-angles represented by the set $S_{angles} = \{(\phi_1, \theta_1), (\phi_2, \theta_2), ..., (\phi_{N_{angles}}, \theta_{N_{angles}})\}$, why not instead beamform directly at the look-angles represented by the particles in the particle filter? Doing so reduces the number of beamforming look-angles from N_{angles} to R (since there are R particles in the filter), and the set of look-angles instead becomes:

$$S_{angles} = \{(\overline{\overline{\phi}}^{\text{agl}}_1, \overline{\overline{\theta}}^{\text{agl}}_1), (\overline{\overline{\phi}}^{\text{agl}}_2, \overline{\overline{\theta}}^{\text{agl}}_2), ..., (\overline{\overline{\phi}}^{\text{agl}}_R, \overline{\overline{\theta}}^{\text{agl}}_R)\} \tag{3.3}$$

This approach represents a *close coupling* between beamforming and particle filtering, and is a particular advantage afforded to us through the use of particle filtering. Because sampling

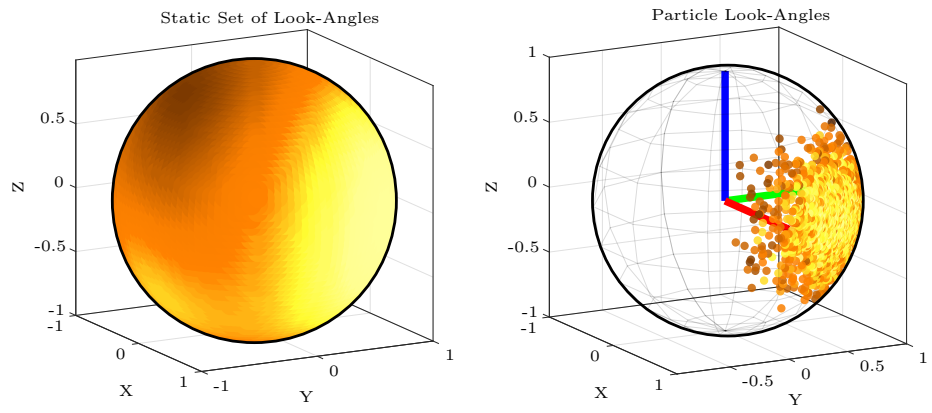

Figure 3.1: Illustration of the beamformed angle measurement distribution from the naive piUSBL pipeline and the sequential Monte-Carlo beamformer (SMCB) pipeline – the beamformed output on a set of look-angles is shown on the unit sphere in the vehicle body-fixed frame (BFF). *Left*: in the original piUSBL pipeline, we beamform on a regular static grid of look-angles that cover the entirety of the unit sphere, and sample from this distribution in the particle filter. *Right*: in the SMCB, we closely couple beamforming and particle filtering by evaluating the beamformer only at the look-angles represented by the particle filter; as a result, the search space is limited by the vehicle motion model to the area of the surface containing the angle distribution maximum. The left uses a total of 7200 look-angles, while the right uses only 1500, representing a significant speed-up in computation.

based methods like the particle filter directly sample at an evaluation point of a distribution, distributions that are themselves generated through iterative evaluation over a set of points (like the output of the beamformer over a set of angles) can be evaluated within the particle filter directly; other Bayesian filtering approaches like the Kalman filter are unable to take advantage of this 'trick', since in the Kalman filter case, a Gaussian fit to the entire measurement distribution must be obtained. Because of this coupling of beamforming and particle filtering (also called sequential Monte-Carlo), we call our approach the *sequential Monte-Carlo beamformer (SMCB)*. This coupling provides a certain complementarity between beamforming and particle filtering – the particles in the filter constrain the search space of beamforming to the area most likely to contain the maximum value of the angle measurement distribution (using the filter's motion model), and the beamformer provides the update to the weight of each particle. This concept is illustrated in Fig. 3.1. Given the same number of particles, this coupling has absolutely no effect on the accuracy performance of the filter, but can drastically reduce the processing time for a single filtering loop to:

$$T_{cycle\_coupled} = R \times n + R \times r = R \times (n + r) \quad \text{ms} \tag{3.4}$$

since we now beamform at R instead of N_{angles} look-angles. The saving in processing time is thus approximately:

$$T_{cycle\_saved} = T_{cycle\_naive} - T_{cycle\_coupled} = (N_{angles} - R) \times n \quad \text{ms} \tag{3.5}$$

which can be significant, especially since it is usually the case that $R \ll N_{angles}$, and the beamforming time n of a single look-angle can be fairly high.

However, there is one outstanding issue that we have yet to address: in the original acoustic processing algorithm (algorithm 1), we precompute and store the phase-shifts associated with each look-angle in the static grid as a necessary step to speed up beamforming – but the set of look-angles represented by the particles is now dynamic, and changes at every time-step. In order to maintain the speed-up provided by this precomputation, in the SMCB we still precompute and store the phase-shifts associated with a static grid of look-angles; however, we simply do not calculate the beamformed output for the entire grid – we restrict the computation to the set of look-angles on this grid that are closest in value to the set of look-angles represented by our particles.

3.4 The sequential Monte-Carlo beamformer

The sequential Monte-Carlo beamformer that we use in the final piUSBL system to provide closed-loop navigation is summarized in algorithm 5 below. Note that it makes many references to the equations and algorithms detailed in chapter 2, sections 2.4 and 2.5, and subsections 2.4.1, 2.4.2 and 2.5.2; note also that we do not restrict the beamforming process to the conventional beamformer (CBF) – this is because we introduce a novel beamforming algorithm that possesses a greater computational efficiency than the CBF in the following chapter.

Algorithm 5 The piUSBL sequential Monte-Carlo beamformer (SMCB)

1: **procedure** SMCB($p(\boldsymbol{x}_{gps}^{llf}), p(\boldsymbol{z}_{depth}^{llf}), \gamma_{ahrs}, \beta_{ahrs}, p(\alpha_{ahrs}), p(v_{sog}), x_i \ : \ i = 1, ..., N$, $S_{angles}, f_{lower}, f_{upper}, f_s, s, \boldsymbol{P}, c, \mathcal{H}(\phi, \theta, \boldsymbol{Y})$)
▷ Continuously estimates the relative position of the beacon in the VCF using our AUV motion model and acoustic range and angle measurements
▷ Inputs: $p(\boldsymbol{x}_{gps}^{llf})$: GPS distribution, $p(\boldsymbol{z}_{depth}^{llf})$: depth measurement distribution, γ_{ahrs}: roll measurement, β_{ahrs}: pitch measurement, $p(\alpha_{ahrs})$: yaw measurement distribution, $p(v_{sog})$: forward speed-over-ground distribution, x_i: measured acoustic signals, S_{angles}: static set of look-angles for precomputation, f_{lower}: lower frequency cutoff, f_{upper}: upper frequency cutoff, f_s: sampling frequency, s: broadcast template, $\boldsymbol{P}$: element positions, c: speed-of-sound, $\mathcal{H}(\phi, \theta, \boldsymbol{Y})$: arbitrary beamforming function
▷ Calculates: $\boldsymbol{\mu}_{beacon}^{vcf}$: relative beacon position estimate, $\boldsymbol{\Sigma}_{beacon}^{vcf}$: state estimate covariance

▷ **Initialization**

for $i = 1 : R$ **do** ▷ Initialize particles

Initialize particle $\boldsymbol{s}_i$ state $\boldsymbol{x}_i$ using Eq. 2.64 and weight $w_i = \frac{1}{R}$

end for

$\boldsymbol{f} = f_{lower} : f_{upper}$ ▷ M equally spaced frequencies (e.g. 7–9 kHz) for CZT

▷ Loop through all static look-angles and pre-compute and store phase shifts to initialize the arbitrary beamformer

$\mathcal{H}(\ldots) \leftarrow$ INITIALIZE STORAGE ▷ Initialize beamformer memory storage

for (ϕ_k, θ_k) *in* S_{angles} **do**

$\mathcal{H}(\phi_k, \theta_k, \ldots) \leftarrow$ SET PHASE SHIFTS WITH $\phi_k, \theta_k, \boldsymbol{P}, c, \boldsymbol{f}$ ▷ Precompute phase shifts for this look-angle

end for

▷ **SMCB Cycle of Predict, Update, Resample, and Estimate**

loop

for $i = 1 : R$ **do** ▷ Predict step

Propagate particle $\boldsymbol{s}_i$ state $\boldsymbol{x}_i$ using Eq. 2.65

end for

$\hat{y}, y_i, \sigma_{max\_sample} = Generate\_Range\_MF(c, f_s, s, x_i \ : \ i = 1, ..., N)$ ▷ Generate the range measurement distribution (detail in algorithm 1)

if $\sigma_{max\_sample} < 5$ **then** ▷ Validity check

Duplicate particle set $\boldsymbol{S}$ into range particle set $\overset{\text{rng}}{\frown}\boldsymbol{S}$ using Eq. 2.66

Duplicate particle set $\boldsymbol{S}$ into angle particle set $\overset{\text{agl}}{\frown}\boldsymbol{S}$ using Eq. 2.67

$S_{particle\_angles} = \{(\overset{\text{agl}}{\frown}\phi_1, \overset{\text{agl}}{\frown}\theta_1), \ldots, (\overset{\text{agl}}{\frown}\phi_R, \overset{\text{agl}}{\frown}\theta_R)\}$

$|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}| = Generate\_Angle(S_{particle\_angles}, f_{lower}, f_{upper}, f_s, \mathcal{H}(\phi, \theta, \boldsymbol{Y}), y_i \ : \ i = 1, ..., N)$ ▷ Evaluate the frequency-domain arbitrary beamformer at the angles represented by the particles (algorithm 4)

for $i = 1 : R$ **do** ▷ Update step

Incorporate acoustic range measurement distribution, $\hat{y}$, for this particle using Eqs. 2.69 & 2.70

Incorporate acoustic angle measurement distribution, $|\hat{\boldsymbol{Z}}^{\boldsymbol{2}}|$, for this particle using Eqs. 2.73, 2.74, 2.75 & 2.76

end for

Reorder $\overset{\text{rng}}{\frown}\boldsymbol{S}$ using ascending weights with Eq. 2.77

Reorder $\overset{\text{agl}}{\frown}\boldsymbol{S}$ using ascending weights with Eq. 2.78

for $i = 1 : R$ **do** ▷ Recombine $\overset{\text{rng}}{\frown}\boldsymbol{S}$ and $\overset{\text{agl}}{\frown}\boldsymbol{S}$ to produce new particles for $\boldsymbol{S}$

Project particle range $\overset{\text{rng}}{\frown}r_i^{\uparrow}$ and angle $\overset{\text{agl}}{\frown}\theta_i^{\uparrow}, \overset{\text{agl}}{\frown}\phi_i^{\uparrow}$ into new $\begin{bmatrix} x_{i,beacon}^{vcf} \\ y_{i,beacon}^{vcf} \end{bmatrix}$ using Eq. 2.79

Update new particle weight w_i using Eq. 2.80

end for

Perform systematic resampling of particles ▷ Resample step

end if

$\boldsymbol{\mu}_{beacon}^{vcf} = \begin{bmatrix} \sum_{i=1}^{R} w_i \cdot x_{i,beacon}^{vcf} \\ \sum_{i=1}^{R} w_i \cdot y_{i,beacon}^{vcf} \end{bmatrix}$ ▷ Estimate step

33: $\boldsymbol{\Sigma}^{vcf}_{beacon} = \frac{1}{R-1}\sum_{i=1}^{R} w_i \left(\begin{bmatrix} x^{vcf}_{i,beacon} \\ y^{vcf}_{i,beacon} \end{bmatrix} - \boldsymbol{\mu}^{vcf}_{beacon} \right) \left(\begin{bmatrix} x^{vcf}_{i,beacon} \\ y^{vcf}_{i,beacon} \end{bmatrix} - \boldsymbol{\mu}^{vcf}_{beacon} \right)^T$

34: **end loop**

35: **end procedure**

The key insight is performed on lines 18 and 19 in algorithm 5, in which the arbitrary beamformer is evaluated at the angles closest to those represented by the particles in the filter.

Since the piUSBL system continuously estimates the position of the beacon *relative* to the vehicle, it enables a novel operating paradigm not yet employed by underwater vehicles – navigation relative to a *moving* beacon whose absolute position is not known to the vehicle. This *relative navigation* paradigm is possible only because the AUV has access to both range and angle estimates to the beacon, allowing it to obtain a relative position fix on every acoustic measurement. This paradigm also has a distinct advantage that enables us to reduce a major drawback associated with all range and angle positioning systems – by constraining vehicle movement to an area local to the beacon, the range-dependent effect of uncertainty in position due to the angle measurement can be bounded; this allows the operator to bound positional error to a desired threshold of uncertainty, and facilitates AUV deployments over large spatial length scales – the beacon itself can be moved over a long distance. The absolute positional error of the AUV is then tied to the positional error of the beacon.

If the acoustic beacon is static and placed at a known global positioning system (GPS) position, we use the relative GPS position of the vehicle to the beacon in order to initialize the filter, as described by Eq. 2.64. However, in the case of *relative navigation* with a moving beacon, we do not want to assume that the relative position of the piUSBL beacon is known to the vehicle upon deployment; instead, we wish for the vehicle to detect and track the beacon, regardless of the fact that the beacon position is unknown to the AUV. Consequently, for relative navigation missions with a moving beacon, we initialize the sequential Monte-Carlo beamformer as follows:

$$\boldsymbol{x}_i(0) = \begin{bmatrix} x^{vcf}_{i,beacon}(0) \\ y^{vcf}_{i,beacon}(0) \\ x^{llf}_{i,beacon}(0) \\ y^{llf}_{i,beacon}(0) \\ z^{llf}_{i,beacon}(0) \\ x^{llf}_{i,auv}(0) \\ y^{llf}_{i,auv}(0) \\ z^{llf}_{i,auv}(0) \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathcal{U}(-r_{max}, r_{max}) \\ \mathcal{U}(-r_{max}, r_{max}) \\ x^{llf}_{i,auv}(0) + x^{vcf}_{i,beacon}(0) \\ y^{llf}_{i,auv}(0) + y^{vcf}_{i,beacon}(0) \\ \mathcal{U}(-r_{max}, 0) \\ x^{llf}_{gps}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ y^{llf}_{gps}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ -z^{llf}_{depth}(0) + \mathcal{N}(0, \sigma^2_{gps}) \\ \vdots \end{bmatrix} \tag{3.6}$$

Essentially, the state elements of each particle in our filter that represent the relative position of the acoustic beacon in the vehicle-carried frame (VCF) are initialized using a Uniform distribution whose extents represent the maximum range of the piUSBL system[1]; the position of the AUV in the local-level frame (LLF) as represented by each particle is initialized around the vehicle's GPS position; and finally, note that since the beacon is no longer static, the position of the beacon in the LLF has a temporal dependency, and is initialized (and propagated in the filter predict step) using the values of the other states.

3.5 Closed-Loop Experimental Results with the Prototype SandShark AUV

To demonstrate the closed-loop navigation capability of the piUSBL system with the incorporated SMCB, we carried out a number of deployments using the prototype SandShark AUV introduced in chapter 2, subsection 2.3.3. These deployments included both absolute navigation where the beacon was fixed in position, as well as relative navigation where the beacon was moving. In these experiments, 1500 particles were used in the SMCB.

3.5.1 Absolute Navigation using a Fixed Beacon

To demonstrate absolute navigation using piUSBL and a single beacon, we carried out two closed-loop deployments of the prototype SandShark AUV (detailed in chapter 2, subsection 2.3.3) on a the Charles River adjacent to the MIT Sailing Pavilion in May of 2017. As was the case for the open-loop experiments performed in the previous chapter, the piUSBL beacon was configured to broadcast a 20 ms, 16–18 kHz linear frequency modulation (LFM) up-chirp (shown to the right of Fig. 3.2), and was affixed to the pavilion dock and submerged to a depth of approximately 1 m at a known GPS positon. The prototype SandShark was programmed to run a mission to follow a racetrack parallel to the dock of 90 m length and 10 m width, at a depth of 2 m and a speed of 1 $\mathrm{m\,s^{-1}}$. The mission length was set to 1200 s, with the vehicle instructed to surface for GPS approximately mid-way through the mission.

Since the main vehicle computer on the prototype SandShark AUV operates with a navigation stack built on the open-source Robot Operating System (ROS), vehicle position as estimated by the SMCB via piUSBL on the backseat (payload computer) is fed back at every time-step to the frontseat (main vehicle computer); this vehicle state estimate is also used by the backseat in order to command desired MOOS-IvP speed, heading and depth to the frontseat, which closes the loop and controls the AUV in order to achieve the desired set-points and to carry out the desired mission. This feedback loop ensures that the backseat

[1] Refer to Fig. 2.13 for a reminder of the reference frames used by our system.

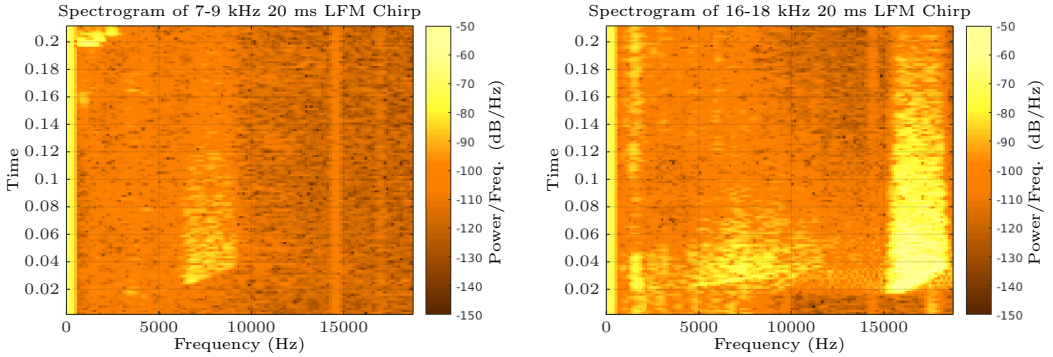

Figure 3.2: Sample in-water spectrograms measured during closed-loop navigation experiments – *Left*: 7–9 kHz, 20 ms LFM up-chirp used for relative navigation experiment. *Right*: 16–18 kHz, 20 ms LFM up-chirp used for absolute navigation experiment.

and frontseat computers have a synchronized estimate of the vehicle state, which is necessary in order to undertake missions in an absolute reference frame using MOOS-IvP behaviors.

Following the same experimental protocol of the final open-loop run in the previous chapter, we also deployed the two commercial Hydroid long baseline (LBL) transponders fastened to the pavilion at a depth of approximately 1 m, with the first transponder at position $(52.8, 23.8)$ m and the second at $(-55.6, -25.6)$ m relative to the piUSBL beacon. As before, the WHOI Micromodem in the prototype SandShark payload was used to query the LBL transponders at a rate of 0.2 Hz. This allows us to compare our solutions to the range values outputted by this independent system, providing a means for quantifying navigation accuracy. Note that the LBL system itself is subject to acoustic effects that result in range outliers. In order to remove these outliers, a simple constant temporal filter is employed. This filter operates by checking the difference between subsequent LBL ranges, and this difference is above that which can be achieved by the vehicle moving at maximum speed in that time period, then that LBL measurement is discarded. This ensures that physically impossible LBL ranges are pruned from each dataset, which removes some obvious outliers.

The plots in Figs. 3.4 and 3.5 illustrate the trajectories for both runs, as estimated by the piUSBL with our integrated sequential Monte-Carlo beamformer (SMCB) on the left, and estimate by naive dead-reckoning on the right. The piUSBL estimates were used by the AUV for closed-loop navigation to follow the desired racetrack. The dead-reckoning solution was generated by not incorporating any corrective acoustic range and angle measurement distributions. Qualitative examination of these plots indicate that the piUSBL approach allows the vehicle to successfully self-localize, as evidenced by the minimal jumps in estimated position whenever the AUV surfaces and GPS reception is restored. In contrast, dead-reckoning experiences jumps in position during run 1 of almost 30 m and 28 m during the mid-mission and end-mission surfacing events respectively, as indicated by the white dashed ellipses in

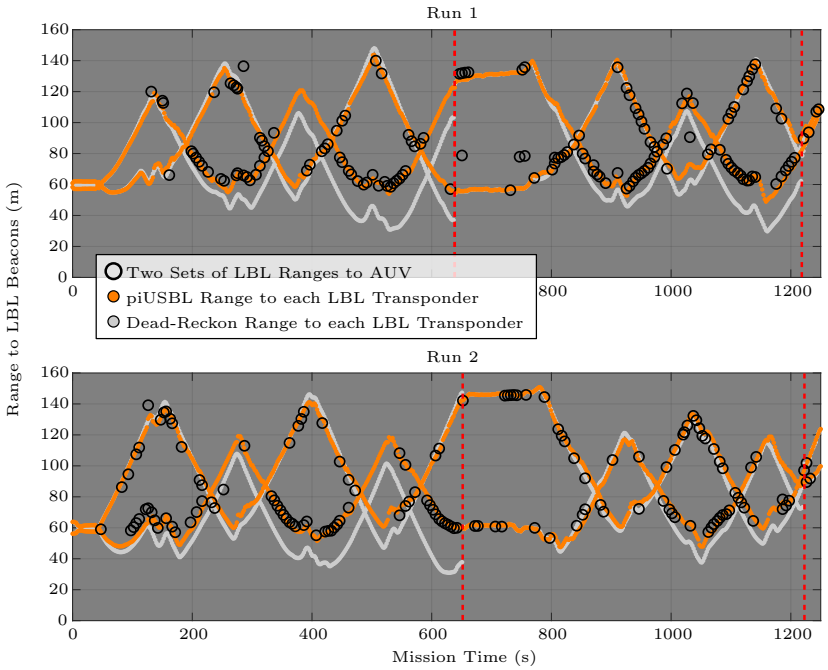

Figure 3.3: Plots of ranges between two commercial LBL transponders and the AUV and the same ranges estimated by piUSBL and dead-reckoning – Micromodem LBL ranges of the two transponders are plotted as black circles; the ranges estimated using the piUSBL trajectories are plotted in orange; the ranges estimated using the dead-reckoning trajectories are plotted in gray; it is apparent that the piUSBL trajectories are consistent with the range measurements of the independent LBL system, which is not the case for dead-reckoning. GPS surfacing events are indicated by red dashed lines. *Top*: Run 1. *Bottom*: Run 2.

Fig. 3.4; similarly, dead-reckoning for run 2 has jumps of about 33 m and 36 m for the two surfacing events indicated in Fig. 3.5. A back-and-forth racetrack mission like this is expected to minimize dead-reckoning error (since the vehicle is continuously transiting between two set points) and demonstrates how quickly this error accumulates (at a rate of almost 3 $\mathrm{m\,min^{-1}}$) in the absence of a Doppler velocity log (DVL)-aided inertial navigation system (INS) and in the presence of water currents. Notice the consistency in the vertical 'jump' in dead-reckoning position whenever the vehicle surfaces – this suggests that the vehicle is commanding a slight upwards velocity vector, which may indicate that it is counteracting downward river currents below the water surface; alternatively, it may indicate an error bias in vehicle heading.

Ranges from these trajectories to the two commercial LBL transponders are plotted in Fig. 3.3. These plots support our previous observations, illustrating close agreement between the ranges output by the independent LBL acoustic system and the trajectory resulting from our piUSBL approach. Again, dead-reckoning quickly diverges from the transponder ranges. Although the LBL system is queried by the vehicle at a rate of 0.2 Hz, only about 32% of

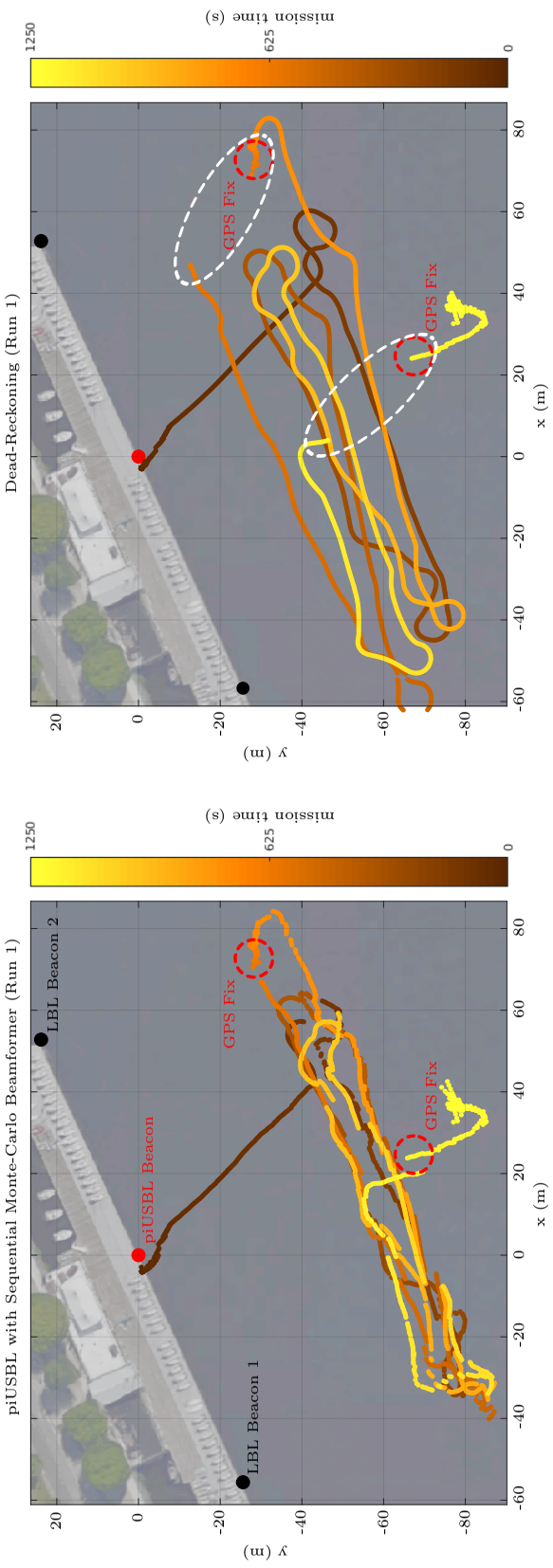

Figure 3.4: Closed-loop, absolute navigation AUV trajectories using a fixed beacon as estimated by the sequential Monte-Carlo beamformer (SMCB) and dead-reckoning for run 1 – the piUSBL beacon is shown as a red dot, and the Hydroid LBL beacons as the black dots affixed to the dock; locations of GPS surfacing events are shown as dashed red circles, and the jump in position for dead-reckoning during these events are highlighted by dashed white ellipses; trajectories are colored with an orange dark-to-light gradient which indicates the mission time in seconds. *Left*: the closed-loop trajectory from the SMCB. *Right*: the trajectory that would have been estimated by dead-reckoning.

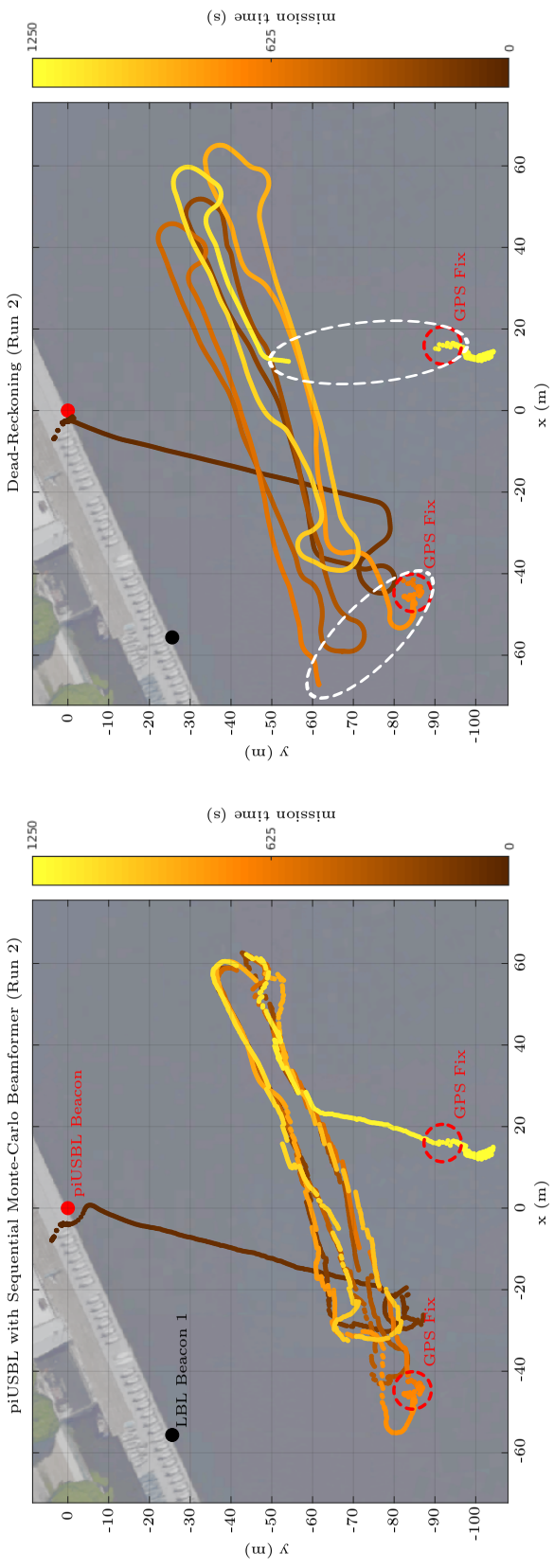

Figure 3.5: Closed-loop, absolute navigation AUV trajectories using a fixed beacon as estimated by the sequential Monte-Carlo beamformer (SMCB) and dead-reckoning for run 2 – the piUSBL beacon is shown as a red dot, and the Hydroid LBL beacons as the black dots affixed to the dock; locations of GPS surfacing events are shown as dashed red circles, and the jump in position for dead-reckoning during these events are highlighted by dashed white ellipses; trajectories are colored with an orange dark-to-light gradient which indicates the mission time in seconds. *Left*: the closed-loop trajectory from the SMCB. *Right*: the trajectory that would have been estimated by dead-reckoning.

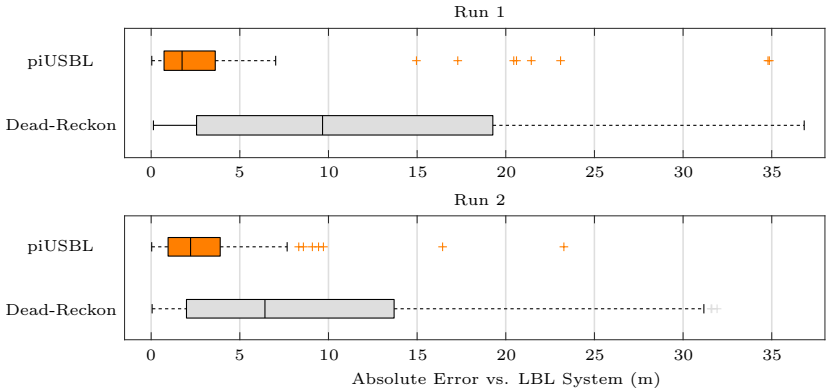

Figure 3.6: Boxplots of absolute difference in range as estimated using the piUSBL and dead-reckoning trajectories against the commercial LBL system – solid centerlines indicate the median value, box edges indicate the 25^{th} and 75^{th} percentiles, and crosses indicate outliers.

those queries were met with a valid response[2], indicating the acoustically challenging nature of the river. As a result, an even smaller percentage ($\sim 10\%$) of LBL ranges occur concurrently for both beacons – without concurrent ranges, LBL-based localization is not possible. This is the reason why we have opted to compare piUSBL to LBL range measurements directly, rather than comparing piUSBL and LBL position estimates (there are simply too few range intersections). This is in contrast to the analysis provided in the the fifth run of the open-loop results in the previous chapter. Outliers in the LBL ranges are apparent (e.g. in run 1 650–760 s when the vehicle is stationary while receiving GPS), but this data still provides us with a means of validating the navigational ability of our system.

Taking the absolute difference between the ranges outputted by the trajectory estimates and the raw ranges from the LBL system allows us to plot error statistics with respect to LBL, as shown in Fig. 3.6. These plots show the absolute difference in range between the LBL measurements and those computed using the piUSBL and dead-reckoning trajectories. For run 1, piUSBL has a median error of 1.74 m, with 75% of measurements falling below 3.62 m; and for run 2, piUSBL has a median error of 2.22 m, with 75% of measurements falling below 3.90 m. The mean absolute error (MAE) for piUSBL is 3.14 m and 2.91 m for runs 1 and 2 respectively, while for dead-reckoning it is 11.14 m for run 1 and 9.42 m for run 2. These results suggest that the piUSBL system significantly improves the navigational ability of the prototype SandShark AUV – they demonstrate that, for absolute navigation using a beacon fixed at a known position, piUSBL navigation as enabled by the sequential Monte-Carlo beamformer provides a low-cost and effective navigation suite for miniature AUVs like the SandShark vehicle.

[2]In this commercial Micromodem-based LBL system, a response is valid only if it is detected with power above a certain threshold.

3.5.2 Relative Navigation using a Moving Beacon

To validate the feasibility of the relative navigation operating paradigm, we performed a proof-of-concept experiment using the prototype SandShark AUV in Ashumet pond in Falmouth, MA in August 2017. In this run, the piUSBL beacon was manually towed by an inflatable kayak, and was set to broadcast a 20 ms, 7–9 kHz LFM up-chirp, as illustrated to the left of Fig. 3.2, at a depth of about 1.5 m. Custom MOOS-IvP behaviors instructed the vehicle to dive to 1.5 m and search for the beacon, home-in on it, and continuously loiter in a 12 m diameter circle around it, as the beacon was periodically and dynamically repositioned. In this case no LBL system was deployed, and so only the internal odometry of the AUV was available to estimate absolute AUV position. To verify that the vehicle was indeed homing in on the beacon, a forward-pointing GoPro[3] camera was mounted to the payload, allowing us to visually confirm the beacon during vehicle flybys.

The dead-reckoning trajectory of the vehicle is plotted in Fig. 3.7, along with the GPS trajectory of the kayak and beacon. The kayak/beacon moved through three different station-keeping latitude/longitude coordinates – the first at about $(41.6346°, -70.5388°)$, the second at approximately $(41.6343°, -70.5387°)$, and the third at about $(41.6341°, -70.5388°)$. Although dead-reckoning is inaccurate, the trajectory of the vehicle clearly indicates that it was successfully able to detect, track, home-in on, and loiter around the beacon, while the beacon was repositioned – this is evidenced by the three visibly distinct circular loitering patterns. The inaccuracy of the absolute dead-reckoning estimate is apparent when looking at the jump in position from a GPS update during an unexpected surfacing event at $(41.6345°, -70.5387°)$, as well as when the vehicle surfaced at the end of the mission.

Successful relative navigation is also supported by imagery captured from the vehicle-mounted GoPro as shown in figure 3.8. These images provide visual confirmation of successful beacon homing and tracking, with image timestamps that correspond well with the times in the dead-reckoning trajectory during which the vehicle came into close proximity of the beacon, as marked by the corresponding dashed red circles in Fig. 3.7. Although not definitive, these preliminary results demonstrated that the concept of relative navigation using piUSBL was sound.

[3] https://gopro.com/

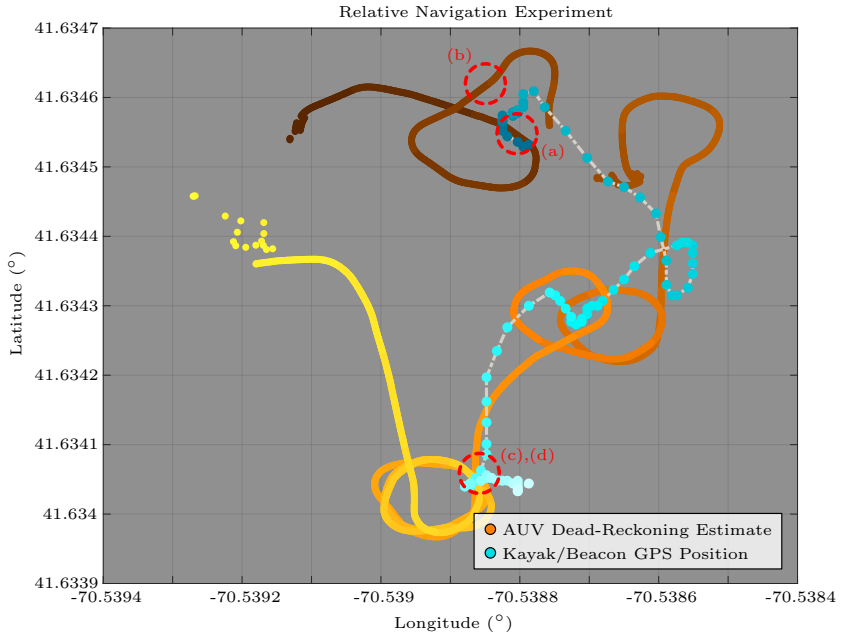

Figure 3.7: Closed-loop, relative navigation AUV trajectory using a moving beacon as estimated by dead-reckoning – the trajectory of the vehicle is colored by an orange dark-to-light gradient that indicates mission time; the position of the kayak and beacon measured by GPS is colored by the blue dark-to-light gradient that indicates mission time; dashed red circles indicate mission times corresponding to beacon flybys during which the beacon was visible in the AUV–mounted GoPro camera, as shown in Fig. 3.8.

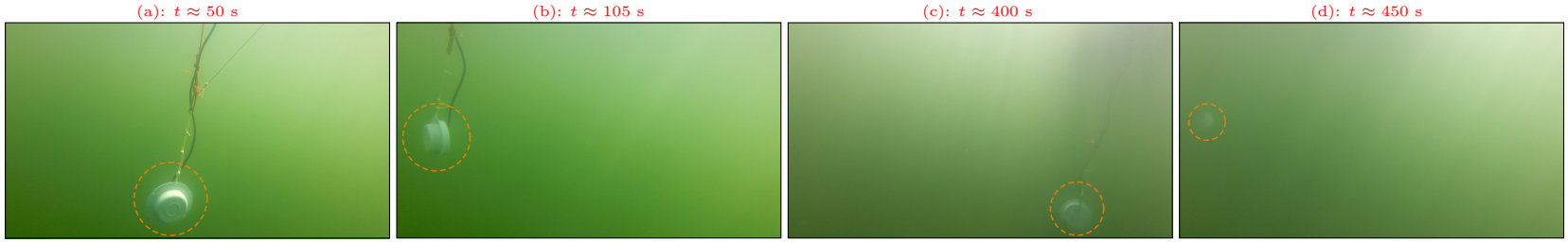

Figure 3.8: AUV–mounted GoPro camera imagery of the piUSBL beacon from the relative navigation run during vehicle flybys.

3.6 Hardware (Bluefin-21 Autonomous Underwater Vehicle Configuration)

As a brief aside, and to demonstrate that the piUSBL concept has utility outside of solely providing a low-cost navigation suite for inexpensive and miniature underwater vehicles, we implemented the piUSBL system stack with the SMCB on a conventional, large-size Bluefin-21 AUV equipped with a DVL-aided INS. Such a vehicle, which is able to generate an accurate estimate of speed using its DVL and an accurate heading estimate using its high-grade inertial measurement unit (IMU), is able to navigate using dead-reckoning for fairly long periods of time without accruing a large increase in positional error – thus, it may seem as though piUSBL can provide little benefit. However, there is one significant advantage that piUSBL can provide: the capability of *relative navigation* against a moving beacon. The use case for the system on the Bluefin-21 AUV is to use this relative navigation capability to enable a return-to-beacon behavior for eventual under-ice operations[4].

3.6.1 Acoustic Beacon

Unlike our previous experiments with the prototype SandShark AUV, for this experiment we make use of a WHOI Micromodem 2 as the acoustic beacon. The beacon consists of a WHOI Micromodem 2 deckbox, which broadcasts acoustic signals into the water using a 10 kHz transducer towfish. The MicroModem 2 is configured to broadcast a 7–9 kHz, 20 ms LFM up-chirp at a rate of 1 Hz, and is triggered by the pulse-per-second (PPS) signal from a GPS receiver on board the ship.

3.6.2 Platform

The platform used for this experiment was a Bluefin-21 AUV[5], called *Macrura*, pictured in Fig. 3.9. *Macrura* is equipped with a navigation suite that includes a Doppler velocity log (DVL) for speed-over-ground estimation, and a high-grade Crossbow attitude and heading reference system (AHRS) with Leica DMC-SX magnetic compass for heading and attitude estimation, which are combined into a standard DVL-aided INS – with this navigation suite, the vehicle is able to travel underwater using dead-reckoning with a navigational drift of 1%–5% of distance traveled [149]. The platform is also equipped with a conductivity-temperature sensor, a pressure sensor to estimate vehicle depth, and a WHOI acoustic MicroModem 2 with a 28 kHz transducer for underwater communication with the ship. GPS and radio receivers are mounted in its mast for positioning and radio communication when the vehicle is on the

[4]Since ice floes drift, acoustic commands must be periodically sent to an under-ice AUV to provide the latest position of the return point; relative navigation allows the vehicle to home-in on the return point dynamically.

[5]https://gdmissionsystems.com/en/products/underwater-vehicles/bluefin-21-autonomous-underwater-vehicle/

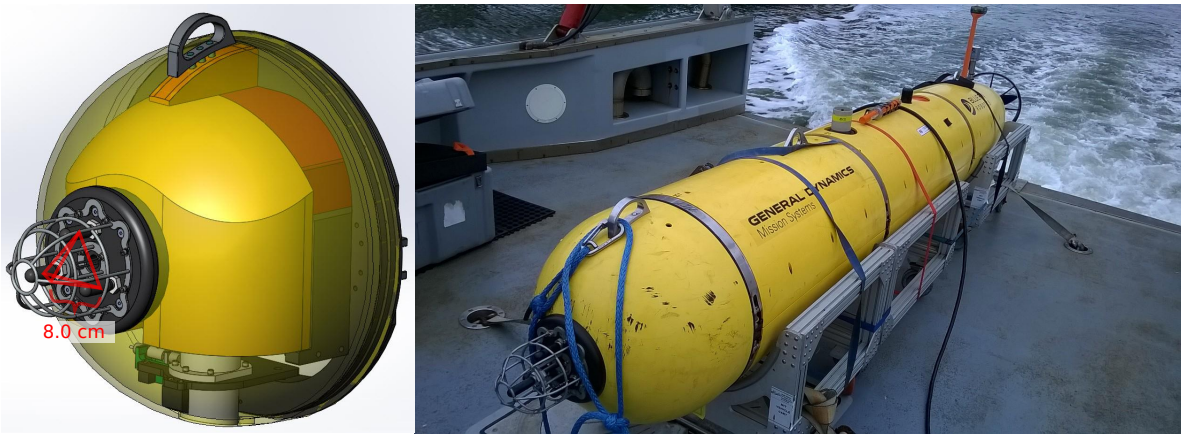

Figure 3.9: Commercial Bluefin-21 AUV (*Macrura*) outfitted with our piUSBL system – this conventional AUV is outfitted with a Doppler velocity log (DVL) and high-grade attitude and heading reference system (AHRS), allowing it to achieve dead-reckoning navigation error rates of 1%–5% of distance traveled. *Left*: computer model of the nose of the vehicle, equipped with the piUSBL tetrahedral hydrophone array. *Right*: photo of *Macrura* on the deck of the deployment ship.

surface. The AUV is propelled using a single propeller, which is actuated to provide vectored thrust. The vehicle is 53.3 cm (21 in) in diameter, and approximately 3 m long.

3.6.3 Payload

The electronics of the *Macrura* piUSBL payload is essentially identical to that of the payload on the prototype SandShark AUV detailed in chapter 2, subsection 2.3.3. As in the prototype SandShark, a four element, nose-mounted tetrahedral array (shown to the left of Fig. 3.9) captures acoustic energy, which is passed through a bandpass circuit and digitized using a DAQ; the DAQ is triggered to record the data in synchrony with the broadcast of the Micromodem beacon on the ship, using the PPS signal from an on-board chip-scale atomic clock (CSAC); data is recorded onto a Raspberry Pi 3 computer. The only difference with the prototype SandShark payload is that the tetrahedral array is larger, due to the larger size of the vehicle – the edges of the regular tetrahedron hydrophone array are 8 cm long.

3.6.4 Receiver USBL Array and Source Signal

Since the array is a regular tetrahedron, the analysis of the tetrahedral array for the prototype SandShark AUV from chapter 2, subsection 2.3.3 holds for this configuration. The only difference is that the frequency of the source signal should be lowered, due to the increase in the size of the array – with an inter-element spacing of 0.4λ–0.5λ, this corresponds to an operating frequency for our 8 cm array of about:

$$0.08 \cdot \frac{1}{0.45\lambda} = \frac{c}{f} \tag{3.7}$$

$$f = \frac{c}{0.08\ \text{m}} \cdot 0.45\lambda \tag{3.8}$$

$$f = \frac{1500\ \text{m/s}}{0.08\ \text{m}} \cdot 0.45\lambda = 8437.5\text{Hz} \tag{3.9}$$

where we use the speed-of-sound of 1500 $\text{m}\,\text{s}^{-1}$, since we perform this experiment in saltwater. As mentioned before, we use a 7–9 kHz up-chirp.

3.7 Closed-Loop Experimental Results with the Bluefin-21 AUV

To test the closed-loop relative navigation capability of our piUSBL system on *Macrura*, we deployed the vehicle for an experiment in Massachusetts Bay in the area of Broad Sound in May 2017. This deployment was part of engineering tests of the vehicle in preparation for its planned deployment to the Arctic for Ice Exercise (ICEX) 2018, and so the relative navigation test was only a small part of the overall deployment. The vehicle was deployed and recovered from a 54 foot catamaran ship, to which the Micromodem beacon towfish was affixed at a depth of approximately 10 m. During these tests the vehicle was instructed to perform a number of MOOS-IvP behaviors, including loiters, yo-yo depth oscillations between 5–10 m, and racetracks; at some point during the deployment the 7–9 kHz signal was set to broadcast, at which point *Macrura* would autonomously detect the signal and switch over into a dynamic homing behavior, where it would dynamically loiter around the drifting beacon at a standoff distance of 150 m. In this homing mode, the vehicle was configured to maintain a depth of 5 m and a speed of 1.7 $\text{m}\,\text{s}^{-1}$, and to continuously circle around the ship at its standoff distance even as the ship drifted due to wind on the surface. The position estimate of the beacon relative to the AUV generated by piUSBL was used by the behavior running on the backseat to request desired heading, speed, and depth commands to the platform, but was not fed back to the frontseat main vehicle computer. As with the closed-loop experiments with the prototype SandShark, 1500 particles were used in the SMCB.

The series of plots in Figs. 3.10 and 3.11 illustrate how the sequential Monte-Carlo beamformer (SMCB) within the piUSBL system was able to detect and refine its estimate of the relative position of the beacon over time, once the beacon was set to broadcast and more acoustic measurements were captured. The particles in the filter were more or less uniformly distributed within the operating area at the beginning of the sequence before the beacon was set to broadcast; once the beacon began broadcasting, we can see how the particles started to 'cluster' at the correct ranges and angles to the beacon relative to the AUV. Note that in

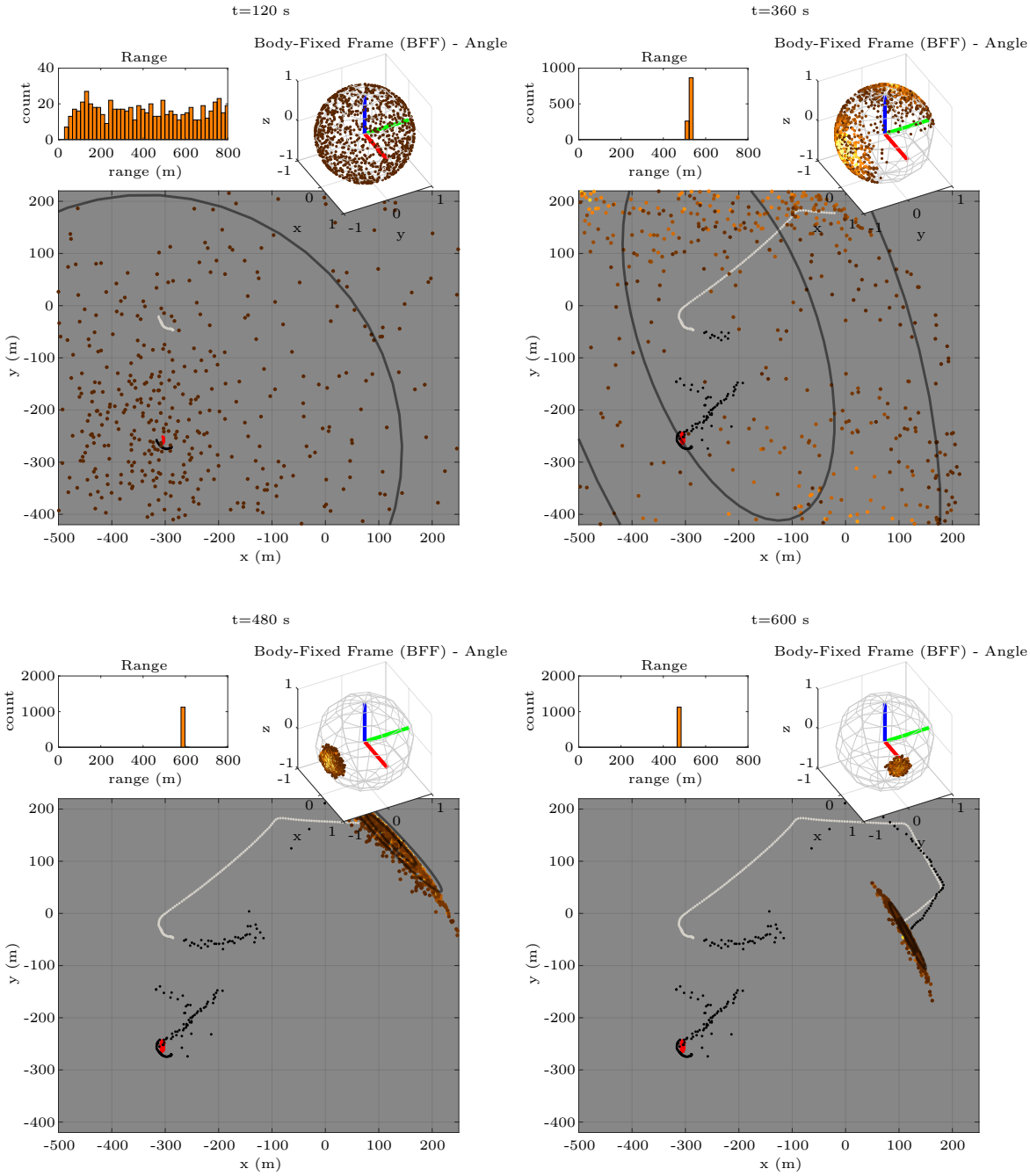

Figure 3.10: 1st 4 of 8 sequence of plots illustrating the sequential Monte-Carlo beamformer (SMCB) converging during the relative navigation experiment with *Macrura* – the times at the top of the plots indicate elapsed time since the ship beacon started broadcasting; filter particles are plotted in orange, with those in the range particle set shown as a histogram in the top left, those in the angle particle set shown in the top right, and those in the LLF shown in the main axis; the estimated trajectory from piUSBL is shown in black, and from the DVL-aided INS in gray; Gaussian fit 1σ and 2σ ellipses for the LLF particles are shown.

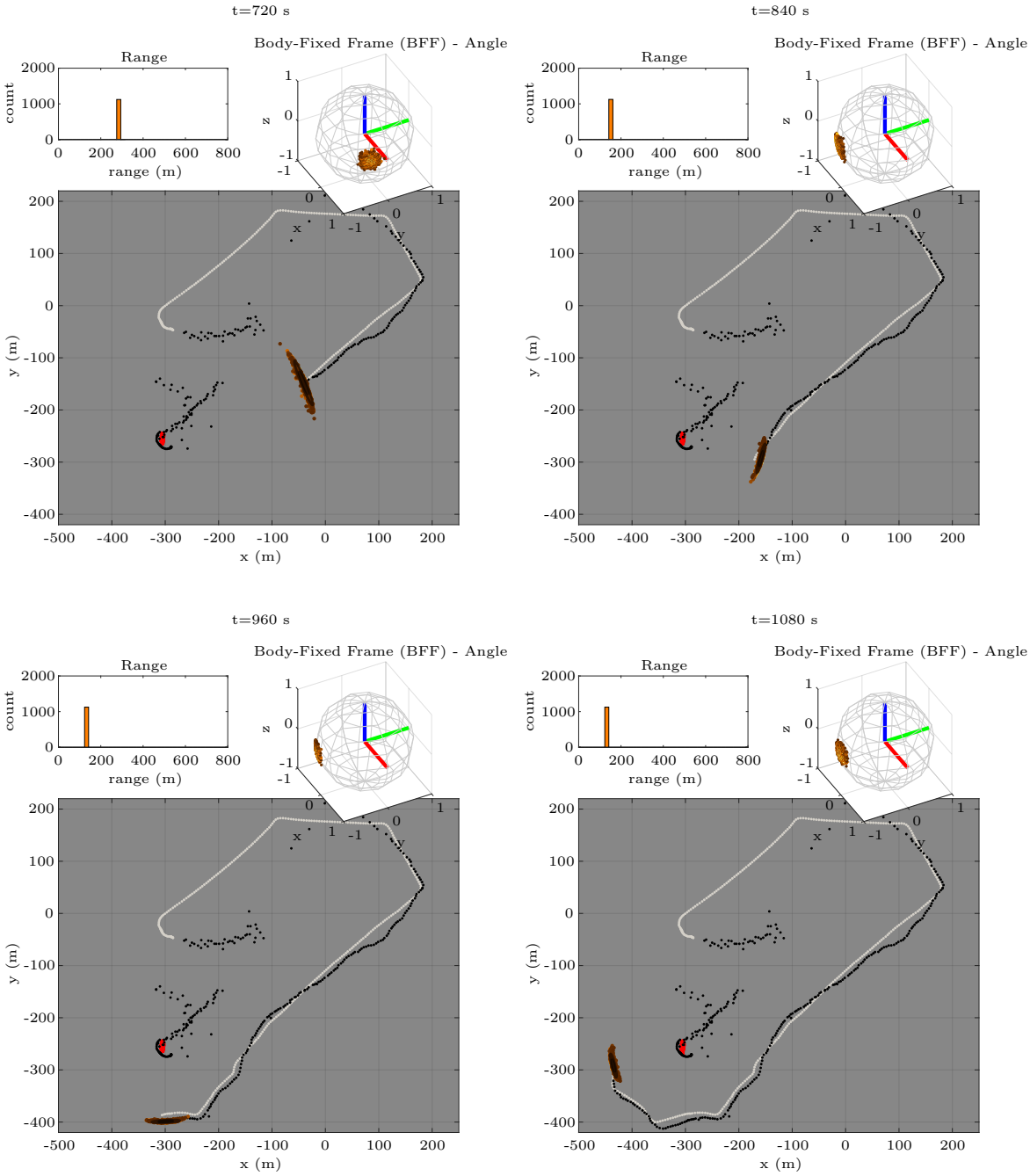

Figure 3.11: 2nd 4 of 8 sequence of plots illustrating the sequential Monte-Carlo beamformer (SMCB) having converged during the relative navigation experiment with *Macrura* – filter particles are plotted in orange, with those in the range particle set shown as a histogram in the top left, those in the angle particle set shown in the top right, and those in the LLF shown in the main axis; the filter has converged and closely follows the trajectory of the DVL-aided INS solution, enabling the vehicle to autonomously track, home-in on and loiter around the ship.

Figure 3.12: Closed-loop, relative navigation trajectory of the Bluefin-21 AUV *Macrura* with a moving ship-based beacon – ship position is in red, drifting in place around $(-300, -250)$ m; the trajectory estimated by the piUSBL sequential Monte-Carlo beamformer (SMCB) and offset by ship position is shown in orange, while the trajectory as estimated by the AUV's DVL-aided INS is shown in gray; particles from the SMCB at times shown in Figs. 3.10 and 3.11 are also shown, along with 1σ and 2σ ellipses of multivariate Gaussian fits.

these figures, we have plotted the states of the filter that represent the position of the vehicle in the local-level frame (LLF), and offset them by the position of the ship in post-processing, in order to compare the track generated by piUSBL (as black dots) with the trajectory from the vehicle's DVL-aided INS solution (in gray). The particles representing the range distribution to the beacon in the range domain are shown in the top left of these figures as a histogram plot, while the particles representing angle distribution to the beacon are shown on the unit sphere in the body-fixed frame (BFF) at the top right of these figures. It is apparent from the particles in Fig. 3.10, that between 120 s and 480 s, the filter had converged and 'locked on' to the relative position of the beacon, as shown by the tight cluster of particles in the range, BFF and LLF domains. Between 480 s and 600 s, we see that the covariances estimated by the multivariate Gaussian fit to the particles had reduced to the point where the vehicle had sufficient confidence about the position of the beacon – as the covariance fell below a certain threshold, the homing behavior to the beacon was 'triggered', which caused the AUV to turn towards the ship as seen at $t = 600$ s in Fig. 3.10. As the sequence continues up to 1080 s in Fig. 3.11, we see that *Macrura* was successfully able to detect and home-in

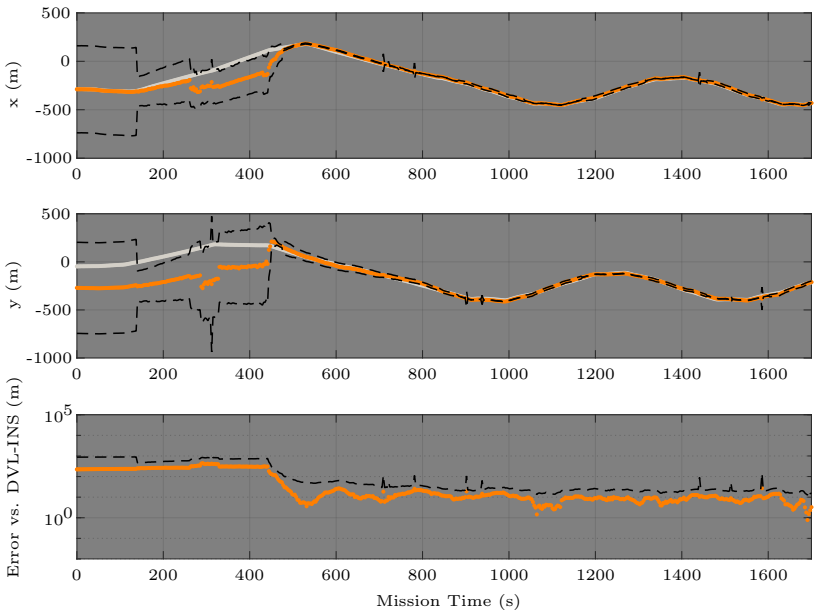

Figure 3.13: Plots of x and y position of *Macrura* as estimated by piUSBL, and 2–norm error against DVL-aided INS – the estimate from piUSBL is shown in orange, with estimated particle standard deviations as dashed black lines, and the reference estimate from the AUV's DVL-aided INS is shown in gray. *Top*: x-position estimate. *Middle*: y-position estimate. *Bottom*: 2–norm error of piUSBL trajectory against reference DVL-aided INS trajectory.

on the drifting beacon, and had completed a half-loiter around the ship (shown in red) – the trajectory generated by the piUSBL system in black closely follows the DVL-aided INS trajectory in gray, signifying a fairly accurate and robust beacon tracking solution by our piUSBL system. The trajectory over the entire sequence of beacon detection, tracking and homing is shown in Fig. 3.12, which illustrates *Macrura* having successfully completed a full relative loiter around the ship at the desired standoff distance of 150 m; the close agreement between the piUSBL trajectory and that of the vehicle's DVL-aided INS is again apparent. These qualitative results provide compelling evidence for the utility and accuracy of the relative navigation paradigm using piUSBL.

The vehicle trajectory of *Macrura* from the moment that the beacon was set to broadcast is plotted in the x and y dimensions in Fig. 3.13;. The top two plots in this figure show the trajectory as estimated by our piUSBL system in orange, along with the estimated standard deviation of a Gaussian fit to the particles in x and y as dashed black lines; the trajectory estimated by the vehicle's DVL-aided INS is used as a reference and is shown in gray. This figure clearly shows how the particles in the SMCB started out very dispersed (as evidenced by the large standard deviations), and began converging very quickly around the 450 s mark, when the standard deviations in x and y collapsed. Note the reason why there was little

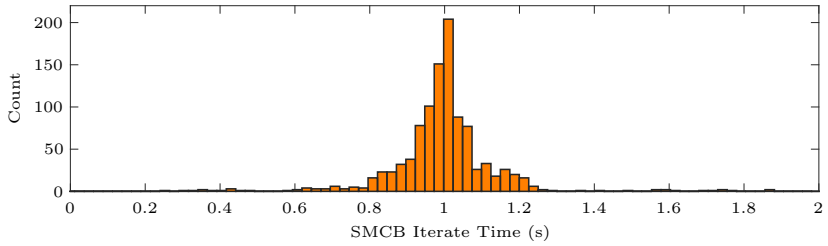

Figure 3.14: Histogram of sequential Monte-Carlo beamformer (SMCB) iterate times with 1500 particles when processing acoustic measurements during the *Macrura* relative navigation experiment.

change in the spread of the particles up to 450 s: looking back at Fig. 3.10, we see that during this first period of the test, the AUV was traveling *away* from the ship, meaning that the nose-mounted USBL array was self-occluded by the AUV body and did not have a clear view of beacon; it was not until around the 450 s mark, once the vehicle had turned to follow a programmed mission path to the South-East, that the array got a clear view of the beacon and the piUSBL system began to obtain valid acoustic measurements, allowing the filter to quickly converge. During the remainder of the test the standard deviations remained very small, and the x and y estimates of piUSBL closely track the reference trajectory.

The bottom plot of Fig. 3.13 shows the 2-norm distance between the trajectory estimated by piUSBL versus the reference trajectory from the DVL-aided INS on a logarithmic scale, with the corresponding standard deviation as estimated from the particles as the dashed black line. We see from this plot that by the 500 s mark, the piUSBL trajectory has come to within 10 m of the reference trajectory, and remains around this level for the remainder of the experiment, with the error fluctuating between 0–30 m. Note that at around 480 s, the AUV is almost 600 m away from the beacon, as seen in Fig. 3.10; as illustrated by the particles in Fig. 3.13, the uncertainty in angle, rather than range, is the largest contribution to this error.

An additional advantage of the sequential Monte-Carlo beamformer (SMCB) is that both the accuracy of its estimate and its iterate (or cycle) time are tied to a single parameter – the number of particles in the filter, R, as is clear from Eq. 3.3. Thus, this single parameter can be easily tuned for the compute power of the platform on which piUSBL is running. Since the acoustic beacon broadcasts at a rate of 1 Hz, we ideally wish to keep the iterate time of the filter below 1 s in order to incorporate as many acoustic measurements as possible. Running the piUSBL system on a Raspberry Pi 3 onboard *Macrura*, with 1500 particles in the SMCB, iterate times are indeed grouped around this desired limit of 1 s, as shown by the histogram of Fig. 3.14. In fact, 1500 particles may be too many for the Raspberry Pi 3, since it is apparent that around 50% of the iterate times take longer than 1 s, which cause the system to discard a large number of acoustic measurements captured by the array.

3.8 Summary of the sequential Monte-Carlo beamformer

In this chapter we have introduced the sequential Monte-Carlo beamformer (SMCB), a close-coupling of particle filtering and beamforming, and a critical modification to our piUSBL system that enables online, closed-loop navigation. The key insight of the SMCB is that we can directly use the particles in the filter as the look-angles to beamform at in the beamformer, allowing us to avoid the increased processing time of a two-stage beamforming plus particle filtering pipeline. Using this approach the particles in the filter inform beamforming and concentrate its computation in the most likely direction of the acoustic beacon, and in turn, the beamformer directly updates the weights of the particles without them having to sample from a fully evaluated underlying angle distribution. The drastic decrease in processing time enables real-time, online, closed-loop piUSBL navigation without sacrificing accuracy. In addition, this close-coupling allows us to increase or decrease filter accuracy and computation time simultaneously by increasing or decreasing the number of particles in the SMCB.

In this chapter we have also introduced the concept of *relative navigation*, a novel operating paradigm for AUVs, in which the vehicle navigates *relative* to a moving beacon. This relative navigation paradigm is enabled by piUSBL, since the vehicle continuously updates an estimate of the position of the beacon relative to itself using acoustic range and angle measurement distributions. With this approach, the AUV can maintain a bounded uncertainty on the estimate of its relative position by operating within a specified radius of the beacon – movement of the beacon itself can then enable deployments over large spatial length scales without a corresponding increase in positional uncertainty.

Using the improvement in processing speed afforded to us by the SMCB, we performed closed-loop experiments with the prototype SandShark AUV, with results from absolute navigation experiments provided in subsection 3.5.1, and with results from a preliminary relative navigation experiment in subsection 3.5.2. These results demonstrate the significant improvement in navigational accuracy provided to such low-cost and miniature AUVs by our piUSBL system, vastly outperforming dead-reckoning.

The piUSBL system was also installed on a conventional Bluefin-21 AUV called *Macrura*, as detailed in section 3.6. Although this vehicle's DVL-aided INS provides sufficient navigational accuracy, piUSBL gives it the capability to operate under the relative navigation paradigm, enabling it to autonomously track and home-in on the acoustic beacon. A relative navigation experiment undertaken with *Macrura* (as detailed in section 3.7) demonstrated the AUV successfully tracking and homing in on the acoustic beacon, with results showing a close agreement between the trajectory estimated by the piUSBL system and that estimated by the AUV's DVL-aided INS.

In the next chapter we tackle another point of improvement identified from initial experiments with the prototype SandShark in chapter 2 – improving the resolution of the angle

measurement distribution output by beamforming. Note that in this chapter we emphasized the fact that the beamformer in the SMCB was arbitrary – this is because the following chapter will introduce a novel beamforming method that allows us to generate angle measurement distributions at a much higher resolution and at a much faster speed than the conventional beamformer (CBF), given the same amount of computer memory.

Chapter 4

Improving Measurement Precision: *Element Pair Decomposition Beamforming*

4.1 Introduction

The close coupling of particle filtering and beamforming in the sequential Monte-Carlo beamformer (SMCB) framework detailed in the previous chapter allows our piUSBL system to achieve real-time processing rates for closed-loop navigation. The SMCB formulation, however, still does not address issues with the angular precision of the conventional beamformer (CBF) – its precision is limited by the fact that we precompute the phase shifts for all look-angles, meaning that memory usage scales linearly with the number of look-angles; this limits their number to the available memory of the platform. In addition, as the resolution of the beamformed output increases with the number of look-angles, the computation time increases in proportion – this is especially problematic for 3D beamforming, since the two dimensions of azimuth and inclination means that the number of look-angles increases quadratically with resolution.

In this chapter we introduce element pair decomposition (EPD) beamforming, a novel beamformer whose memory usage and computation time are tied to the *number of pairs of elements* in the array. For certain use-cases (such as ours), this scaling factor is especially useful, since it allows us to increase the beamforming resolution dramatically without an associated dramatic increase in memory use or computation time. On the other hand, EPD beamforming carries with it some accompanying drawbacks, particularly in terms of angular resolution[1] due to its unique beampattern estimate; these properties make it unsuitable for

[1] Angular resolution refers to the array's ability to distinguish between two sources close in space, and is related to the half-power beamwidth of the main lobe of its beampattern.

many other applications. Here we detail EPD beamforming, explain the key insight that led to its development, and compare its performance against conventional beamforming. We describe the drawbacks and advantages of both methods, providing the reader with some insight into how EPD beamforming may be advantageous for their application.

4.2 Issues with Conventional Beamforming

For simplicity, let us consider generating a static grid of look-angles that covers the entire sphere by dividing the 360° azimuth domain using N_{azim} divisions, and dividing the 180° inclination domain using N_{incl} divisions – as a result, we have a set of $N_{incl} \times N_{azim}$ look-angles that we can represent as a matrix:

$$S_{angles} = \left\{ \begin{bmatrix} (\phi_1, \theta_1) & (\phi_2, \theta_1) & \dots & (\phi_{N_{azim}}, \theta_1) \\ (\phi_1, \theta_2) & (\phi_2, \theta_2) & \dots & (\phi_{N_{azim}}, \theta_2) \\ \vdots & \vdots & \ddots & \vdots \\ (\phi_1, \theta_{N_{incl}}) & (\phi_2, \theta_{N_{incl}}) & \dots & (\phi_{N_{azim}}, \theta_{N_{incl}}) \end{bmatrix} \right\} \tag{4.1}$$

where $0 \leq \phi_i < 360°$ and $0 \leq \theta_i < 180°$. Now, since the domain of inclinations is half the length of the domain of azimuths, let us make the further simplifying assumption (for the purposes of our analysis later in this chapter) that $N_{azim} = 2 \times N_{incl}$, so that the grid 'cells' are square. If we label $N_{incl} = C$, then the number of look-angles is:

$$N_{angles} = 2C^2 \tag{4.2}$$

Recall from the previous chapter, in section 3.2, that in order to generate our 'heatmap' of angles (represented by Eq. 4.1), or the full *angle measurement distribution*, we have to iterate over the entire set of look-angles and, when using conventional beamforming, we have to evaluate the CBF spatial filter at every one of these look-angles. Regardless of whether or not we precompute the phase shifts for this set of look-angles, the complexity is quadratic in the division parameter C ($\mathcal{O}(C^2)$). If we precompute and store the phase shifts for every look-angle in our grid (as we did for our prototypical piUSBL system in chapter 2 and for our closed-loop SMCB improvement in chapter 3) then the cycle time for each iteration is drastically reduced, but now we introduce a memory requirement that is also on the order of C^2. This quadratic complexity in both memory and computation time is the source of all our issues with the conventional beamformer (CBF) – we optimized the cycle time and reduced the cost of evaluating the CBF spatial filter for each look-angle by precomputing and storing the phase shifts for our grid of look-angles; but in doing so, we have incurred a cost in terms of memory. Note that this trade-off is specific to our application, since both the computational

and memory budgets of the Raspberry Pi 3 are extremely limited – given, for example, a graphical processing unit (GPU), or a high-end computer with lots of random access memory (RAM), then the CBF can be GPU-parallelized for extremely fast heatmap computation, or the computation can be kept CPU-serialized and a very high-resolution heatmap can also be quickly generated by storing the very large number of associated phase-shifts. In our system we have neither the luxury of computational parallelization, or a lot of memory.

Our previous innovation of the sequential Monte-Carlo beamformer (SMCB) allowed us to reduce the computation time by explicitly reducing the number of look-angles, N_{angles}, from a static grid of $2C^2$ angles, to R particle angles – however, we still precomputed the phase shifts for our grid of $2C^2$ look-angles, limiting the maximum precision of our angle measurement distribution. In this chapter we are primarily interested in improving this resolution by reducing memory usage – as such, for the remainder of the chapter we assume that we compute the entire heatmap, or full angle measurement distribution, over $2C^2$ look-angles.

4.3 Constructing the Element Pair Decomposition Beamformer

EPD beamforming was initially inspired by the generalized cross-correlation phase-transform (GCC-PHAT) method used for time difference of arrival (TDOA) source localization [157] [126] [128] [129]. These GCC-PHAT-based methods often use a two-stage approach: (i) first the TDOAs between all pairs of elements in the array are estimated, usually by performing a cross-correlation between the signals on each element in the pair and selecting the arg-maximum; (ii) second, these TDOAs are input into a nonlinear system of equations and optimized over position (x, y) using least-squares in order to obtain an estimate of the source [126] [128]. This approach was extended by incorporating the GCC-PHAT into the steered response power PHAT (SRP-PHAT) method, in which near-field beamforming is incorporated so as to obtain a more robust estimate of source position [127] [128] [129]. In the SRP-PHAT, instead of extracting the arg-maximum from the cross-correlation and performing a nonlinear least-squares optimization, a grid search is performed over *near-field positions*. Consider a near-field beamforming approach: each near-field position on a grid contributes a unique time-delay between each element in the array and its origin, which can be calculated by subtracting the time-of-flight from that position and the array origin from the time-of-flight to each element[2]; by 'undoing' these time-delays for each element and summing over all elements, we are essentially performing conventional beamforming, only in the *near-field* rather than in the *far-field*. SRP-PHAT differs from such a near-field CBF in the following manner: instead of referencing and undoing the time-delays against the array origin and summing over all elements, the SRP-PHAT considers all *pairs of elements*, referencing their signals against

[2]This can be considered as referencing the array elements against a 'virtual' element at the origin.

each other. For each pair, it cross-correlates their signals together; the resulting signal is then phase-shifted (time-delayed) by the TDOA that is expected by a given position in the near-field search grid; finally, all phase-shifted, cross-correlated signals from all pairs are summed together to generate the likelihood that the signal originated from that position on the grid.

Although EPD beamforming is fundamentally different to the SRP-PHAT method, in that it works in the far-field (just as conventional beamforming), it doesn't cross-correlate element signals against one another, and it references signals against *pair origins* (a term we explain later), it is greatly inspired by the reduction of the array into pairs of elements; it turns out, as we will see later, that such a *pair decomposition* is greatly beneficial in speeding up computation and reducing memory usage in certain 3D beamforming situations. In some sense it can also be considered as the natural endpoint of *subarray processing* [130], in which small arrays are considered as the elements of a larger 'superarray', a topic which we briefly touched upon in chapter 1; however, it also differs from these subarray approaches, in that we consider all unique *pair combinations* as the 'subarrays' - thus, in our approach a single element within the array is part of multiple 'subarrays', unlike standard approaches where the elements within each subarray are not shared between subarrays.

4.3.1 Key Insight

When performing conventional beamforming with a 1-dimensional linear array, the convention is to place the N elements of the array along the z-axis; recall from the explanation of beamforming in chapter 2, subsection 2.4.2, that the array manifold vector contains all relevant array characteristics and is defined by:

$$\boldsymbol{v}(\omega) = \begin{pmatrix} e^{-j\omega\tau_1} \\ e^{-j\omega\tau_2} \\ \vdots \\ e^{-j\omega\tau_N} \end{pmatrix} \quad \text{WHERE} \quad \omega\tau_i = \frac{\omega \boldsymbol{a}^T \boldsymbol{p}_i}{c} \tag{4.3}$$

where $\boldsymbol{p}_i$ is the position of element i, and a is given by:

$$\boldsymbol{a} = \begin{pmatrix} -\sin(\theta) \cdot \cos(\phi) \\ -\sin(\theta) \cdot \sin(\phi) \\ -\cos(\theta) \end{pmatrix} \tag{4.4}$$

However, in the case of a 1D linear array, we clearly see that since the positions of the N elements are zero in x and y, then the time delay in Eq. 4.3 reduces to:

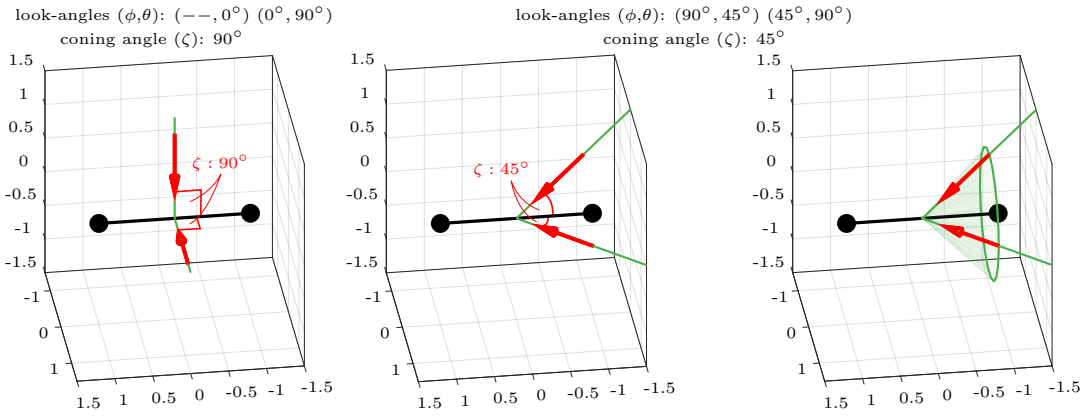

Figure 4.1: Illustration of the key insight of element pair decomposition (EPD) beamforming – multiple 3D look-angles for a pair of elements (or any 1D line array) can always be collapsed from their 2D azimuth-inclination, (ϕ, θ), into a single 1D *coning angle*, ζ; this means that the set of 2D azimuths-inclinations that lie on a 'cone' whose axis is parallel to the axis of the line array can be described by a single *coning angle*, as illustrated with the cone in the right figure, which collapses all look-angles on the cone to a single 45° *coning angle*. The two plots on the left provide examples of two unique 3D look-angles that are described by a single equivalent *coning angle*.

$$\tau_i = \frac{-\cos(\theta) z_i}{c} \tag{4.5}$$

Thus, for linear arrays, any 3-dimensional look-angle described by an azimuth and inclination, ϕ and θ, is completely described *solely* using its inclination angle. This insight is the key to EPD beamforming, and is illustrated in Fig. 4.1. Consider an array composed of only two elements as shown in this figure – in the leftmost plot we imagine an incoming plane wave that is incident onto this pair of elements from two different directions; in both these cases the signal arrives at both elements *simultaneously* – thus, it is impossible for the pair to disambiguate between which of the two angles the signal arrived from. Similarly, in the center plot there are two arrival angles for a plane wave that result in the same time-delay of the signal arriving at the two elements – again, the pair of elements cannot determine which of the two the signal came from. In general, for any pair of elements in an array, there exists sets of angles for which a signal arriving at the pair will produce the same time-delay – these sets of angles lie along the surface of 'cones' whose axes are parallel to the line connecting the pair, as illustrated in the rightmost plot of Fig. 4.1. We refer to the angle between the axis of the cone and its lateral surface as the *coning angle* – all the 3D azimuth-inclination look-angles that point along the lateral surface of this cone can be described by this *single coning angle*, and this result allows us to collapse the search space of 3D beamforming drastically.

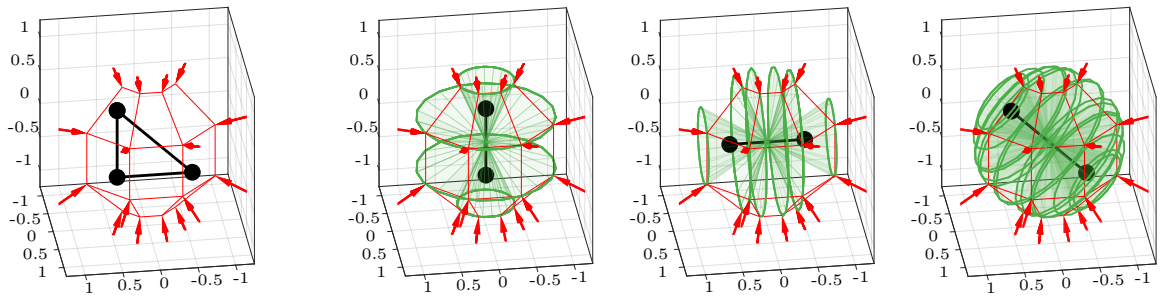

Figure 4.2: Illustration of how 3D look-angles can be represented by coning angles in the space of the array's decomposition into element pairs – the 3-element right-angle array shown on the left is decomposed into its three unique element pairs, shown on the right; the 3D look-angles can be collapsed to 4 coning angles for the vertical element pair, into 6 coning angles for the horizontal element pair, and remains as 16 coning angles in the diagonal element pair.

Having described the key insight that led to the development of EPD beamforming, we now show how this insight is applied to an arbitrary 3D array. Let us consider a simple 3-element right-angle planar array illustrated on the leftmost plot of Fig. 4.2; In this plot we show the array in black, and a set of 16 3D look-angles that we wish to beamform at in red[3]. Now, instead of beamforming at each of these look-angles, let us instead *decompose the array* into all unique pairs of elements, as shown in the next three plots of Fig. 4.2. Now we see something interesting: for the first *element pair* (which stands vertically), we see that the set of 16 look-angles can instead be represented by 4 coning angles – a reduction of 4 in the number of angles to beamform at! For the next element pair (which lies horizontally), we see that the set of look-angles reduces to 6 coning angles, a slight reduction; and for the final element pair (diagonal), we see that the set of look-angles cannot be collapsed, with 16 coning angles remaining. If we beamform at these coning angles for each pair of elements, we must beamform at a total of $4 + 6 + 16 = 26$ angles, rather than the original 16 look-angles; thus, in this example it is less efficient in terms of both computation and memory to do so as compared to the 3D CBF, but it provides us with an intuitive understanding of the EPD beamforming process.

4.3.2 3D Beamforming using Element Pair Coning Angles

Now let us consider approaching this insight from the opposite direction – instead of determining the coning angles for each element pair using the original 3D look-angles, let us instead *construct* the 3D look-angles from the three sets of coning angles, as illustrated in Fig. 4.3. In this figure, we see each of the array's element pairs on the left side; for the vertical element pair, we divide the 180° coning angle space into 3 coning angles; for the horizontal element

[3]Actually, the look-angles are shown pointing in their opposite directions – this is shown for clarity.

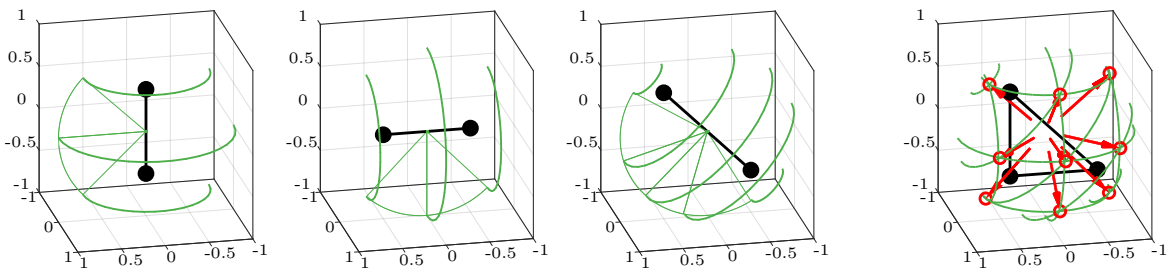

Figure 4.3: Illustration of how 3D look-angles can be constructed by coning angles in the space of the array's decomposition into element pairs – each element pair of the array contributes a set of coning angles to the EPD beamforming output, as shown in the three plots on the left; these sets of coning angles intersect all together along 9 3D look-angles, as shown in red on the right. The number of coning angles scales with the number of element pairs in the array times the number of coning angles, while the number of 3D look-angles scales at a much faster quadratic rate.

pair, we do the same; and for the diagonal element pair, we divide the space into 5 coning angles. In the rightmost plot of this figure, we plot all three sets of coning angles together, along with the entire array – it is apparent that all three sets of cones from each element pair intersect all together at exactly 9 look-angles, shown in red in the rightmost plot. Thus, in this example, we have used $3+3+5 = 11$ coning angles to represent a set of only 9 look-angles – this example is again less efficient! However, note that if we continue to increase the number of look-angles in the constructed 'grid', the number of coning angles scales at a much slower rate than the resulting number of look-angle intersections. For C coning angles in the vertical and horizontal element pairs of this array, the number of intersections scales as C^2, while the number of coning angles across all three element pairs scales as:

$$C_{total} = C + C + (2C - 1) = 4C - 1 \tag{4.6}$$

where the first term is the number of coning angles for the vertical element pair, the second term is the number for the horizontal element pair, and the third is the number for the diagonal element pair. Thus, for this particular case, with this array and beamforming on a square grid of 3D look-angles, the total number of coning angles grows linearly, while the total number of constructed look-angles grows quadratically! Clearly, for a large number of 3D look-angles, EPD beamforming has a significant advantage in terms of computation and memory usage.

The illustration in Fig. 4.3 surfaces a couple of caveats with the EPD beamforming approach. Firstly, notice that for each element pair, its origin is exactly half-way between the

two elements – we term these origins *pair origins*. Since *pair origins* are not aligned with the *array origin* (shown in the right plot of Fig. 4.3), the output of the EPD beamformer is only an *estimate* of the CBF output – this is due to the fact that we do not compensate for the time-delay between the array origin and pair origins, as we will explain later. Pair origins are necessary to exploit the efficiency of using coning angles, and restricts the use of EPD beamforming to the far-field case where time-delays between elements are invariant to an origin shift. Secondly, we notice that the construction of 3D look-angles is dependent on the geometry of the array – depending on how coning angles are constructed, in general, intersections will only occur between two element pairs. For example, if the diagonal element pair in Fig. 4.3 had only a single coning angle at 90°, only 3 of the 3D look-angles will have an intersection using all three element pairs – the remaining 6 look-angles will be constructed using the intersections from just the vertical and horizontal element pairs. The accuracy of the look-angle construction is thus a complicated function of the number of element pairs in the array, the geometry of the array, and the number of coning angles in each element pair – its analysis is beyond the scope of this thesis.

The concept of element pair decomposition (EPD) beamforming is thus quite simple – instead of performing 3D beamforming over a set of look-angles that covers the sphere, we instead perform this *element pair decomposition* of the array, and for each element pair, we beamform over a set of *coning angles*. In general, it is infeasible to select these coning angles for a given array geometry so that the coning angles from all possible element pairs intersect over a grid, as we illustrated in Fig. 4.3 – instead, we use a single set of C coning angles for all element pairs:

$$S_{coning} = \{\zeta_1, \zeta_2, \ldots, \zeta_C\} \tag{4.7}$$

where $0 \leq \zeta_i < 180°$. Generally, the 180° domain of coning angles is divided uniformly to generate this set. After beamforming over these coning angles for each element pair, we sum their outputs to produce the full EPD beamforming output over an entire sphere of 3D look-angles. To do so, we take the set of 3D look-angles over the sphere (for example, the original look-angle set given in Eq. 4.1), and for each look-angle in this set, we find the *nearest* coning angle from each element pair and sum their outputs – thus we use a simple *nearest neighbor* summation technique.

We illustrate the concept of EPD beamforming in Fig. 4.4 (for clarity, this conceptual illustration is restricted to 2D). The top of this figure shows the regular CBF – conventional beamforming is performed over a set of 18 look-angles (shown in green), and for the look-angle which points in the closest direction of the incoming plane wave, the measured signals on the elements are phase-shifted into constructive alignment, producing the largest response for the beamformer. The bottom of this figure shows EPD beamforming – in this case, the array

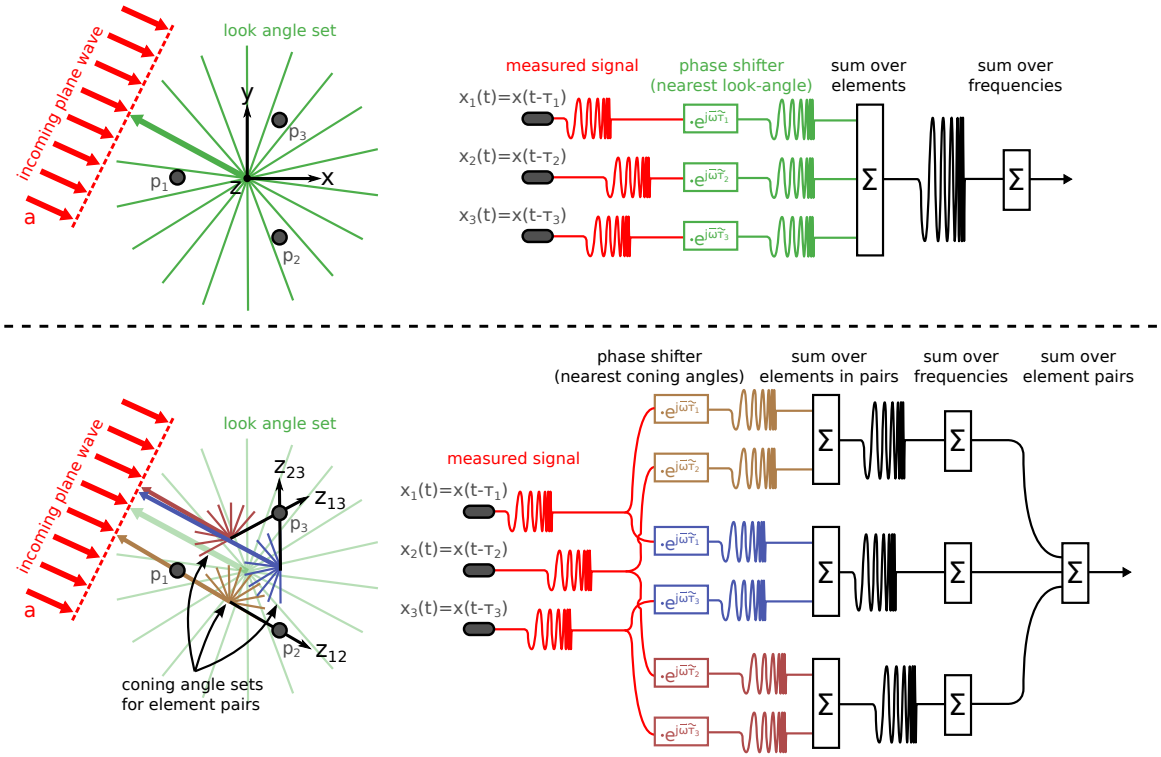

Figure 4.4: Conceptual illustration of element pair decomposition beamforming – the signals measured by each receiver are time-delayed or phase-shifted according to the geometry of the array, the angle of the incoming plane-wave, and the frequency of operation; *Top*: in conventional beamforming, the output power is calculated over a set of look-angles (shown in green), phase-shifting the received signals according to each look-angle, then summing the phase-shifted signals of all elements and summing all frequency components; the output power is maximum for the look-angle pointing in the direction of the incoming plane-wave. *Bottom*: in EPD beamforming, the array is decomposed into element pairs, with a set of coning angles used for each element pair; each element pair is placed within its own coordinate system (z_{12}, z_{13}, z_{23}), whose origin is halfway between the pairs of elements, so that the pair can be treated as a linear array; to determine the EPD beamforming output for a desired look-angle (in semi-transparent green), the nearest coning angle is found for each element pair, the received signals on the pair are phase-shifted according to the coning angle, and these phase-shifted signals summed over the pair and the frequency components; finally, these values are summed over all element pairs to determine the estimated output for that look-angle.

is first decomposed into three element pairs, each of which resides within its own coordinate system (labeled as z_{12}, z_{13} and z_{23}); this allows us to treat each element pair as a linear array, whose origin is halfway between the pair (the *pair origin*). For each element pair, conventional beamforming is performed over a set of *coning angles* (shown in red, blue and brown centered at the pair origins), and the EPD beamforming output for each look-angle is calculated by summing the beamformed outputs of each element pair at the coning angles that are nearest

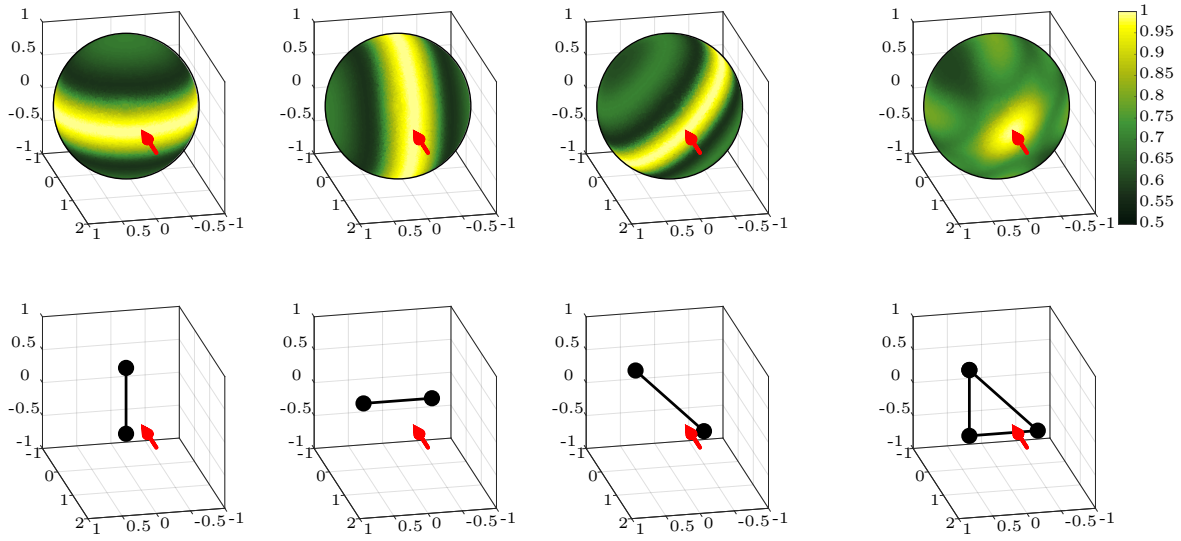

Figure 4.5: Illustration of how the element pair decomposition (EPD) beamforming output is constructed from element pair beamformed outputs – after beamforming over the set of coning angles for each element pair in the array as shown in the three left figures, their beamformed outputs are summed to obtain the full EPD beamforming output as shown in the rightmost figure; the ambiguity associated with each element pair is removed via constructive and destructive summation, allowing the array to detect the direction of the incoming plane wave, indicated by the red arrow. Note that the output is normalized by its maximum.

to the look-angle – this figure illustrates this procedure for the look-angle that is pointing in the direction of the incoming plane-wave.

This resulting output from EPD beamforming is conceptually illustrated for our 3-element array in Fig. 4.5. In this figure, the coning angles in the set increment by 0.5°, and so we beamform each element pair at a half-degree resolution; the incoming plane wave is incident from the direction indicated by the red arrow. For illustrative purposes, we visualize the beamformed output for each element pair over the entire sphere by projecting the output for each coning angle onto the sphere – by doing so, we can clearly see the rotational ambiguity associated with these coning angles as 'banding' that is rotationally symmetric around the axis of each pair. The EPD beamforming output is shown as the rightmost figure, obtained by summing the outputs of the array's pairs – this has enabled us to disambiguate the direction of the incoming plane wave through the constructive and destructive summation of the outputs of individual element pairs.

Consider again the computational and memory cost of conventional beamforming, which we talked about at the start of this chapter – the cost is proportional to the number of look-angles, and we showed that for a static grid of look-angles covering the sphere, the number of look-angles is $2C^2$ where C was the number of inclinations. For EPD beamforming, if we

use the same number C as the number of coning angles, then we demonstrated in this section that the computational and memory cost grows as the product of the number of element pairs in the array and the number of coning angles:

$$N_{angles} = PC \tag{4.8}$$

where P is the number of element pairs. The computational gain and reduction in memory is significant for small arrays where the total number of pair combinations is small. In fact, the memory requirement is even smaller than PC: note that if two element pairs within an array have elements that are separated by the same distance, and we use the same set of coning angles for both pairs, then the phase shifts associated with these two element pairs are identical since they are identical linear arrays within their own coordinate systems – as a result, we only have to store one set of phase shifts for both element pairs. Thus, we see that the memory requirement has an *upper bound* of PC. In our case, with a regular tetrahedral array, the memory savings are enormous – the computational cost is PC, while the memory cost is only C, since all pairs of elements within the array have the same separation distance!

4.3.3 Issues with EPD Beamforming

So far we have demonstrated that EPD beamforming can be significantly more efficient for arrays that have a small number of element pair combinations. This indicates to us the first issue with EPD beamforming – as the number of elements in the array grows, the number of unique pair combinations grows as:

$$P = \binom{N}{2} = \frac{N!}{2!(N-2)!} = \frac{N(N-1)}{2} \tag{4.9}$$

where N is the number of elements in the array. This factorial growth means that EPD beamforming is generally only useful when the number of elements in the array is small, or at least when compared to the desired resolution of the output, i.e. $PC \ll C^2$.

Array Origin Phase Offsets

The illustration of Fig. 4.4 demonstrates another issue with EPD beamforming – there is an offset between each pair origin and the array origin which induces an additional time delay that is not compensated for when beamforming individual element pairs. As a result, there is an additional 'offset' in the summed signals of each element pair, seen after the first summation in this diagram. Unfortunately, this is a fundamental limitation of EPD beamforming, and it forces us to sum over all frequency components before summing over all element pairs – as such, the resulting output is only an estimate of the full CBF output.

Another consequence of failing to compensate for these origin offsets is that the resulting EPD beamforming output has a much higher 'floor' in its beampattern. Since the floor of the beampattern from conventional beamforming is always greater than zero, summing over the CBF output from all element pairs raises the floor through summation. This result is apparent in Fig. 4.5, where the floor of the beampatterns of each individual element pair is much lower than the summed EPD beamforming output. This means that the half-power beamwidth of the main lobe from EPD beamforming is wider than that of the CBF.

Reconstruction from Element Pair Coning Angles

Another issue with EPD beamforming is the reconstruction of the full array output from the coning angles of individual element pairs. We use simple nearest neighbor summation: after finding the coning angle from each element pair that is closest to the desired look-angle, we simply sum their outputs. As a result, the reconstruction accuracy is dependent on a number of intertwined factors: (i) the number of coning angles C in the set S_{coning} (the finer the resolution, the better the reconstruction); (ii) the geometry of the array (how the coning angles from each pair intersect affects the quality of the reconstruction); and (iii) the set of look-angles themselves (they can be carefully selected to be as close as possible to the intersections of coning angles, as was done in Fig. 4.3). Analysis of the theoretical accuracy of this reconstruction is beyond the scope of this work, but later on in this chapter we qualitatively illustrate how this reconstruction is affected by the number of coning angles C, and provide some numerical results on the maximum angular error between the nearest coning angle in each element pair and the corresponding look-angle.

Integration with the Sequential Monte-Carlo Beamformer

To integrate EPD beamforming and our sequential Monte-Carlo beamformer (SMCB) from the previous chapter, we simply replace the CBF with EPD beamforming to generate the beamformer output power for each particle in the filter. This highlights a fourth issue with EPD beamforming – it does not provide any computational speedup over conventional beamforming within the SMCB, since both beamformers would be evaluated at the same number of particle look-angles. However, the memory savings provided by EPD beamforming still stand, and these savings enable additional functionality for our piUSBL system. By storing fewer phase-shifts for a given resolution of look-angle grid, we can store the phase-shifts at a finer frequency resolution, or over a wider bandwidth. As we will see in chapter 6, this wider bandwidth is critical for allowing us to command different vehicle behaviors by broadcasting and detecting signals in different frequency ranges. In addition, as mentioned before, these memory savings enable beamforming at a much higher look-angle grid resolution, improving angle measurement precision substantially.

4.4 Element Pair Decomposition Beamforming

Now that we have described the basic principle behind EPD beamforming, let us formulate the process more formally. Given some arbitrary array defined by its N element positions $\boldsymbol{p}_i$, we first decompose the array into its $\frac{N(N-1)}{2}$ element pairs. We do this by first finding all unique combinations of pairs of elements:

$$S_{unique} = \{(\boldsymbol{p}_1, \boldsymbol{p}_2), \ldots, (\boldsymbol{p}_i, \boldsymbol{p}_j)\} \quad \text{SUCH THAT} \quad i \neq j \tag{4.10}$$

Note that if the set contains $(\boldsymbol{p}_i, \boldsymbol{p}_j)$ then it cannot contain $(\boldsymbol{p}_j, \boldsymbol{p}_i)$, since the pairs must be unique combinations. For each element pair, we then calculate the distance between the elements:

$$D_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j|| \tag{4.11}$$

and we construct the element pairs as a set of linear arrays in their own coordinate systems, such that each pair origin is halfway between their two elements:

$$S_{pairs} = \left\{ \begin{bmatrix} \frac{-D_{12}}{2} & \frac{D_{12}}{2} \end{bmatrix}^T, \ldots, \begin{bmatrix} \frac{-D_{ij}}{2} & \frac{D_{ij}}{2} \end{bmatrix}^T \right\} = \{\boldsymbol{P}_{12}, \ldots, \boldsymbol{P}_{ij}\} \tag{4.12}$$

WHERE

$$\boldsymbol{P}_{ij} = [P_i \;\; P_j]^T \qquad \text{WITH} \qquad P_i = \frac{-D_{ij}}{2}, \qquad P_j = \frac{D_{ij}}{2} \tag{4.13}$$

We refer to the unique coordinate system in which an element pair linear array resides as the *coning angle space*. In actuality, if two element pairs have the same distance we only keep one copy of the linear array to save memory – however, this is a detail we omit for simplicity. In the coning angle space, the time delay experienced by each element in the pair as referenced against the pair origin is given by:

$$\tau_i = \frac{-\cos(\zeta) P_i}{c} \quad \text{AND} \quad \tau_j = \frac{-\cos(\zeta) P_j}{c} \tag{4.14}$$

which can be written more simply as:

$$\boldsymbol{\tau}_{ij} = \frac{-\cos(\zeta)}{c} \boldsymbol{P}_{ij} \tag{4.15}$$

where ζ is the coning angle and c is the speed-of-sound. We can then generate the wideband CBF spatial filter (with M frequency components) for the element pair as usual as:

$$\boldsymbol{H}_{ij}[\boldsymbol{\omega};\zeta] = e^{j\boldsymbol{\omega}\boldsymbol{\tau}_{ij}} = \begin{bmatrix}\boldsymbol{H}_i[\boldsymbol{\omega};\zeta]\\ \boldsymbol{H}_j[\boldsymbol{\omega};\zeta]\end{bmatrix} = \begin{bmatrix}e^{j\boldsymbol{\omega}\tau_i}\\ e^{j\boldsymbol{\omega}\tau_j}\end{bmatrix} = \begin{bmatrix}e^{j\omega_1\tau_i} & e^{j\omega_2\tau_i} & \dots & e^{j\omega_M\tau_i}\\ e^{j\omega_1\tau_j} & e^{j\omega_2\tau_j} & \dots & e^{j\omega_M\tau_j}\end{bmatrix} \tag{4.16}$$

The beamformed output of the element pair for a given coning angle is thus given by:

$$\boldsymbol{Z}_{ij}[\boldsymbol{\omega};\zeta] = \frac{1}{2}\left(\boldsymbol{H}_i[\boldsymbol{\omega};\zeta] \odot \boldsymbol{X}_i[\boldsymbol{\omega}] + \boldsymbol{H}_j[\boldsymbol{\omega};\zeta] \odot \boldsymbol{X}_j[\boldsymbol{\omega}]\right) \tag{4.17}$$

where $\odot$ is the element-wise multiplication operator. This beamformed output is essentially the sum of the phase-shifted signals that were received by the two elements in the element pair; we then sum to obtain the frequency-averaged output:

$$|\hat{Z}_{ij}[\zeta]| = \frac{1}{M}\sum_{k=1}^{M}|\boldsymbol{Z}_{ij}[\omega_k;\zeta]| \tag{4.18}$$

Now, this provides us with the output at the coning angle for a given element pair; to sum across all element pairs, we must first be able to convert from the space of azimuth/inclination look-angles (ϕ,θ) to the coning angle (ζ) space of each element pair. This conversion is just a simple calculation of the angle between two vectors – that of the vector joining the two elements, and the vector represented by the look-angle:

$$\text{LET}\ \ \boldsymbol{a}_{ij} = \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{||\boldsymbol{p}_i - \boldsymbol{p}_j||} \quad \text{AND} \quad \boldsymbol{b}(\phi,\theta) = \begin{bmatrix}\sin(\theta)\cos(\phi)\\ \sin(\theta)\sin(\phi)\\ \cos(\theta)\end{bmatrix} \tag{4.19}$$

$$\text{THEN}$$

$$\zeta = \arccos(\boldsymbol{a}_{ij} \bullet \boldsymbol{b}(\phi,\theta)) \tag{4.20}$$

Thus, Eq. 4.18 can be rewritten in terms of the look-angle:

$$|\hat{Z}_{ij}[\phi,\theta]| = \frac{1}{M}\sum_{k=1}^{M}|\boldsymbol{Z}_{ij}[\omega_k;\arccos(\boldsymbol{a}_{ij} \bullet \boldsymbol{b}(\phi,\theta))]| \tag{4.21}$$

Finally, to get the full EPD beamforming output for a given look-angle, we must sum over all element pairs:

$$|\hat{Z}[\phi,\theta]| = \frac{1}{M}\sum_{i=1,j=i+1}^{N}\sum_{k=1}^{M}|\boldsymbol{Z}_{ij}[\omega_k;\arccos(\boldsymbol{a}_{ij}\bullet\boldsymbol{b}(\phi,\theta))]| \quad \text{WHERE} \quad i\neq j \tag{4.22}$$

where, again, the pairs selected using element i and j must be unique combinations; this concludes the derivation of EPD beamforming.

In principle, given a static grid of look-angles, we can pre-calculate the required coning angles for each element pair to intersect the look-angles precisely using Eq. 4.20. However, in practice, for more flexibility we simply precompute the phase shifts for a static set of coning angles for each element pair, and use Eq. 4.20 to find the nearest precomputed coning angle in the set; this also allows us to reap the benefit of the greater memory savings provided by identical pair separation distances. Thus, the output for a given look-angle is the summation of the nearest neighbor coning angles for each element pair. If the grid of look-angles is static, a look-up table can also be constructed to bypass the use of Eq. 4.20 during runtime, providing an additional speedup. The EPD beamforming process is summarized in algorithm 6.

Algorithm 6 element pair decomposition (EPD) Beamforming

procedure EPD_BEAMFORMING($S_{angles}, S_{coning}, f_{lower}, f_{upper}, c, f_s, x_i \ : \ i = 1, ..., N$)
▷ Calculates the EPD beamforming output for a set of look-angles
▷ Inputs: $S_{angles} = \{(\phi_1,\theta_1), ..., (\phi_{N_{angles}},\theta_{N_{angles}})\}$: set of look-angles, $S_{coning} = \{\zeta_1, ..., \zeta_{N_{coning}}\}$: set of coning angles, f_{lower}: lower frequency cutoff, f_{upper}: upper frequency cutoff, c: speed-of-sound, f_s: sampling frequency, x_i: measured signal on element i
▷ Outputs: $\hat{\boldsymbol{Z}}$: EPD angle measurement over look-angles
▷ **Precomputation of coning angle phase shifts**
    $\boldsymbol{f} = f_{lower} : f_{upper}$ ▷ M equally spaced frequencies for CZT
    $\boldsymbol{H} = zeros(N_{coning}, M, 2, \frac{N(N-1)}{2})$ ▷ Element pair spatial filters: phase-shift storage
    $idx = 1$ ▷ Element pair combination index
    for $i = 1 : N$ **do** ▷ Loop through all element pair combinations
        for $j = (i+1) : N$ **do**
            for ζ_k *in* S_{coning} **do**
                $P_i = \frac{-||\boldsymbol{p}_i - \boldsymbol{p}_j||}{2}$
                $P_j = \frac{||\boldsymbol{p}_i - \boldsymbol{p}_j||}{2}$ ▷ Eq. 4.11 – Eq. 4.13
                $\tau_i = \frac{-\cos(\zeta)P_i}{c}$
                $\tau_j = \frac{-\cos(\zeta)P_j}{c}$ ▷ Eq. 4.14
                $\boldsymbol{H}[k,:,1,idx] = e^{-2j\pi\tau_i\boldsymbol{f}}$
                $\boldsymbol{H}[k,:,2,idx] = e^{-2j\pi\tau_j\boldsymbol{f}}$ ▷ Eq. 4.16
                $idx = idx + 1$ ▷ Increment element pair combination index
            end for

end for
end for
▷ **EPD beamforming computation over set of look-angles**
$X_i = CZT(x_i, f_{lower}, f_{upper}, f_s, M)$ ▷ Chirp Z-Transform of measured signals with M frequencies in desired frequency range
$\hat{\boldsymbol{Z}} = zeros(len(S_{angles}))$ ▷ Storage for angle measurement
for (ϕ_k, θ_k) *in* S_{angles} **do** ▷ Loop through set of look-angles and get EPD beamforming output
$idx = 1$ ▷ Element pair combination index
for $i = 1 : N$ **do** ▷ Loop through all element pair combinations
for $j = (i+1) : N$ **do**
$\boldsymbol{a}_{ij} = \frac{\boldsymbol{p}_i - \boldsymbol{p}_j}{||\boldsymbol{p}_i - \boldsymbol{p}_j||}$
$\boldsymbol{b}(\phi, \theta) = \begin{bmatrix} \sin(\theta_k)\cos(\phi_k) \\ \sin(\theta_k)\sin(\phi_k) \\ \cos(\theta_k) \end{bmatrix}$ ▷ Eq. 4.19
$\zeta = \arccos(\boldsymbol{a}_{ij} \bullet \boldsymbol{b}(\phi, \theta)))$ ▷ Eq. 4.20
$l = find\_nearest(\zeta, S_{coning})$ ▷ Find the index of the coning angle nearest to ζ in the set S_{coning}
$\boldsymbol{Z}_{ij} = \frac{1}{M} \sum \frac{1}{2}(\boldsymbol{H}[l, :, 1 : idx] \odot X_i + \boldsymbol{H}[l, :, 2 : idx] \odot X_j)$ ▷ Eq. 4.17 & Eq. 4.18
$idx = idx + 1$ ▷ Increment element pair combination index
end for
end for
$\hat{\boldsymbol{Z}}[k] = \hat{\boldsymbol{Z}}[k] + \boldsymbol{Z}_{ij}$ ▷ Add the output of this element pair to the total EPD output for this look-angle
end for
return $\hat{\boldsymbol{Z}}$
end procedure

4.4.1 Anytime Stopping and Adaptive Pair Selection

Because the output of EPD beamforming is generated through the successive summation of the beamformed output of element pairs, this naturally enables a direct trade-off between beamforming accuracy and computation time – as the outputs from more element pairs are added to the solution, the more robust the solution becomes; however, we are free to stop adding more element pair outputs at any time during the computation. Thus, EPD beamforming allows for *anytime stopping*, at the cost of a degraded total output. To visualize how the total output is generated, we plot in Fig. 4.6 the output of our EPD beamforming process for a regular tetrahedral array, as element pair outputs are added one-at-a-time; the incoming acoustic signal is incident onto the array from an azimuth and inclination of $\phi = 90°$ and $\theta = 90°$, with an operating frequency set such that its wavelength is twice the length of the separation distance of the array elements; the acoustic signal is simulated with a signal-to-noise ratio (SNR) of 0 dB; with 4 elements in the tetrahedral array, there are a total of 6

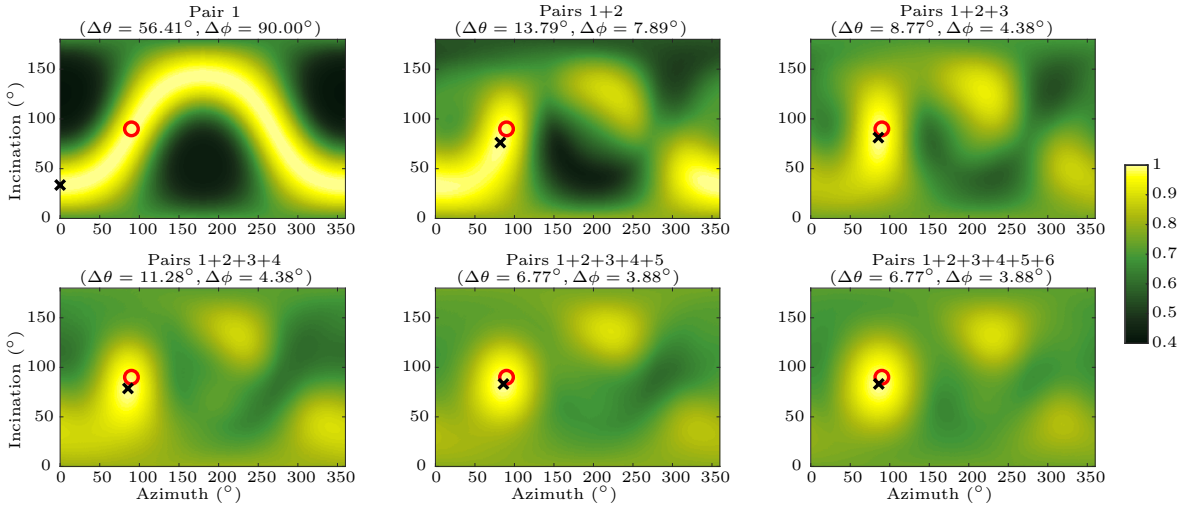

Figure 4.6: Illustration of anytime stopping with element pair decomposition (EPD) beamforming – since the total output is constructed by the sequential summation of the beamformed outputs of element pairs, the calculation can be 'stopped' at anytime, resulting in a degraded output. The 0 dB incoming signal is incident from the look-angle of $\theta, \phi = 90°$, shown as the red circle; the maximum output of EPD beamforming is shown as the black cross, with the difference in inclination and azimuth from the true direction shown as $\Delta\theta$ and $\Delta\phi$ in the title of each plot. As more element pairs outputs are added, the accuracy improves, but the 'floor' increases.

element pairs. For this set of outputs, the number of coning angles was set to 360, resulting in an element pair angular resolution of 0.5°, and the look-angle grid had the same step-size of 0.5° in azimuth and inclination, giving a total number of look-angles of $360 \times 720 = 259200$.

It is apparent from Fig. 4.6, that as the outputs from more element pairs are added together, the ambiguity associated with the coning angles quickly disappears – in fact, once the first three pairs are added, the array has detected the direction to the source within 10°. As more pairs are summed, this accuracy continues to improve – with five element pairs, the error in inclination is 6.77° and the error in azimuth is 3.88°. Note that including the outputs from all element pairs improves the robustness of the solution – if a single element is recording bad data, the effect of its contribution to the total output will be lessened. Thus, EPD beamforming enables the operator to directly trade-off solution accuracy and computation time, allowing the algorithm to be dynamically tuned to the available computational resources of the platform. This figure again highlights one of the drawbacks of EPD beamforming: the 'floor' of the total output is raised as more element pairs are added – so although accuracy is not compromised, the array's ability to distinguish between two sources close in space (or its angular resolution) is impacted since the beamwidth of the main lobe is reduced.

This successive summation of the outputs of element pairs also enables EPD beamforming to *adaptively* select which pairs of elements are to be included in the full output based on

some criteria. This could prove useful in certain applications or array designs. As an example, consider an array that conforms to the body of a vehicle – for an incident acoustic wave from a certain direction, it is obvious that only a subset of elements within the conformal array would be able to detect the signal; these elements should be included within the EPD beamforming process, while elements within the acoustic shadow should not, since the addition of those signals would only degrade the accuracy of the beamforming output. The selection of elements could be done algorithmically, for example, by determining whether a signal is present within the measurement, or by checking if the addition of successive pair outputs converges upon some stable value.

4.4.2 Coning Angle Resolution

In order to generate the EPD beamforming output for a desired look-angle, the nearest coning angle for each element pair is found and the beamformed output for each pair at those coning angles are summed; as such, the accuracy of the total output is dependent on how finely the coning angle set discretizes the coning angle space – the higher the resolution of the set (the more coning angles), the higher the accuracy of the output. To visualize how the resolution of the coning angle set affects the accuracy of the output reconstruction over the set of look-angles, we vary the number of equal-sized coning angles in the set and plot the resulting look-angle reconstruction in Fig. 4.7. We again use a regular tetrahedral array, with the acoustic signal incident onto it from $\theta = \phi = 90°$, with its operating frequency set to twice the length of the element separation distance of the array; the SNR of the signal is set to infinity (i.e. no noise). The number of coning angles was varied from 5 to 180, discretizing the coning angle space from 36° to 1°. The grid of look-angles that we reconstruct has a step size of 0.5° in azimuth and inclination, for a total number of look-angles of $360 \times 720 = 259200$.

Looking at Fig. 4.7, we see that the quality of the reconstructed output over the grid of look-angles is highly dependent on the number of coning angles used for each element pair; when the number of coning angles is very small, for example when we use only 5 or 11 coning angles, the reconstruction is very discretized with 'patches' of equal value, as in the first two plots in the top left of this figure. As a result, the accuracy of the output is greatly diminished. The advantage of using fewer coning angles is a further reduction in memory use, since fewer phase shifts must be stored. It is interesting to note that the observed 'pattern' of patches is not only dependent on the coning angles in the set, but its major features are entirely the result of the geometry of the array, which dictate how the coning angles of different element pairs intersect. Looking at the sequence of images in this figure, we see that as the number of coning angles increases, the accuracy improves as the reconstruction becomes 'finer' – with just 45 coning angles per pair, the reconstruction has enabled the array to determine the angle to the source to within about 2°. With 180 coning angles (shown at the bottom right

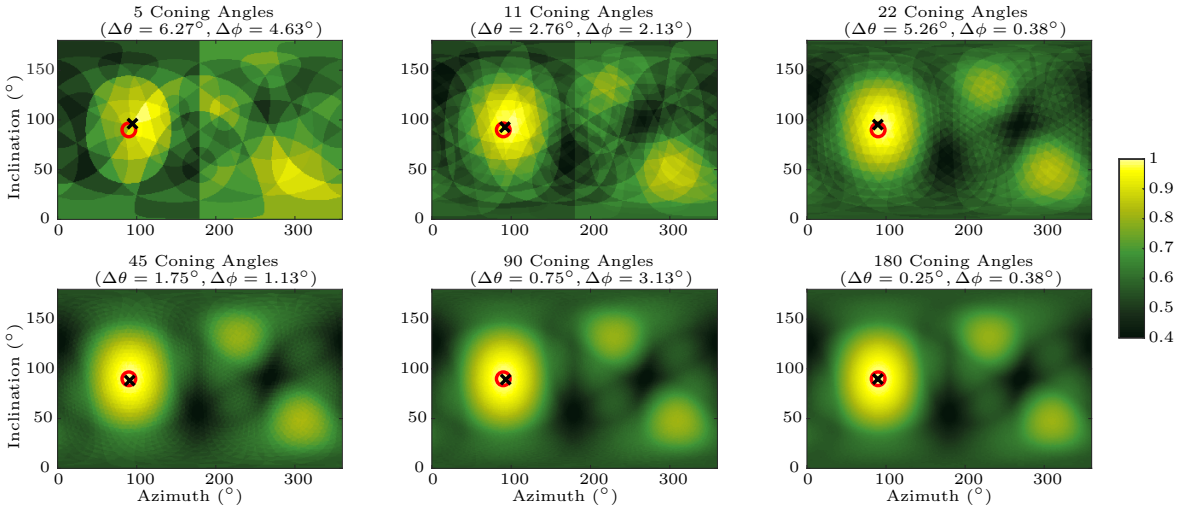

Figure 4.7: Illustration of how the coning angle resolution of element pair decomposition (EPD) beamforming affects reconstruction – since we use nearest neighbor summation, in which we find the nearest coning angle in each element pair to reconstruct the output at a desired look-angle, the number of coning angles has a significant impact on the quality of the output. The ∞ dB incoming signal is incident from the look-angle of $\theta, \phi = 90°$, shown as the red circle; the maximum output of EPD beamforming is shown as the black cross, with the difference in inclination and azimuth from the true direction shown as $\Delta\theta$ and $\Delta\phi$ in the title of each plot. For a small number of coning angles, the total output is low-quality and 'discretized', with a pattern generated by the intersections of coning angles from all element pairs, a pattern that is unique depending on the geometry of the array. As the number of coning angles increases, the reconstruction becomes 'finer'.

of Fig. 4.7 with a step size of 1°), the directional error is just 0.25° in inclination and 0.38° in azimuth.

4.5 Comparing EPD and Conventional Beamforming

Now that we have formulated the EPD beamforming approach, we are interested in how it compares to the conventional beamformer (CBF) in terms of fundamental properties such as its beampattern, accuracy, angular resolution, and processing speed. In this section we provide a very limited comparative analysis between the two approaches, noting fundamental differences. To compare the two approaches we use the base parameters listed in table 4.1. EPD beamforming has unique advantages and disadvantages, and a deeper investigation of its properties may provide a rich avenue for future work – however, an extensive analysis of our approach is beyond the scope of this thesis.

$\mathbf{f_s}$	$\mathbf{c}$	$\mathbf{S_{angles}}$	$\mathbf{S_{coning}}$
37500 Hz	1481 $\mathrm{m\,s^{-1}}$	$\theta = 0^\circ : 1^\circ : 179^\circ$ $\phi = 0^\circ : 1^\circ : 359^\circ$ (180×360 look-angles)	$\zeta = 0^\circ : 1^\circ : 179^\circ$ (180 coning angles)

Table 4.1: Base parameters for comparative analyis of EPD and conventional beamforming.

4.5.1 Beampattern

To begin, we first compare the beampatterns output by both methods; to do so, we compare the outputs using the array geometries listed (in cm) in table 4.2. We include the 1D uniform linear array in order to more easily visualize the beamformed output in 2D – for this array, we beamform over a single inclination angle.

Element	ULA	Reg. Tetrahedron	Reg. Octahedron	Reg. Cube
1	$(0, 0, 0)$	$(0, 0, 6.54)$	$(0, 0, 5.66)$	$(4, 4, 4)$
2	$(8, 0, 0)$	$(4.62, 0, 0)$	$(4, 4, 0)$	$(-4, 4, 4)$
3	$(16, 0, 0)$	$(-2.31, -4, 0)$	$(-4, 4, 0)$	$(4, -4, 4)$
4	$(24, 0, 0)$	$(-2.31, 4, 0)$	$(4, -4, 0)$	$(-4, -4, 4)$
5	$(32, 0, 0)$	–	$(-4, -4, 0)$	$(4, 4, -4)$
6	$(40, 0, 0)$	–	$(0, 0, -5.66)$	$(-4, 4, -4)$
7	$(48, 0, 0)$	–	–	$(4, -4, -4)$
8	$(56, 0, 0)$	–	–	$(-4, -4, -4)$

Table 4.2: Array geometries element positions (x, y, z) (in cm) to compare beamformed outputs of EPD and conventional beamforming.

For each of these arrays, we simulate the incoming acoustic plane wave using a 8250–10250 Hz, 20 ms LFM up-chirp, incident onto the array from a randomly selected azimuth and inclination look-angle; each element of the array records 8000 samples, which represents 213.3 ms of data at our sampling rate, 20 ms of which contains the incident chirp; the center (or operating) frequency of 9250 Hz is twice the wavelength of the separation distance of each element in these arrays, which is 8 cm. A few representative examples of the beamformed outputs (or array responses) from both approaches are shown on the next few pages, in Figs. 4.8, 4.9, 4.10, and 4.11. These figures plot the output from conventional beamforming in the top row, the output from EPD beamforming in the middle row, and the difference between the two in the bottom row; in these plots, the true look-angle is plotted as the red circle (or solid line), and the estimates from the CBF and from EPD beamforming are plotted as blue and black crosses (or dashed lines) respectively. The array responses plotted in these figures are normalized such that their maximum is 1, which is the usual convention for such plots.

Looking at these figures, we see that the array response from conventional beamforming and from EPD beamforming share many similar features, especially in terms of the position

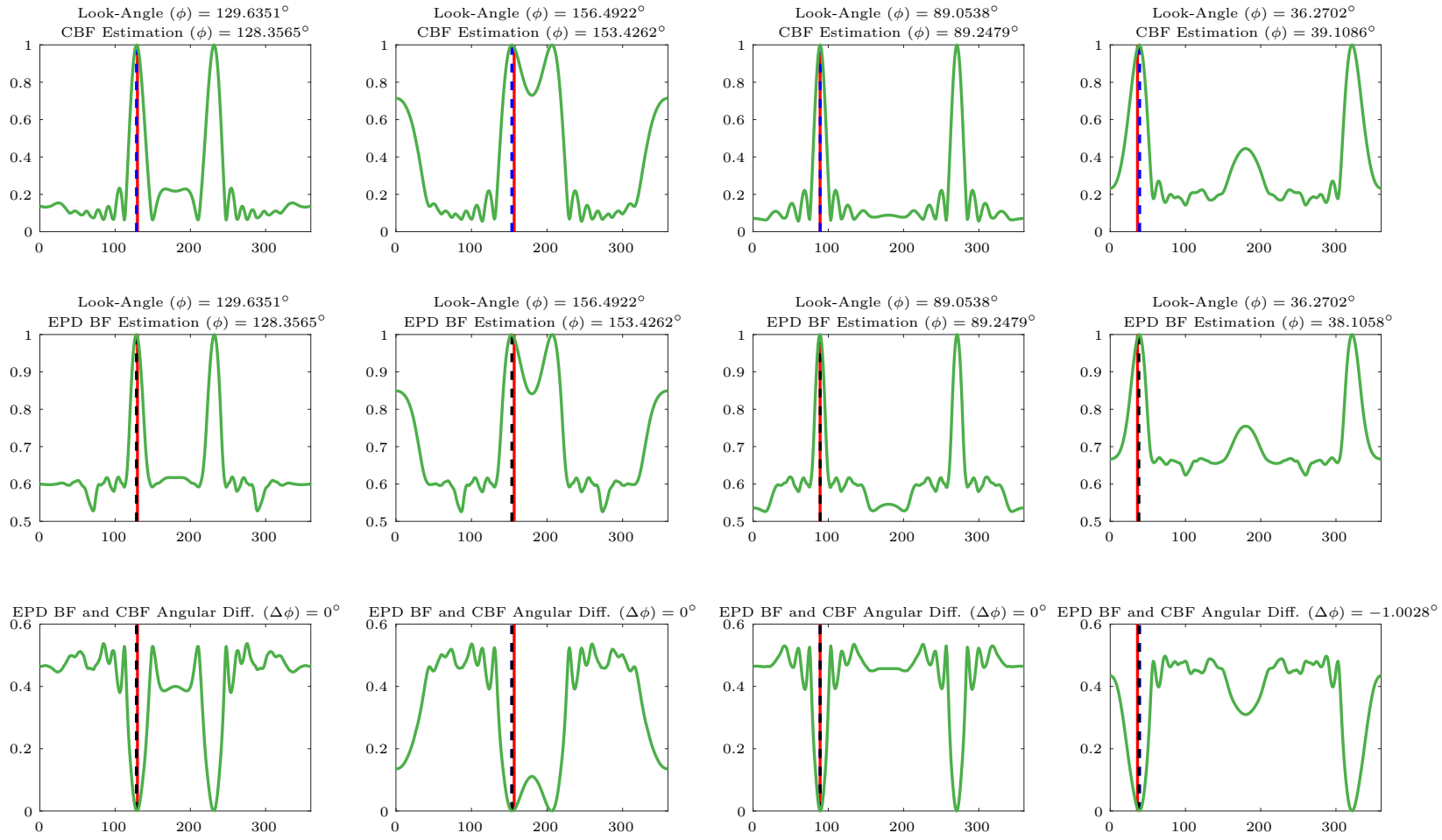

Figure 4.8: Comparison of sample array responses from conventional and EPD beamforming for a 8 element uniform line array – the true direction of arrival (DOA) of the incoming acoustic signal is indicated by the solid red line, and the estimates from maximum likelihood estimate (MLE) using conventional and EPD beamforming are indicated by the dashed blue and black lines respectively. *Top Row*: array response from conventional beamforming. *Middle Row*: array response from EPD beamforming. *Bottom Row*: subtracting the conventional beamforming response from the EPD beamforming response.

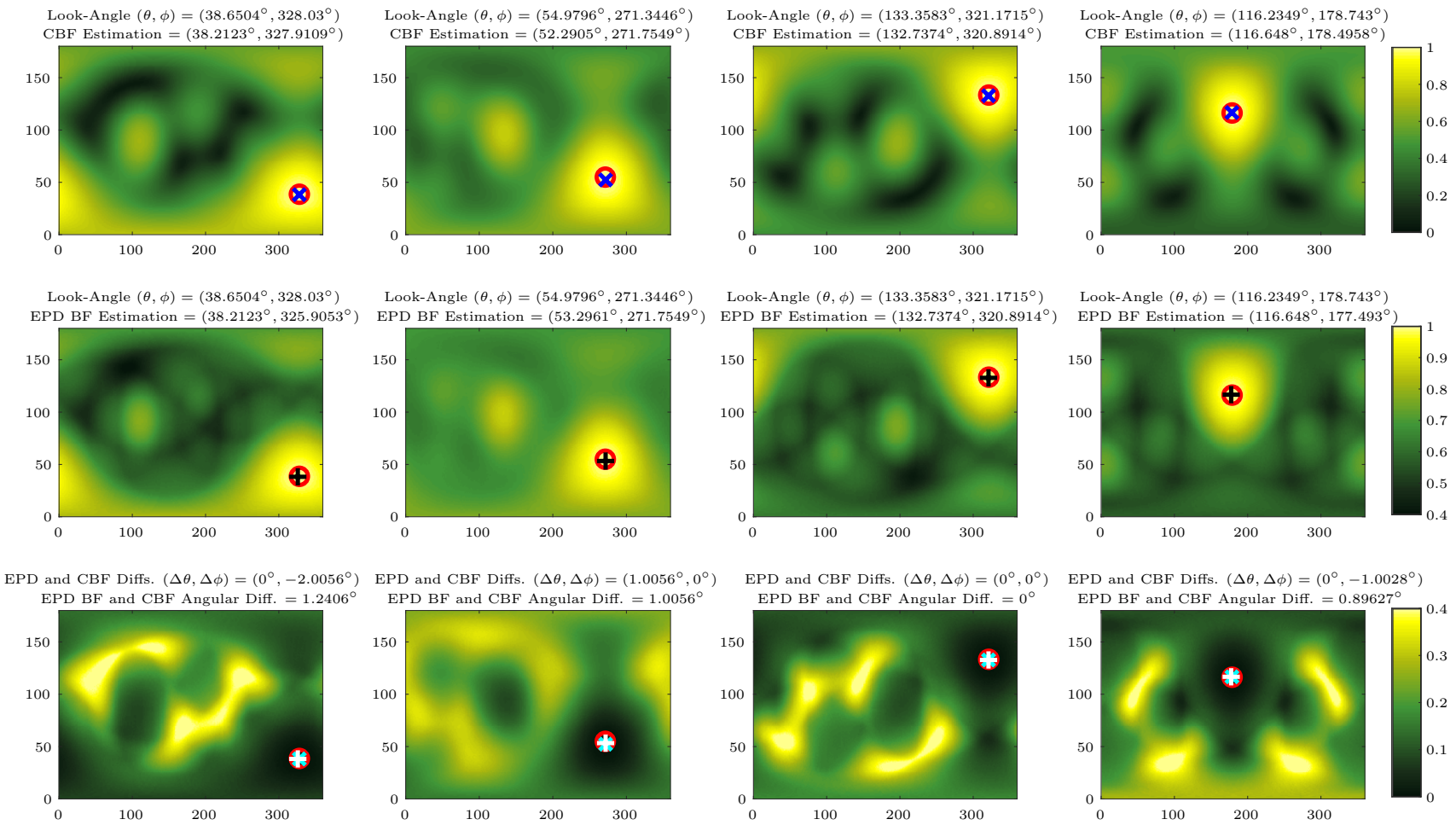

Figure 4.9: Comparison of sample array responses from conventional and EPD beamforming for a 4 element regular tetrahedral array – the true direction of arrival (DOA) of the incoming acoustic signal is indicated by the red circle, and the estimates from MLE using conventional and EPD beamforming are indicated by the dashed blue (cyan in the bottom row) and black (white in the bottom row) crosses respectively. *Top Row*: array response from conventional beamforming. *Middle Row*: array response from EPD beamforming. *Bottom Row*: subtracting the conventional beamforming response from the EPD beamforming response.

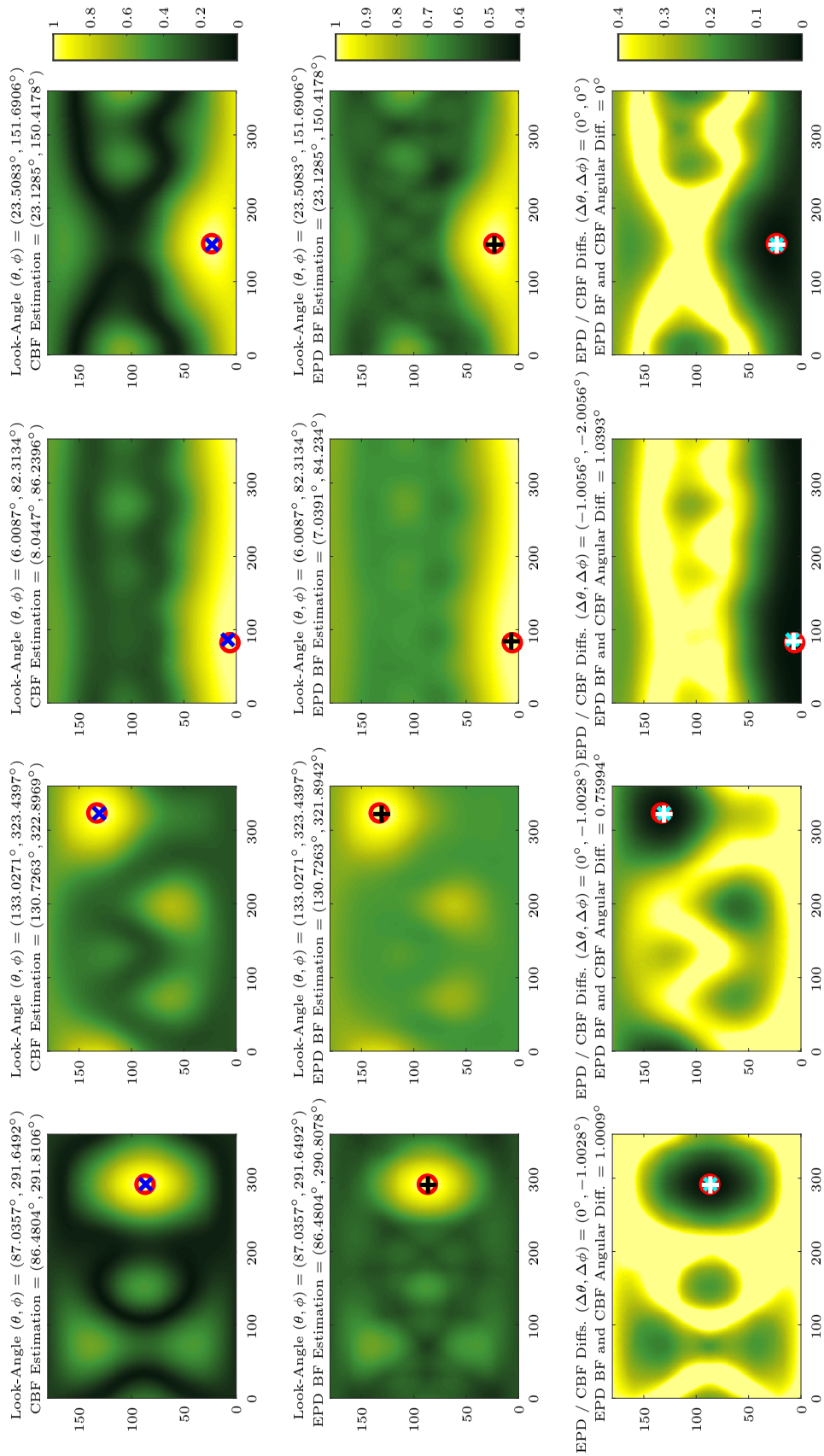

Figure 4.10: Comparison of sample array responses from conventional and EPD beamforming for a 6 element regular octahedron array – the true direction of arrival (DOA) of the incoming acoustic signal is indicated by the red circle, and the estimates from MLE using conventional and EPD beamforming are indicated by the dashed blue (cyan in the bottom row) and black (white in the bottom row) crosses respectively. *Top Row*: array response from conventional beamforming. *Middle Row*: array response from EPD beamforming. *Bottom Row*: subtracting the conventional beamforming response from the EPD beamforming response.

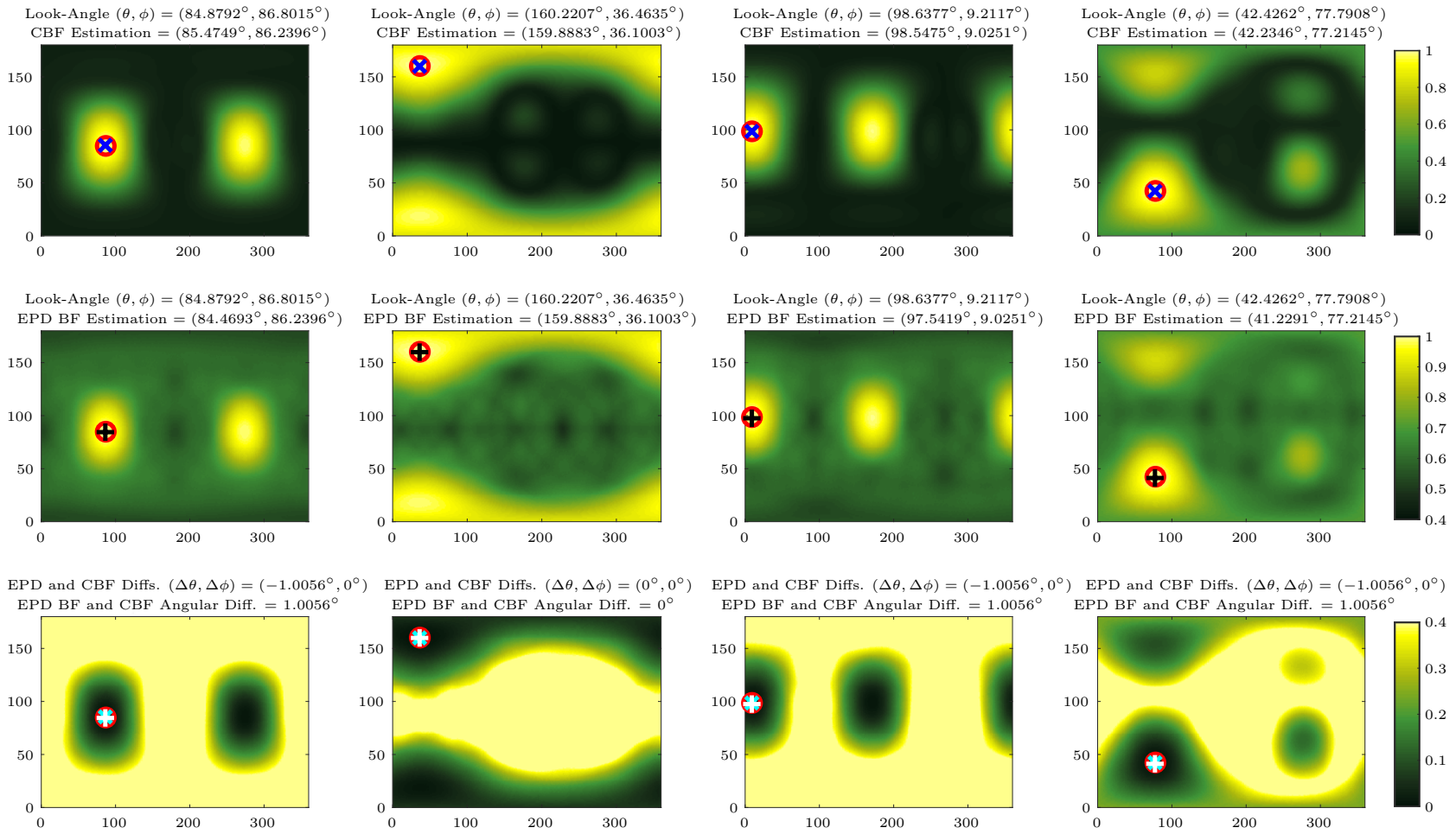

Figure 4.11: Comparison of sample array responses from conventional and EPD beamforming for a 8 element regular cube array – the true direction of arrival (DOA) of the incoming acoustic signal is indicated by the red circle, and the estimates from MLE using conventional and EPD beamforming are indicated by the dashed blue (cyan in the bottom row) and black (white in the bottom row) crosses respectively. *Top Row*: array response from conventional beamforming. *Middle Row*: array response from EPD beamforming. *Bottom Row*: subtracting the conventional beamforming response from the EPD beamforming response.

of maxima. Looking at the last rows, which illustrate the difference between the response from EPD beamforming and the response from conventional beamforming, we see that the minimum occurs close to the true direction of arrival (DOA), indicating that both approaches share a similar level of accuracy. Note the increased 'floor' of EPD beamforming, caused by the summation of the floors from CBF beamforming of element pairs – it is apparent by looking from Fig. 4.9 to Fig. 4.10 and to Fig. 4.11, that as the number of elements in the array increases (from 4 to 6 and to 8), that the floor of the response continues to increase; this is especially apparent looking at the difference plots, which shows how the area with a difference larger than 0.4 continues to increase with the addition of more elements to the array. At the same time however, it appears that (as with conventional beamforming) the width of the main lobe continues to decrease with an increasing number of elements. The increase in the level of the floor of EPD beamforming suggests that the half-power beamwidth is increased using this approach, which may indicate a reduction in the resolving power of the array; however, this may not necessarily be true, since this ability is tied more closely to the null-to-null beamwidth [158], which suggests that the total *range* (peak-to-trough) of the response of the array is more important – this parameter is difficult to estimate however.

4.5.2 Accuracy

To estimate the accuracy of both approaches, the simulated measurement on each element is corrupted with noise. White Gaussian noise is added to the signal received by each element, such that the SNR is randomly set to anywhere between 0 dB and 25 dB for each simulation. 200 simulations were performed for each array, and the angular difference between the true DOA of the simulated plane wave and the MLE value from the response of both conventional and EPD beamforming was determined using:

$$\epsilon = \arccos(\boldsymbol{v}_{mle} \bullet \boldsymbol{v}_{true}) \tag{4.23}$$

WHERE

$$\boldsymbol{v}_{true} = \begin{bmatrix} \sin(\theta_{true})\cos(\phi_{true}) \\ \sin(\theta_{true})\sin(\phi_{true}) \\ \cos(\theta_{true}) \end{bmatrix} \quad \text{AND} \quad \boldsymbol{v}_{mle} = \begin{bmatrix} \sin(\theta_{mle})\cos(\phi_{mle}) \\ \sin(\theta_{mle})\sin(\phi_{mle}) \\ \cos(\theta_{mle}) \end{bmatrix} \tag{4.24}$$

where θ_{true} and ϕ_{true} are the true inclination and azimuth of the simulated incoming plane wave, and θ_{mle} and ϕ_{mle} are the arg-maximum inclination and azimuth from the output of either conventional or EPD beamforming. The accuracy statistics resulting from these simulations are plotted as boxplots in Fig. 4.12.

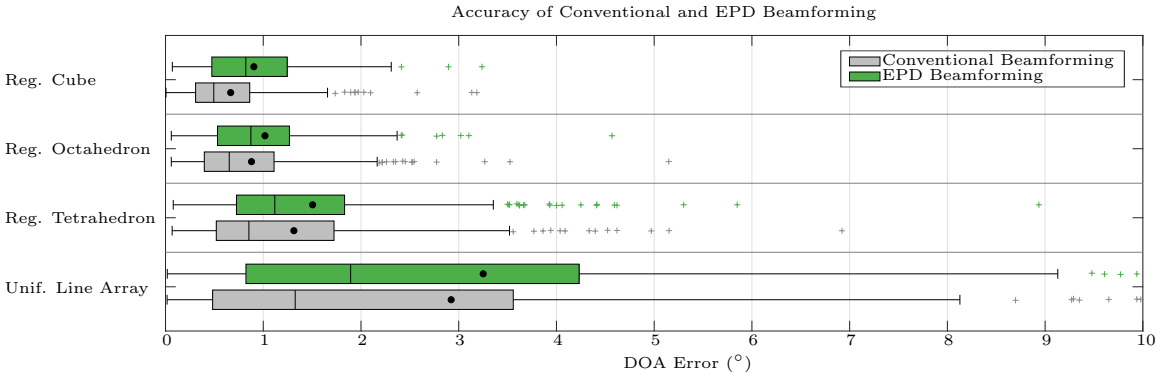

Figure 4.12: Comparison of accuracy of conventional and EPD beamforming for our four array geometries – this accuracy is calculated as the angular difference between the true look-angle and the MLE value look-angle from both approaches over 200 simulations; the median values are plotted as the solid line in the center of each boxplot, while the mean values are plotted as solid black circles; the box edges extend to the 25^{th} and 75^{th} percentiles, while the crosses indicate outliers.

Looking at this figure, it is apparent that as the number of elements in the 3D array increases (from the tetrahedron to the octahedron to the cube), the accuracy improves for both approaches. This is intuitive, since with the addition of more measurements the array becomes more robust to noise – the noise on each element tends to 'average out', since they are uncorrelated. As expected, the uniform line array, despite having the same number of elements as the cube, has the widest variation in accuracy, due to the degradation in the array response as the source moves from broadside to end-fire. This plot demonstrates that although EPD beamforming has a similar level of accuracy as the CBF for all array geometries, its accuracy on average is slightly worse – this is likely a combination of a number of factors, including the number of coning angles used to 'reconstruct' the output over the desired look-angles, as well as the increase in the 'floor' of the response and the subsequent increase in beamwidth.

4.5.3 Half-Power Beamwidth

The half-power beamwidth of the main lobe represents the -3 dB level from the peak, or about the 0.707 level in the array response of both approaches. We can estimate the half-power beamwidth using the equations:

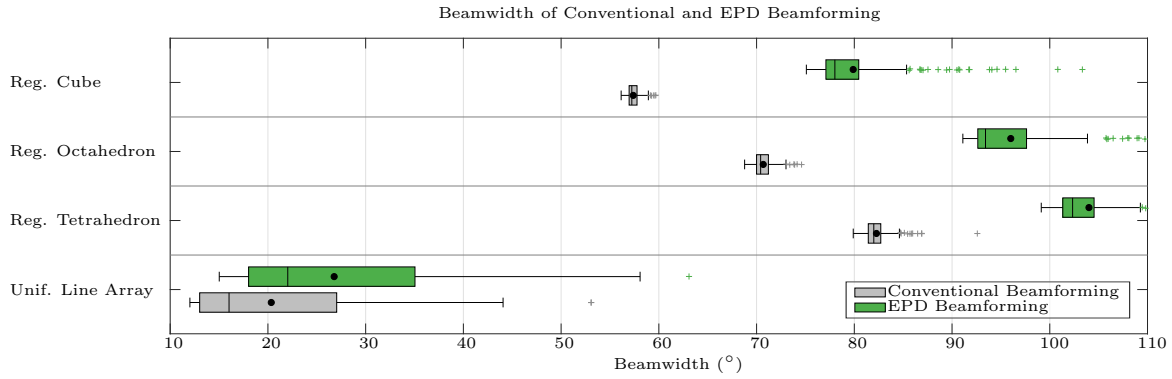

Figure 4.13: Comparison of the half-power beamwidth of conventional and EPD beamforming for our four array geometries – this beamwidth is calculated as the angular difference between the MLE value look-angle and the nearest -3 dB look-angle from both approaches over 200 simulations; the median values are plotted as the solid line in the center of each boxplot, while the mean values are plotted as solid black circles; the box edges extend to the 25^{th} and 75^{th} percentiles, while the crosses indicate outliers.

$$BW_{3dB} = \min_{\boldsymbol{v}_{3dB} \in S_{3dB}} \left(\arccos(\boldsymbol{v}_{3dB} \bullet \boldsymbol{v}_{true})\right) \tag{4.25}$$

WHERE

$$\boldsymbol{v}_{true} = \begin{bmatrix} \sin(\theta_{true})\cos(\phi_{true}) \\ \sin(\theta_{true})\sin(\phi_{true}) \\ \cos(\theta_{true}) \end{bmatrix} \quad \text{AND} \quad \boldsymbol{v}_{3dB} \in S_{3dB} = \begin{bmatrix} \sin(\theta_{3dB})\cos(\phi_{3dB}) \\ \sin(\theta_{3dB})\sin(\phi_{3dB}) \\ \cos(\theta_{3dB}) \end{bmatrix} \tag{4.26}$$

where S_{3dB} is the set of all azimuth and inclinations that occur at the -3 dB (or 0.707) level. The statistics for the estimated beamwidth over 200 simulations with added white Gaussian noise are shown as boxplots in Fig. 4.13.

This figure clearly illustrates the increase in the half-power beamwidth resulting from the array response using EPD beamforming – for all array geometries, there is a degradation in this value when using our approach. This is due to the increase in the 'floor' of the response, which is a direct result of the summation of element pair beamformed outputs, as described previously. This increase in the beamwidth may suggest a loss of resolving power (or a reduction in the ability for the array to resolve two closely-spaced sources) when using EPD beamforming, although this may not necessarily be true. Resolving power is better related to the null-to-null beamwidth, which suggests that the entire *range* of the array response should be normalized in some way (since this value is a measure between the peak and the trough of the response). Unfortunately, the null-to-null parameter is difficult to estimate for EPD beamforming, since the output of individual element pairs must be frequency-averaged and

made absolute before summing over element pairs, meaning that the location of troughs are difficult to track. Further work to determine the effect of EPD beamforming on the resolving power of the array is necessary in the future.

This figure also illustrates the improvement in beamwidth as more elements are added to the 3D array – as expected, the beamwidth reduces with more elements, since the addition of elements provides the array with more 'evidence' that the received signal arrives from a certain direction. This improvement occurs for both conventional and EPD beamforming, although the increase in beamwidth from EPD beamforming per array geometry is significant (between 20° and 25°). Since the geometry of the line array is such that it provides significant resolving power in one direction (broadside), it has both the narrowest half-power beamwidth, and the most variation in beamwidth, since beamwidth increases as the source moves from broadside to end-fire.

4.5.4 Speed-Up Factor

To illustrate the speed-up factor that EPD beamforming achieves, we simply measure the computation time of both approaches on a standard laptop computer[4]. Note that for the three 3D arrays (tetrahedron, octahedron and cube), we beamform over the full grid of 180×360 look-angles, with 180 coning angles used for EPD beamforming; for the uniform line array, we only beamform over 360 azimuths, at a single inclination of 90° (again we use 180 coning angles). For our 4-element tetrahedral array we have 6 element pairs, for our 6-element octahedral array we have 15 element pairs, and for our 8-element line array and cube array we have 28 element pairs. The statistics of the computation time of both approaches are shown as boxplots in Fig. 4.14, with the lower plot showing a zoomed-in section of the upper plot.

For the three 3D arrays, this figure illustrates the enormous speed up achieved by EPD beamforming when evaluating the output over a full 2D grid of look-angles. Note that for both approaches, there is a increase in the computation time as we increase the number of elements in the array – this is expected, since the computation of both involves more multiplication-adds as more elements are added to the array. However, note the more than order of magnitude decrease in computation time of EPD beamforming over the CBF – this significant speed up is due to one simple fact: the CBF must be evaluated over the full grid of $180 \times 360 = 64800$ look-angles, while EPD beamforming must be evaluated only $6 \times 180 = 1080$ times, $15 \times 180 = 2700$ times, or $28 \times 180 = 5040$ times, for the tetrahedral, octahedral and cube arrays respectively; the output over the full grid of look-angles is then easily generated using nearest-neighbor summation, which is very quick.

[4]The laptop used has 8GB of memory, and an Intel Core i7-3630QM CPU @ 2.40GHz × 8.

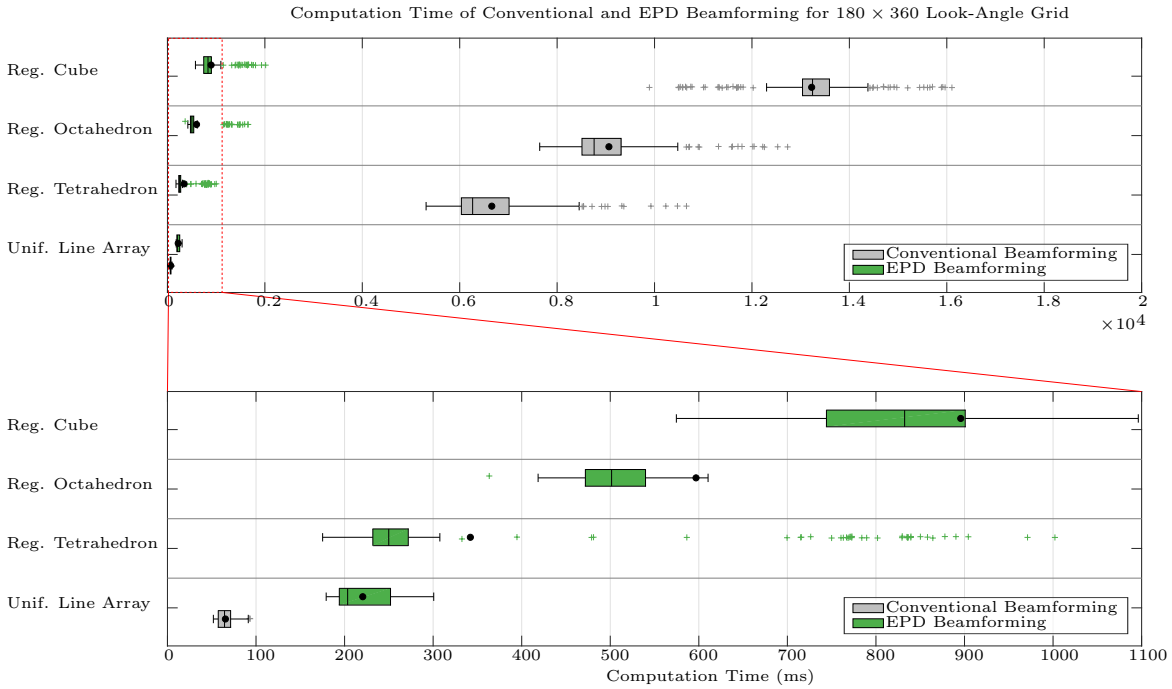

Figure 4.14: Comparison of the computation times of conventional and EPD beamforming for our four array geometries – this computation time is measured on a standard laptop computer from both approaches over 200 simulations; the median values are plotted as the solid line in the center of each boxplot, while the mean values are plotted as solid black circles; the box edges extend to the 25^{th} and 75^{th} percentiles, while the crosses indicate outliers. The lower plot shows the zoomed-in portion outlined by the dashed red box in the upper plot.

For the uniform line array, we notice something very different – conventional beamforming is more than twice as fast as EPD beamforming. This is because there is absolutely no gain to be had over the CBF when beamforming over a single dimension, since the CBF only has to be evaluated 360 times, while EPD beamforming must be evaluated $28 \times 180 = 5040$ times. This disadvantage of EPD beamforming is also representative of the lack of speedup when evaluating the beamformer within the sequential Monte-Carlo beamformer (SMCB) – the number of look-angles is fixed, since it is set to the number of particles in the filter. However, EPD beamforming still enables a major reduction in memory use, which subsequently enables a much finer resolution for the grid of look-angles – this is because the phase shifts over this grid must be precomputed and stored in the conventional beamforming case, but not in the EPD beamforming case.

4.5.5 Look-Angle Resolution, Accuracy, Speed and Memory Usage

To better understand the relationship between the resolution of the look-angle grid and the accuracy of the subsequent DOA estimate from both beamforming solutions, we vary the grid

resolution and note its impact on the accuracy of the estimate and the amount of time the solution takes to compute, for both conventional and EPD beamforming. In addition, we make a brief comment on the amount of memory used by both approaches as we vary the grid resolution. In this analysis, we use the configurations listed in table 4.3 below – the 360° domain in azimuth is divided equally into N_ϕ angles, and the 180° domain in inclination is divided equally into N_θ angles, resulting in a $N_\theta \times N_\phi$ grid of look-angles; the number of coning angles in the 180° coning angle space is set equal to the number of azimuthal angles, so that we improve the accuracy of EPD beamforming to a level more in line with the accuracy of conventional beamforming. We are primarily interested in the improvement in speed and memory use of EPD beamforming over conventional beamforming for a given level of accuracy.

config	$\mathbf{S_{angles}}$ $(N_\theta \times N_\phi)$	$\mathbf{S_{coning}}$ (N_ϕ)
1	$23 \times 45 = 1035$ look-angles	45 coning angles
2	$45 \times 90 = 4050$ look-angles	90 coning angles
3	$68 \times 135 = 9180$ look-angles	135 coning angles
4	$90 \times 180 = 16200$ look-angles	180 coning angles
5	$113 \times 225 = 25425$ look-angles	225 coning angles
6	$135 \times 270 = 36450$ look-angles	270 coning angles
7	$158 \times 315 = 49770$ look-angles	315 coning angles
8	$180 \times 360 = 64800$ look-angles	360 coning angles

Table 4.3: Look-angle configurations for comparative analysis of speed and memory usage.

We again use the same simulation of an incoming LFM chirp, with the measurements corrupted by white Gaussian noise to achieve random SNRs between 0 dB and 25 dB. However, in this case, we select the incoming angle randomly from the set of look-angles for a given configuration in table 4.3. For each configuration listed, we run 200 simulations for both beamforming approaches, using the regular tetrahedral, octahedral and cube array geometries. Angular error for each simulation is calculated using Eq. 4.24, and we calculate the root-mean-square of the errors collected over all 200 runs, using this value as the representative error for the beamforming method in that configuration. We also calculate the mean computation time over all 200 runs, and use this value as the representative computation time for the beamforming method in that configuration.

The plot of Fig. 4.15 illustrates the root-mean-square of the angular errors for conventional beamforming as dashed lines and for EPD beamforming as solid lines. Again we see that as the number of elements in the array increases, the accuracy improves and the error reduces. With the number of coning angle equal to the number of azimuthal angles in the look-angle grid, we see that the difference in accuracy of EPD beamforming and conventional beamforming is negligible, with EPD beamforming being very slightly less accurate.

With EPD beamforming providing the same level of accuracy as conventional beamforming for these look-angle grid configurations, we now take a look at the speedup provided by EPD

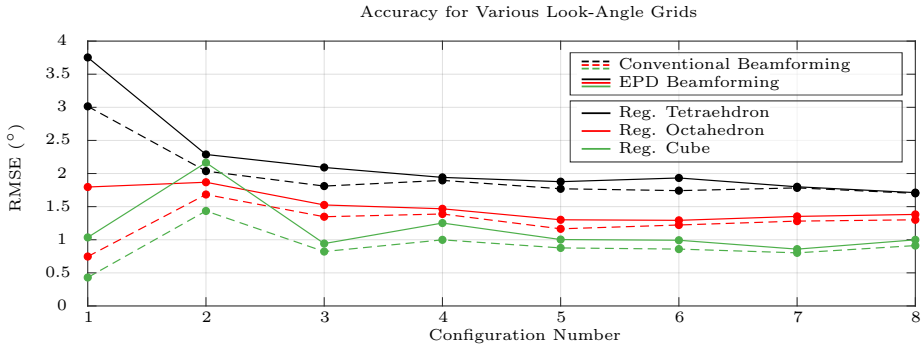

Figure 4.15: Root-mean-square of errors in direction of arrival (DOA) of conventional and EPD beamforming using look-angle grids of various resolution listed in table 4.3 – the root-mean-square of errors from conventional beamforming are shown for the three 3D array geometries as dashed lines, and for EPD beamforming as solid lines.

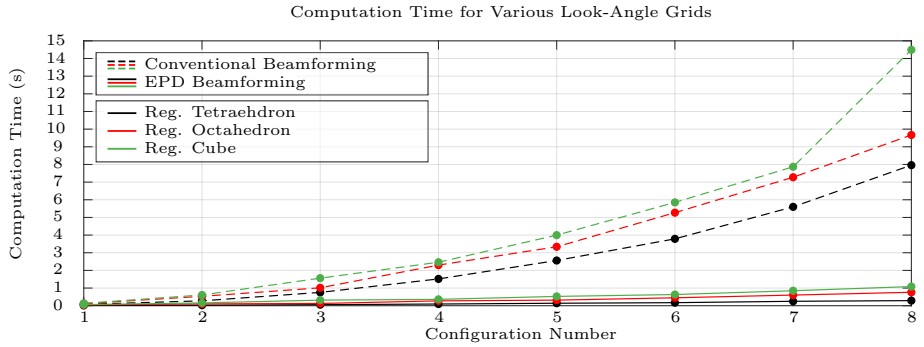

Figure 4.16: Computation time of conventional and EPD beamforming using look-angle grids of various resolution listed in table 4.3 – the computation time from conventional beamforming are shown for the three 3D array geometries as dashed lines, and for EPD beamforming as solid lines.

beamforming. The run compute times for each of the configurations is plotted in Fig. 4.16, with those of conventional beamforming shown as dashed lines, and EPD beamforming as solid lines. It is apparent that EPD beamforming is significantly more computationally efficient. This figure demonstrates that as the resolution of the look-angle grid increases, the computation time for the CBF increases quadratically, in line with the quadratic increase in the number of look-angles in the grid; conversely, for EPD beamforming, the compuation time increases linearly with the number of coning angles used. The speedup factor for computing these various grids of look-angles is plotted in Fig. 4.17, shown for each array geometry – the gain in speed increases with the increasing grid resolution, with EPD beamforming able to speed up the computation time of a 180×360 grid of look-angles by more than 10 times without a loss of accuracy.

Finally, let us note the amount of memory used by each beamforming approach for the configurations listed in table 4.3. For conventional beamforming, we must store the phase

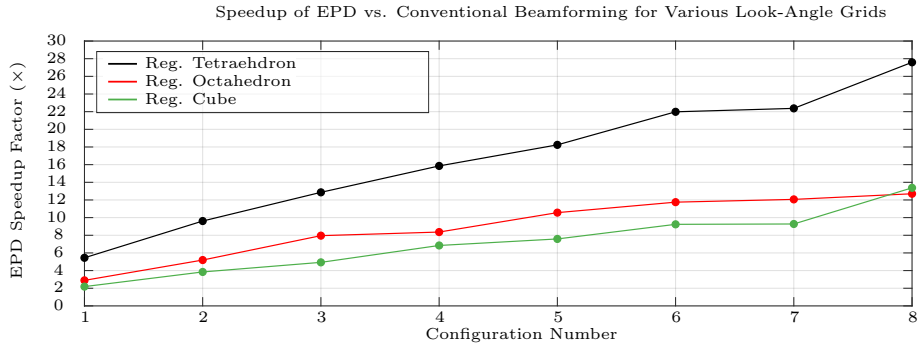

Figure 4.17: Speedup of EPD over conventional beamforming using look-angle grids of various resolution listed in table 4.3.

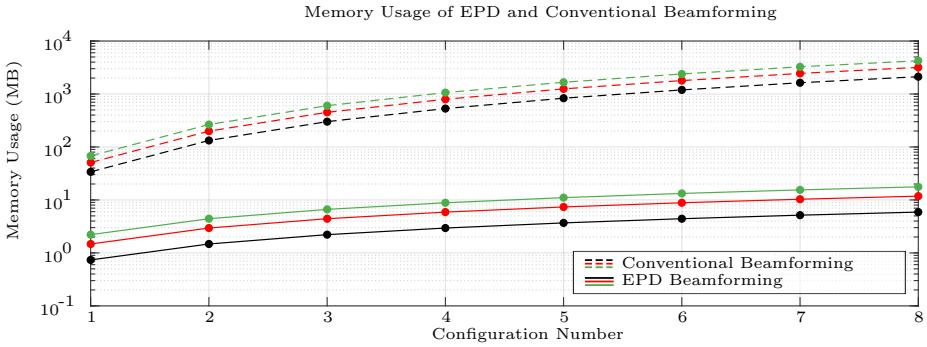

Figure 4.18: Memory usage of EPD and conventional beamforming using look-angle grids of various resolution listed in table 4.3.

shifts over all M frequencies for each look-angle in the grid and for all N elements in the array – this requires $N_\theta \times N_\phi \times M \times N$ complex doubles. On the other hand, EPD beamforming only requires us to store the phase shifts over all M frequencies for all coning angles belonging to all *unique* element pairs, since the phase shifts for element pairs with the same separation distance between elements need only be stored once. For the regular tetrahedral array, there is only 1 unique pair (since all element pairs have the same separation distance); for the regular octahedral array, there are only 2 unique pairs (the element pairs making up the edges, and the pairs made of elements at opposite vertices); and for the regular cube array, there are only 3 unique pairs (the element pairs making up the edges, the pairs across diagonals of the faces, and the pairs that are connected via the center of the cube). Thus the memory required by EPD beamforming is significantly smaller, and is equal to storing $N_\phi \times M \times P_{unique} \times 2$ complex doubles, where P_{unique} is the number of unique element pairs, and there is the factor of 2 for each element in the pair. The plot of Fig. 4.18 illustrates the amount of memory required by each beamforming approach for the look-angle grids of varying resolution listed in table 4.3; note that we use $M = 512$ for this figure.

This figure illustrates something remarkable – the memory savings of EPD beamforming

are enormous, with the memory required being more than 2 *orders of magnitude less* than conventional beamforming for a 180×360 grid of look-angles. This is the *key advantage* of EPD beamforming that enables a significant improvement in the *precision* of our angle measurement distribution. In our prototypical piUSBL system described in chapter 2, and even with our improved sequential Monte-Carlo beamformer (SMCB) for filtering, we store the phase shifts associated with a grid of look-angles when using the conventional beamformer (CBF) – this is a necessity in order to beamform at real-time rates on a Raspberry Pi 3; however, due to the limited memory on-board the computer, the resolution of this grid of look-angles is low, limiting piUSBL accuracy due to the loss of angular precision. By using EPD beamforming, we only need to store a much smaller number of phase shifts associated with the set of coning angles for each unique element pair – this allows us to reconstruct the beamformed output at any *arbitrary* look-angle [5] at a much finer resolution.

4.6 Summary

In this chapter we have described a novel beamforming method that we call element pair decomposition (EPD) beamforming. This approach replaces the conventional beamformer (CBF) in our piUSBL system, and drastically improves the precision of our acoustic angle measurement distribution. The key insight of EPD beamforming is that any given look-angle for an arbitrary 3D array can be described in the coning angle space of the element pairs that make up the array using a single coning angle rather than using two angles (an inclination and an azimuth angle). This insight allows us to collapse the search space of a grid of look-angles from two dimensions to one – rather than beamforming at every look-angle on a grid, we can instead beamform at every coning angle on a line for each element pair, and reconstruct the original grid from these outputs. This enables significant improvements in computational efficiency and memory usage, with the improvement in memory usage allowing us to perform beamforming at a finer resolution. We described EPD beamforming in detail, commented on its unique advantages, disadvantages and features, and provided statistics of its accuracy, beamwidth, computational speedup and memory usage as compared to conventional beamforming.

This novel beamforming method is the final technical improvement to our prototypical piUSBL system. In the next chapter we describe the implementation of our final piUSBL system modified with the improvements of the sequential Monte-Carlo beamformer (SMCB) and element pair decomposition (EPD) beamforming on an autonomous surface vehicle (ASV) and on a fleet of commercial SandShark AUVs, and provide a comprehensive analysis of the accuracy of our piUSBL system for localization.

[5] The look-angle doesn't necessarily have to be part of a grid of look-angles, but may also, for example, be at the look-angle represented by a particle in the SMCB.

Chapter 5

Testing the Pipeline: *System Verification and Evaluation*

5.1 Introduction

Results in preceding chapters have demonstrated that our Passive Inverted Ultra-Short Baseline (piUSBL) positioning system has enabled closed-loop navigation for a low-cost, miniature autonomous underwater vehicle (AUV), and has also enabled a relative homing capability for a conventional, Doppler velocity log (DVL)-equipped 21–inch diameter large AUV. The piUSBL system has functioned as predicted during these operations; however, we have yet to quantify the accuracy of this approach against ground-truth data. With the additional increased precision provided by element pair decomposition (EPD) beamforming, we set out to evaluate and verify the accuracy of our final piUSBL positioning system stack. In this chapter, we detail the implementation of the piUSBL system on a WAM-V autonomous surface vehicle (ASV) equipped with differential GPS (DGPS). Ground-truth position provided by DGPS enables the comprehensive analysis of the accuracy of our piUSBL approach using two independent piUSBL beacons for additional validation. This data provides the reader with valuable insights into the true accuracy and limitations of piUSBL positioning in real-world settings.

Additionally, this chapter details the implementation of our final piUSBL system on a fleet of three commercial SandShark AUVs, and walks through the procedure required in order to calibrate the system to mitigate local acoustic effects. Positioning results before and after calibration are provided for the interested reader.

5.2 Hardware (WAM-V Autonomous Surface Vehicle Configuration)

As outlined in chapter 2 section 2.3, the design and hardware of the piUSBL system on all platforms share many commonalities. In this section we detail the specific configuration for the WAM-V ASV.

5.2.1 Acoustic Beacons

The piUSBL system typically makes use of a single acoustic beacon, as in chapters 2 and 3. To validate our measurements of accuracy, in this chapter we use two identical beacons, allowing us to calculate two independent sets of piUSBL accuracy statistics, as well as the accuracy of passive long baseline (LBL) using the intersection of one-way travel-time (OWTT) range-only measurements to both beacons. An understanding of passive LBL accuracy will come in useful in the next chapter, where passive LBL is used to validate our piUSBL system running on three commercial AUVs.

5.2.2 Platform

To gather experimental data so as to determine the accuracy of our piUSBL system in real-world settings, the system was implemented on the WAM-V ASV[1] shown in Fig. 5.1, an ASV co-developed at MIT SeaGrant, MIT LAMSS and Olin college for the AUVSI RobotX competition [155]. The WAM-V is a 5 m long vehicle equipped with a dynamic suspension system that supports a payload bed above twin inflatable pontoons. This platform is propelled using two differential drive Torqeedo[2] Cruise motors powered by four Torqeedo[2] 26-104 batteries, and is equipped with a PC-104 computer and a Hemisphere[3] Vector V102 differential GPS (DGPS) receiver for vehicle localization and navigation. The V102 DGPS has a positioning accuracy of 1 m or less 95% of the time, and a heading accuracy of 0.75°, providing a suitably accurate ground-truth comparison for our acoustic measurements. As with all our autonomous platforms, ASV mission planning and execution is performed autonomously using the MOOS-IvP framework [156].

5.2.3 Payload

The WAM-V ASV is equipped with a piUSBL receiver, allowing us to generate range and angle measurement distributions between the ASV and either of the two acoustic beacons. Unlike previous configurations, the piUSBL receiver on the WAM-V is configured as a pyramidal

[1] https://www.wam-v.com/
[2] https://www.torqeedo.com/
[3] https://www.hemispheregnss.com/

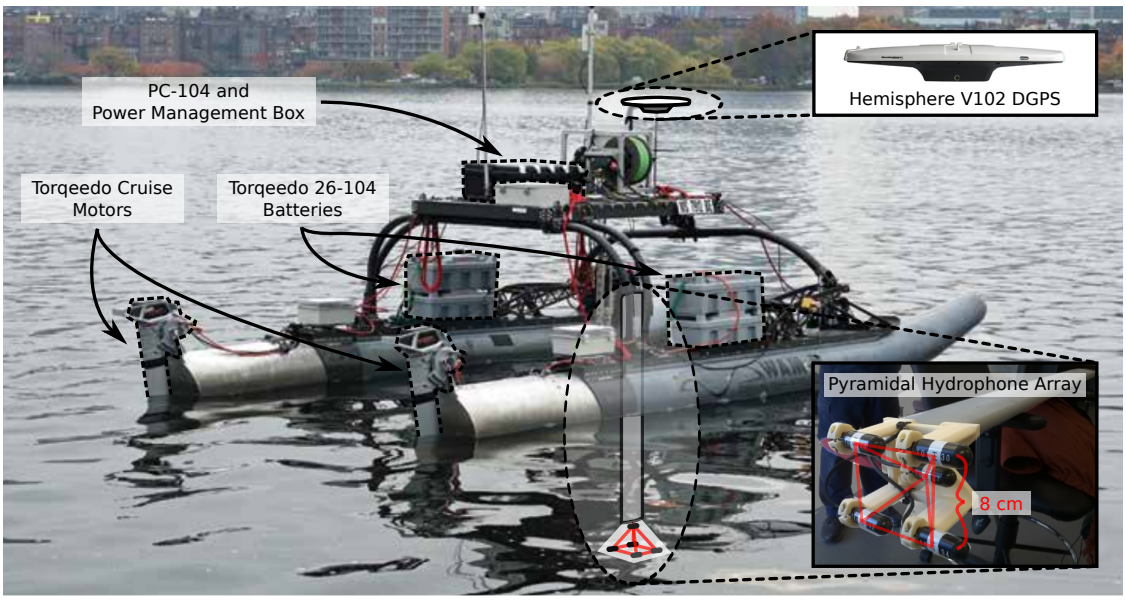

Figure 5.1: WAM-V autonomous surface vehicle outfitted with the piUSBL payload – the twin pontoon platform is equipped with differential drive Torqeedo Cruise motors powered by four Torqeedo 26-104 batteries, and navigates solely using differential GPS; it is outfitted with our piUSBL system with an 8cm pyramidal hydrophone array mounted on a submerged boom rigidly affixed to the port-side pontoon.

array, using five hydrophones rather than four in a tetrahedral configuration. Four of the five elements form the square base of the pyramid with the remaining element positioned above the center of the base such that the elements that form each pyramid edge are 8 cm apart. This pyramidal array was attached to the end of a 1.5 m aluminum boom using a 3D-printed mount (shown in the bottom right inset of Fig. 5.1). The boom was rigidly attached to the bow of the port-side pontoon. As in other receiver configurations, acoustic energy captured by the array is converted into a voltage signal and filtered using a passive resistor-capacitor bandpass circuit, and converted into a digital signal using a digital acquisition device (DAQ). However, unlike previous configurations, continuous access to the global positioning system (GPS) on the surface allows us to perform pulse-per-second (PPS) synchronization using a GPS receiver rather than a chip-scale atomic clock (CSAC); the DAQ is triggered to record using the PPS signal from a Garmin 18xLvC GPS puck. The DAQ in this case is configured to record 16000 samples of acoustic data from each element of the array at 37.5 kS/s every second, storing the data as CSV files on the on-board PC-104 computer – assuming a sound speed in freshwater of 1481 $\mathrm{m\,s^{-1}}$, this translates to a maximum range of about 632 m.

5.2.4 Receiver USBL Array and Source Signal

The greater computational efficiency afforded by the sequential Monte-Carlo beamformer (SMCB) (chapter 3) and EPD beamforming (chapter 4) provided us with enough compu-

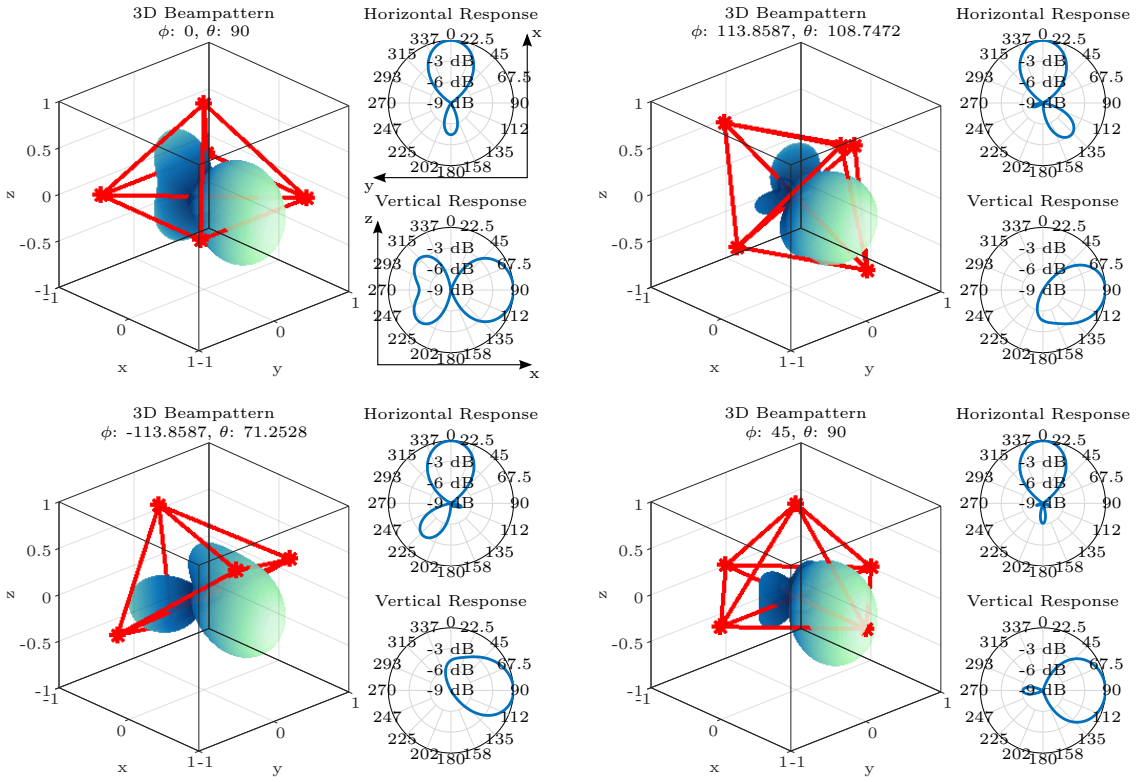

Figure 5.2: 3D beampatterns of a five element regular pyramidal array – beampatterns are shown for an incoming plane wave whose wavelength is double the array inter-element spacing (i.e. half-wavelength array spacing); as with the case for the prototypical four element tetrahedral array, the beampattern horizontal response has a near-consistent main lobe width, reflecting its circular symmetry in the azimuthal direction; however, the main lobe width varies quite dramatically in the vertical direction.

tational bandwidth to add a fifth element to the ultra-short baseline (USBL) array. This additional fifth element is advantageous, since it reduces the beamwidth of the main lobe in the array response – this allow us to improve the angular resolution of the array by reducing its 3dB (half-power) beamwidth. To retain as much as possible the spherical symmetry of the prototypical array and the associated angular-independence of the main lobe, this new array was designed as a regular pyramid with equal edge size as shown in Fig. 5.1. To demonstrate these characteristics of improved angular resolution and maintained beamwidth consistency, the theoretical 3D beampatterns for this pyramidal array are shown in Fig. 5.2; these plots illustrate the array response for an acoustic plane-wave incident from various directions, whose wavelength is double the edge size of the array. Similarly to the tetrahedral array, this new design maintains a near-consistent beamwidth in the horizontal direction regardless of the angle of the incoming acoustic energy; however, its response in the vertical direction varies quite dramatically, a result of the fact that the array is not symmetrical in this direction.

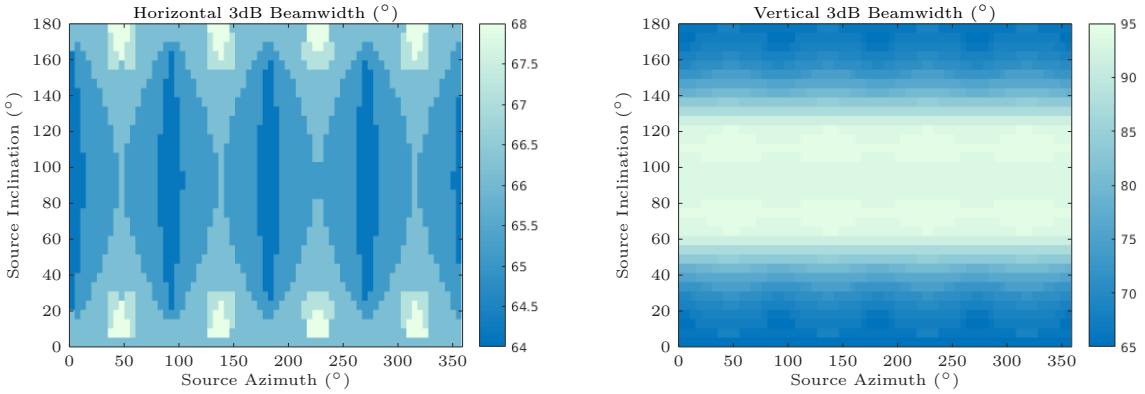

Figure 5.3: Horizontal and vertical 3dB beamwidths of a five element regular pyramidal array – as the incoming acoustic plane wave is varied in azimuth and inclination (incoming direction), the shape of the main lobe changes slightly; the left plot shows the 3dB (half-power) beamwidth in the horizontal plane, while the right plot shows the 3dB (half-power) beamwidth in the vertical plane; the beamwidth remains fairly consistent horizontally, with a marked reduction in width as compared to the tetrahedral array (between 64° and 68° which is 16° narrower); however, the vertical response shows a dramatic reduction in consistency, with the beamwidth varying up to 14° wider down to 19° narrower than the tetrahedral array, depending on inclination. Unfortunately, the vertical response is poorest in the inclination regime that we operate at (60° – 90° inclination).

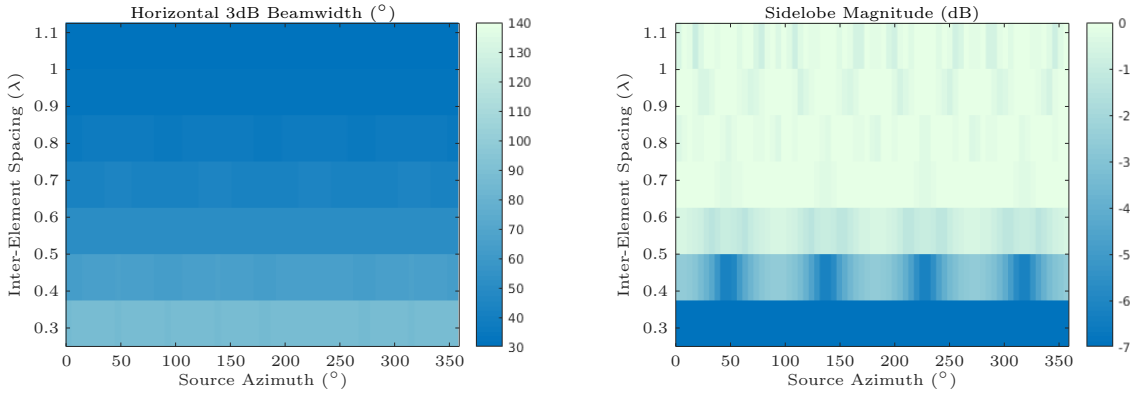

Figure 5.4: Horizontal 3dB beamwidths and sidelobe magnitudes of a five element regular pyramidal array – as the wavelength (λ) of the incoming acoustic plane wave is reduced (frequency is increased), the 3dB (half-power) beamwidth of the main lobe is reduced, thereby improving the angular resolution of the array; however, doing so increases the magnitude of the sidelobe, thereby introducing acoustic noise from unwanted directions. Note that the inclination of the source is set to a constant 90° in these plots.

This beamwidth consistency in the horizontal plane and inconsistency in the vertical plane is highlighted in Fig. 5.3, which plots the horizontal and vertical half-power beamwidths as heatmaps for various azimuth/inclination angles of the incoming acoustic wave. It is apparent

that the vertical beamwidth is highly dependent on the inclination of the incident plane-wave, while the horizontal beamwidth is consistent at around 66°, a considerable improvement over the tetrahedral array (82°).

As was done for the prototypical tetrahedral array (subsection 2.3.3), we examine the frequency-dependent trade-off between main lobe width and sidelobe magnitude by plotting these as heatmaps for a variety of azimuths and wavelengths (with respect to inter-element spacing) of the incident plane-wave. These heatmaps are shown in Fig. 5.4. For this pyramidal geometry we have a slightly greater range of frequencies to select without risking grating lobes as compared to the tetrahedral geometry – the sidelobe magnitude is more than 1dB lower than the main lobe for inter-element spacings (edge sizes) of up to 0.6λ. Between 0.4λ and 0.6λ, the half-power beamwidth is between 64° and 52°. For an 8 cm edge size, this translates to an operating frequency of $7.4 - 11.1$ kHz, which is well within the operating range of our acoustic beacons. In practice, any frequency less than 12 kHz works well with this new array geometry.

5.3 Acoustic Measurement Distributions

Raw digitized acoustic data collected by the piUSBL receiver and stored on the on-board computer is processed offline to generate range and angle measurement distributions that allow us to quantify the achievable accuracy of piUSBL. As always, we follow the matched filtering and beamforming steps detailed in algorithm 1 to process our acoustic data – however, there are a couple of minor modifications:

- Array geometry: the pyramidal array element positions for the WAM-V are given by:

$$\boldsymbol{P_{wamv}} = \begin{bmatrix} \boldsymbol{p_1^T} \\ \boldsymbol{p_2^T} \\ \boldsymbol{p_3^T} \\ \boldsymbol{p_4^T} \\ \boldsymbol{p_5^T} \end{bmatrix} = \begin{bmatrix} 0.04 & -0.04 & 0 \\ 0.04 & 0.04 & 0 \\ -0.04 & -0.04 & 0 \\ -0.04 & 0.04 & 0 \\ 0 & 0 & 0.05657 \end{bmatrix} \tag{5.1}$$

- Matched filtering: the steps followed for generating the range measurement are identical to those in algorithm 1; however, the validity check is omitted and all acoustic measurements are kept. This is because we are gathering accuracy statistics of the system offline, and so caution in removing acoustic outliers is not necessary.

- Beamforming: we replace the conventional beamformer (CBF) with our EPD beamforming algorithm detailed in chapter 4, algorithm 6. As a consequence, for the purposes of collecting statistics, we generate angle measurement distributions with a resolution

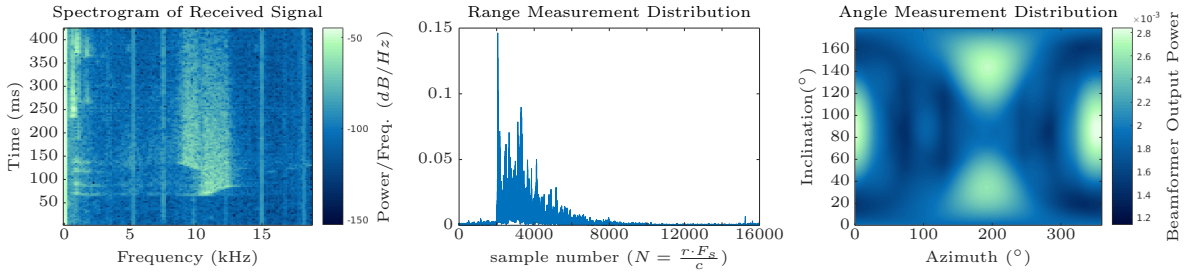

Figure 5.5: Spectrogram and example acoustic measurement distributions collected for the purposes of evaluating piUSBL accuracy – *Left*: in-water spectrogram of signals received by the WAM-V ASV, with the 11–9 kHz and 10–12 kHz LFM chirps from the two acoustic beacons clearly visible. *Center*: example range measurement distribution for distance between the array and a beacon. *Right*: example angle measurement distribution for angle between the array and a beacon. Acoustic measurements are complex and multimodal.

of 720×360, with azimuths and inclinations of $N_{azim} = 720$ and $N_{incl} = 360$; this generates angle distribution heatmaps with 259200 look-angles, at a precision of $0.5°$ in both azimuth and inclination.

Acoustic processing is performed on the two distinct linear frequency modulation (LFM) signals broadcast by our two acoustic beacons. Example range and measurement distributions resulting from this acoustic processing are illustrated in Fig. 5.5.

5.3.1 Estimating Speed of Sound

After processing the acoustic data logged during field experiments with the ASV, a series of n range measurement distributions are generated via combined matched filtering (Eqs. 2.11–2.15 in chapter 2). These steps require a conversion from the original domain of sample numbers N to the range domain r using some estimate of the speed of sound c, as stated in Eq. 2.15. Since these ASV experiments provide us with DGPS ground-truth, we can directly estimate this speed of sound through linear regression. The maximum likelihood estimate (MLE) values from each of our range measurement distributions provides us with an estimate of sample delay N_i^{MLE}, which is the range scaled by the constant factor $\frac{F_s}{c}$; given the true range r_i^{dgps} between the vehicle and the beacon from DGPS position, an estimate of this scaling factor can be determined via linear regression:

$$(\xi^*, \chi^*) = \min_{\xi,\chi} \sum_{i=1}^{n} (r_i^{dgps} - \xi - \chi N_i^{MLE})^2 \tag{5.2}$$

where ξ is the linear offset and χ is the slope. Note that since the acoustic beacon and the ASV are time-synchronized, and the delay between PPS triggering and the broadcast of the beacon is negligible, the offset ξ is expected to be very small. Performing this optimization

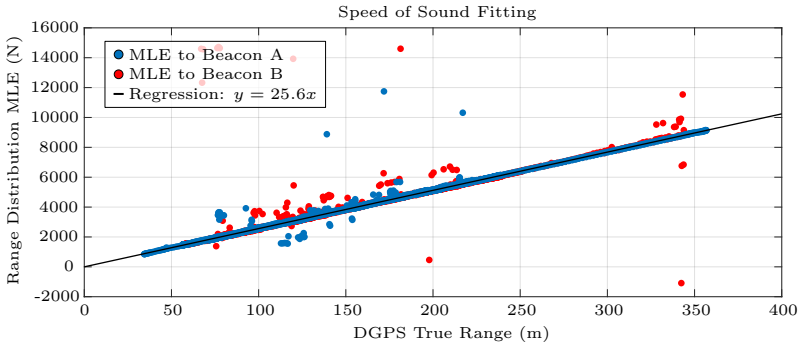

Figure 5.6: Plot of regression to determine speed of sound using MLE values from the collection of range distributions – the MLE sample offset values for beacon A are shown in blue, and for beacon B in red; the linear best fit using iteratively re-weighted least squares over all data is shown as the black line.

using iteratively re-weighted least squares over the logged dataset that we describe later in section 5.4.1, speed of sound during this experiment is calculated as:

$$\chi^* \approx \frac{F_s}{c} = 25.604 \qquad c = \frac{F_s}{\chi^*} = \frac{37500}{25.604} \approx 1464.64 \text{ m s}^{-1} \tag{5.3}$$

This sound speed is lower than that which we usually assume for experiments (1481 m s^{-1}), but agrees with the late time of year (November) at which this particular experiment was performed – the cold weather and water reduces the speed of sound in freshwater. This speed of sound regression is illustrated in Fig. 5.6, which shows the true range r_i^{dgps} vs the sample offset from MLE N_i^{MLE} for both acoustic beacons, as well as the regressed linear fit. The MLE value for range measurement i is thus given by:

$$r_i^{MLE} = \frac{1464.64 \text{ m/s}}{37500 \text{ S/s}} \cdot N_i^{MLE} \tag{5.4}$$

5.3.2 Angle Biases and Calibration

The close proximity between the USBL array and the twin inflatable pontoons of the ASV result in local acoustic interference which degrades the accuracy of our angle measurement distributions. The local acoustic interactions with the pontoons manifest as measurement biases, shifting the maxima of the angle distributions away from their true values. These biases are systemic, and as a result, produce an 'offset' between the maximum azimuth in the angle distribution and the true azimuth between the array and the beacon – this offset varies depending on azimuth, as illustrated in blue on the top of Fig. 5.7. The consistency of these azimuth-dependent biases across varying ranges and mission times demonstrate their

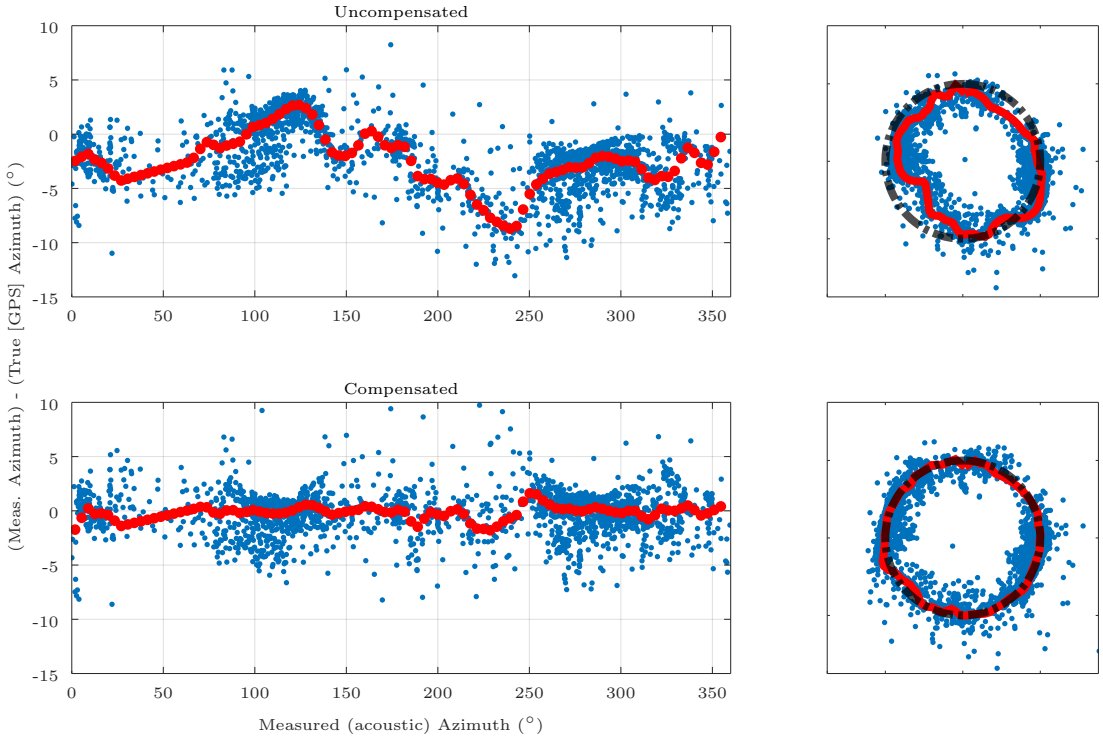

Figure 5.7: Plot of MLE azimuthal values from the collection of angle distributions versus the difference between these values and azimuths from DGPS ground-truth – biases in acoustic azimuth measurements vary with measured azimuth between the piUSBL array and a beacon due to local acoustic interactions with the ASV. *Top*: uncompensated difference in azimuth between acoustic measurements and true (DGPS) azimuth (blue), as well as medians of 100 bins (red). *Bottom*: after compensation by subtracting the median 'offset' of the nearest bin. *Right*: projections of biases onto a circle demonstrate that they are circularly consistent and azimuth-dependent.

systemic nature, and as such, are most likely due to local acoustic effects with the vehicle pontoons.

We use a simple strategy to compensate for these biases: the measured azimuthal biases are divided into 100 equal bins, and the median value of each bin is calculated (shown as red circles in Fig. 5.7); after beamforming, the azimuth corresponding to the largest value of the angle measurement distribution is determined, and the median value for the bin closest in azimuth is subtracted from the measurement. This results in a compensated acoustic azimuth measurement that mitigates these local acoustic effects, as shown on the bottom of Fig. 5.7. This illustrates that after compensating for these biases, the difference between the true azimuth and the azimuth calculated from the maximum of the angle measurement distribution becomes centered around zero, with the median value for the 100 bins becoming almost 'flat'.

5.3.3 Acoustic Position and Outliers

Given acoustic range and angle measurement distributions, we can combine their maximum likelihood estimates (r^{MLE}, θ^{MLE}, ϕ^{MLE}) with ASV heading (ψ^{dgps}) to project an estimate of the position ($x^{vcf}_{beacon}, y^{vcf}_{beacon}$) of the beacon relative to the ASV:

$$\begin{bmatrix} x^{vcf,MLE}_{beacon} \\ y^{vcf,MLE}_{beacon} \end{bmatrix} = r^{MLE} \begin{bmatrix} \sin(\psi^{dgps}) & -\cos(\psi^{dgps}) \\ \cos(\psi^{dgps}) & \sin(\psi^{dgps}) \end{bmatrix} \begin{bmatrix} \sin(\theta^{MLE}) \cdot \cos(\phi^{MLE}) \\ \sin(\theta^{MLE}) \cdot \sin(\phi^{MLE}) \end{bmatrix} \tag{5.5}$$

where we assume that any pitch (β_{asv}) and roll (γ_{asv}) experienced by the ASV is negligible, and thus project the position onto a 2D plane. Given the known position of the beacon in the local-level frame (LLF) ($x^{llf}_{beacon}, y^{llf}_{beacon}$), the MLE of vehicle position in the LLF is:

$$\begin{bmatrix} x^{llf,MLE}_{asv} \\ y^{llf,MLE}_{asv} \end{bmatrix} = \begin{bmatrix} x^{llf}_{beacon} \\ y^{llf}_{beacon} \end{bmatrix} - \begin{bmatrix} x^{vcf,MLE}_{beacon} \\ y^{vcf,MLE}_{beacon} \end{bmatrix} \tag{5.6}$$

where we assume that the depth of the beacon and the depth of the USBL array are equal, and so they lay on the same 2D plane.

The time sequence of positions generated by these MLE values contain outliers that are not temporally consistent. To get a better understanding of the range, angle and positional accuracy of our acoustic measurement distributions, we remove these outliers using a simple temporal filter in which estimates are discarded if they exceed a specified threshold (where $v_{max} := 3$ m/s):

$$\text{KEEP} \quad (x^{llf,MLE}_{asv}, y^{llf,MLE}_{asv}) \quad \text{AND} \quad r^{MLE}, \theta^{MLE}, \phi^{MLE} \quad \text{IFF :}$$
$$\sqrt{(x^{llf,MLE}_{asv}(t) - x^{llf,MLE}_{asv}(t-\Delta t))^2 + (y^{llf,MLE}_{asv}(t) - y^{llf,MLE}_{asv}(t-\Delta t))^2} \leq \Delta t \cdot v_{max} \tag{5.7}$$

These outliers occur when MLE selects a false maximum that appears in the acoustic measurement distributions. Undesirable acoustic effects such as multipath contribute to these false maxima exceeding the value of true maxima. This outlier removal step is justified by the fact that Bayesian filtering will tend to remove these outliers by tracking the 'true' mode over time - thus, MLE values without these outliers represent a truer picture of measurement accuracy.

5.4 Acoustic Measurement Accuracy

5.4.1 Experimental Dataset

A dataset to analyze the accuracy of acoustic measurements calculated from the piUSBL system was gathered using the WAM-V ASV in a field experiment carried out on the Charles River by the MIT Sailing Pavilion in Cambridge, MA in November 2017. In this experiment, the two acoustic beacons were affixed to the Pavilion dock at a depth of approximately 1 m, with the beacons spaced 80 m apart – in our local-level frame (LLF), beacon A was placed at $(40, 0)$ and beacon B at $(-40, 0)$. Beacon A transmitted a $10-12$ kHz, 20 ms LFM up-chirp, and beacon B transmitted a $11-9$ kHz, 20 ms LFM down-chirp, both in sync at a rate of 1 Hz (as seen in Fig. 5.5). The WAM-V ASV was first remotely driven approximately 210m away from the beacons, then driven back toward the beacons in steps of 30m. During each step a manual 360° rotation was performed, allowing us to gather angle measurement distributions from all azimuths. The ASV was then instructed to autonomously perform a lawnmower pattern at a speed of 0.7 $\mathrm{m\,s^{-1}}$, with 9 legs of 150 m length, with a spacing between legs of 20 m, and finally driven back toward the dock remotely. The experiment lasted approximately 2600 s, resulting in the vehicle track seen in black in Fig. 5.8.

5.4.2 Range and Azimuth Error

The processing steps following algorithm 1 in chapter 2, section 2.4, and the outlier removal using Eq. 5.7, produce temporally-consistent MLE values for range and angle between the ASV and beacon A as well as beacon B. These values can be compared to ground-truth range r^{dgps} and azimuth ϕ^{dgps}, calculated using DGPS position (x^{dgps}, y^{dgps}) and heading (ψ^{dgps}):

$$r^{dgps}(t) = \sqrt{(x^{dgps}(t) - x^{llf}_{beacon})^2 + (y^{dgps}(t) - y^{llf}_{beacon})^2} \tag{5.8}$$

$$\phi^{dgps}(t) = (\arctan((y^{dgps}(t) - y^{llf}_{beacon})/(x^{dgps}(t) - x^{llf}_{beacon})) + \psi^{dgps}(t) + \frac{\pi}{2}) \mod 2\pi \tag{5.9}$$

$$\theta^{dgps} \approx \frac{\pi}{2} \qquad \text{SINCE} \qquad z^{llf}_{beacon} \approx z^{llf}_{asv}$$

Comparison plots of range, azimuth and inclination between the MLE values calculated using the piUSBL system and DGPS can be seen in Fig. 5.9, which demonstrate that they are in good agreement. In addition, the estimate of inclination from MLE shows fair stability around 90°, which indicates that the array and boom experience only minor variations in pitch due to non-perfect rigid attachment of the boom to the ASV; however, these variations are a possible source of measurement error in our system which we cannot mitigate without proper sensing of the attitude of the boom and attached receiver.

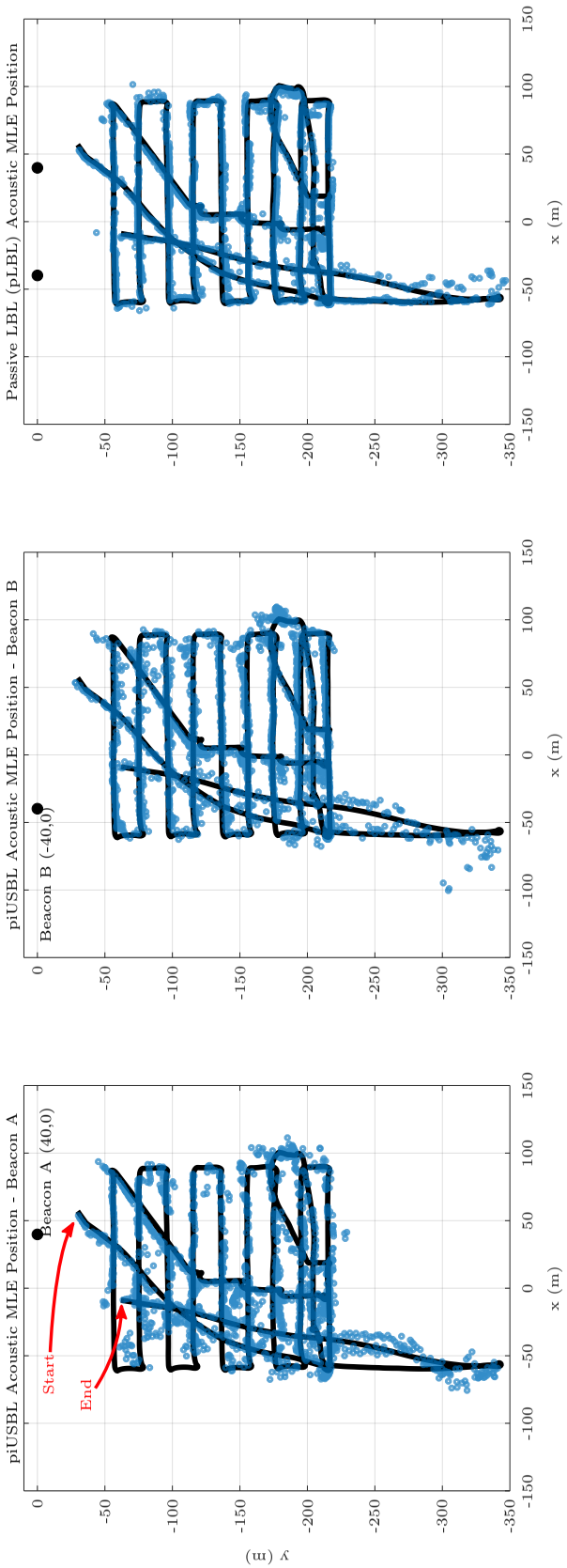

Figure 5.8: Plots of DGPS and MLE trajectories of the WAM-V ASV during an experiment carried out on the Charles River in November 2017 – ground-truth position from the Hemisphere V102 DGPS is shown as a black track, and maximum likelihood estimate (MLE) values of position (after outlier removal) from acoustic range and angle measurement distributions are shown as blue circles. *Left*: MLE positions calculated using range and angle from beacon *A*. *Center*: MLE positions calculated using range and angle from beacon *B*. *Right*: MLE positions calculated using range-only measurements from both beacons.

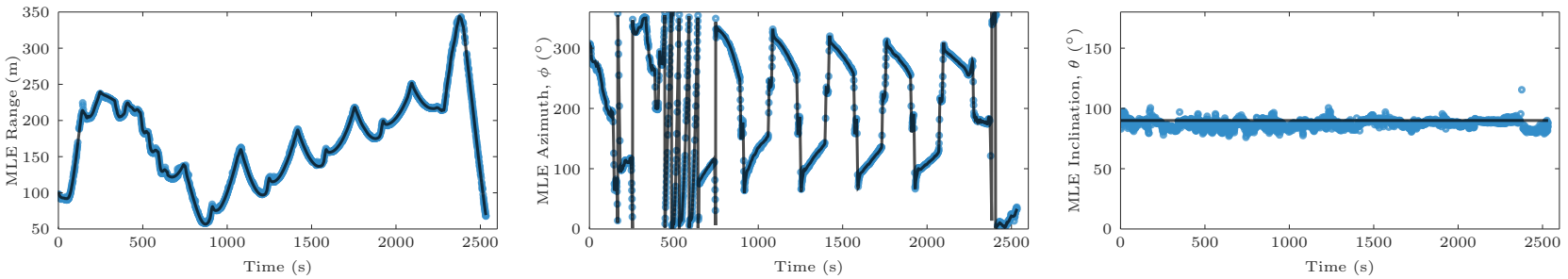

Figure 5.9: MLE values from range and angle measurement distributions between the ASV and beacon B – MLE values are shown as blue circles, while true range and angle values calculated using DGPS using Eqs. 5.8 and 5.9 are shown in black; acoustic MLE values are in very close agreement to the true values. *Left*: range. *Center*: azimuth. *Right*: inclination.

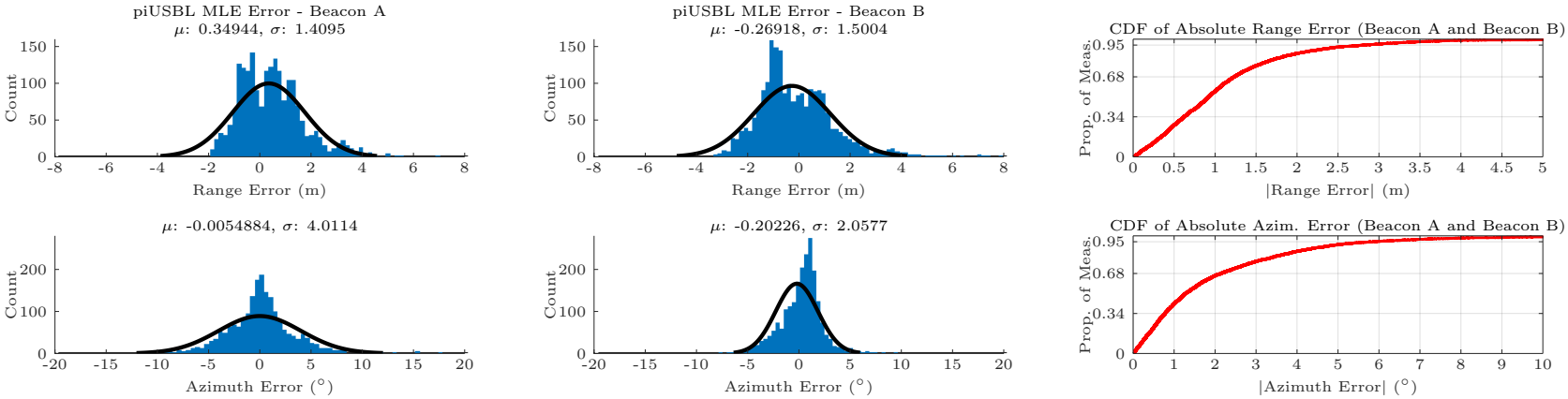

Figure 5.10: Range and azimuth differences in value between MLE from acoustic measurement distributions and from DGPS – range and azimuth error histograms between MLE values from piUSBL acoustic measurement distributions against DGPS are shown along with Gaussian fits with their means and standard deviations. *Left*: error histograms for beacon A. *Center*: error histograms for beacon B. *Right*: empirical cumulative distribution functions (CDFs) for absolute range and azimuth errors using combined data from both beacons.

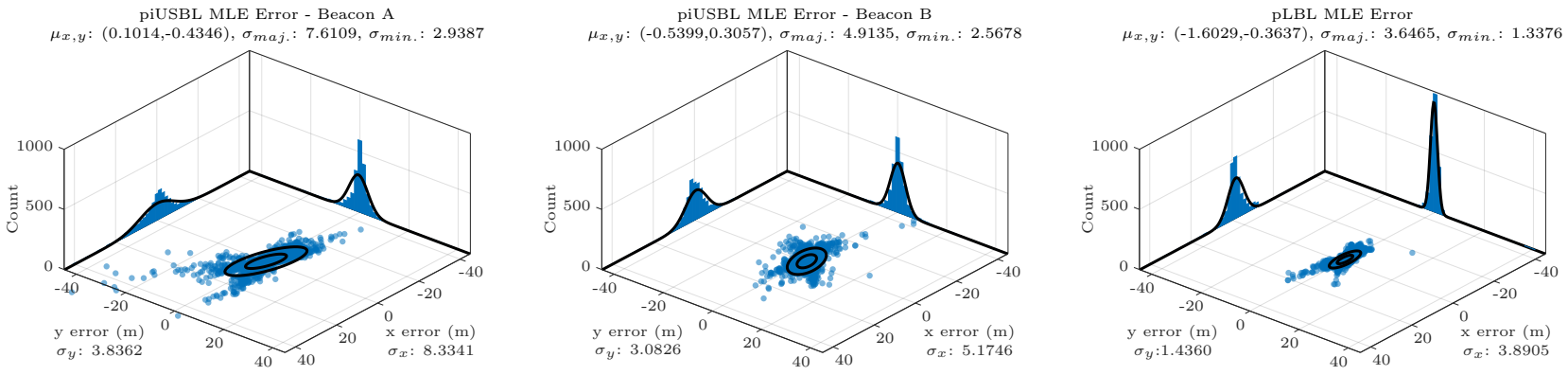

Figure 5.11: Position differences in value between MLE from acoustic measurement distributions and from DGPS – the distributions of position error between MLE values from acoustic measurements distributions (for both piUSBL and passive LBL) against DGPS are shown as blue circles, with 1σ and 2σ ellipses in black; the associated marginal error histograms in the x and y dimensions are projected on the 'walls', with Gaussian fits in black. *Left*: position errors for piUSBL to beacon *A*. *Center*: position errors for piUSBL to beacon *B*. *Right*: position errors for range intersections to both beacons from passive LBL.

	Range Error (m)		**Azimuth Error (°)**		**Position Error (m)**				
	$\boldsymbol{\mu}_r$	$\boldsymbol{\sigma}_r$	$\boldsymbol{\mu}_\phi$	$\boldsymbol{\sigma}_\phi$	$\boldsymbol{\mu}_{x,y}$	$\boldsymbol{\sigma}_{major}$	$\boldsymbol{\sigma}_{minor}$	$\boldsymbol{\sigma}_x$	$\boldsymbol{\sigma}_y$
piUSBL Beacon *A*	0.349	1.409	-0.005	4.011	(0.101, -0.435)	7.611	2.939	8.334	3.836
piUSBL Beacon *B*	-0.269	1.500	-0.202	2.058	(-0.540, 0.306)	4.914	2.568	5.175	3.083
piUSBL All (*A* & *B*)	0.029	1.490	-0.107	3.157					
pLBL *A* & *B* (range-only)					(-1.603, -0.364)	3.647	1.338	3.891	1.436

Table 5.1: Error statistics from MLE using range and angle measurement distributions against DGPS for range, azimuth and position – statistics are shown for both beacons individually and with data combined, as well as for positions from projection of range and angle; combining piUSBL data from both beacons provides statistics only for range and azimuth, since averaging position from the two datasets provides little additional insight; passive LBL from range-only intersections of both beacons provides statistics only for position, since range statistics remain identical to the piUSBL cases.

The difference between MLE values from range and angle distributions and DGPS ground-truth values represent the measurement error in range and azimuth of our piUSBL system, which we can illustrate using histogram plots with Gaussian fits as shown in Fig. 5.10. The leftmost plots illustrate the range and azimuth errors between the array and beacon A, and the center plots illustrate the equivalent errors for beacon B; statistics from these errors are shown in table 5.1. Using data from both beacons, these statistics indicate that the piUSBL system can provide raw measurement accuracies of $\mu \pm \sigma = 0.03 \pm 1.49$ m and $\mu \pm \sigma = -0.11 \pm 3.16°$ in range and azimuth respectively. Note that the error in range of ± 1.49 m corresponds extremely well with the jitter of approximately 1 ms experienced by the piUSBL beacons; as previously mentioned in subsection 2.3.1, a delay in the broadcast of the beacon by 1 ms corresponds to an error in range of about 1.5 m – range accuracy can thus be improved by a reduction of this jitter. The rightmost plots of Fig. 5.10 illustrate the empirical cumulative distribution functions (CDFs) of absolute error in range and azimuth using all acoustic data (from both beacons); these plots indicate that 68% of raw range measurements have an absolute error of less than 1.25 m, and 68% of raw azimuth measurements fall below an absolute error of 2.15°.

5.4.3 Signal to Noise Ratio

Many factors contribute to the level of accuracy of our system, such as the composition of the acoustic environment (including man-made structures, water column depth, riverbed material, sound-speed profile, etc.), the number of elements in the USBL receiver, and the design of the broadcast signal; to better understand the limitations of our piUSBL system within our operational environment, it is beneficial to estimate the signal-to-noise ratio (SNR) from experimental data. As explained in subsection 2.4.1, we demonstrated that the matched filter is the optimal LTI filter for maximizing the SNR in the presence of additive white Gaussian noise – thus, the output of the matched filter that we use for range estimation can be used to directly estimate the SNR [160]:

$$y(t) = 4\Re \int_0^\infty \frac{\boldsymbol{X}(f) S^*(f)}{S_n(f)} e^{2\pi i f t} df \tag{5.10}$$

where $S_n(f)$ is the power spectral density (PSD) of the noise at the output of the filter, and which we estimate with Welch's method using recorded acoustic data when the beacons were not broadcasting as illustrated in Fig. 5.12. To estimate the SNR of our system, we simply take the maximum output of the matched filter given by Eq. 5.10. This estimation indicates that the SNR of our system is approximately 10–25 dB depending on various factors, including range to the beacons, vehicle self-noise[4], and other acoustic effects. The distribution of estimated SNRs for both beacons are plotted as histograms in Fig. 5.13.

[4]Vehicle self-noise contributes to an increased noise floor, and includes motor noise (which varies throughout the mission), vibration, fluid flow noise (which varies with speed), and electronic noise in the piUSBL system.

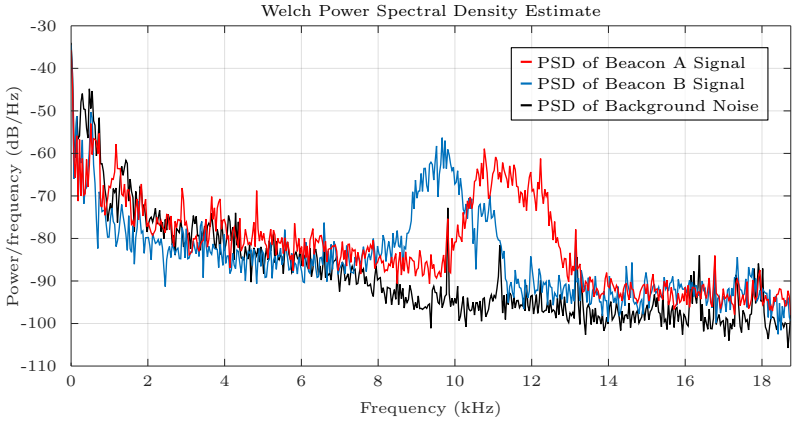

Figure 5.12: Sample plots of PSD estimated using Welch's method for broadcast beacon signals and background noise as measured by the WAM-V piUSBL system – the power spectral density (PSD) for sample acoustic measurements in which only beacon A and only beacon B were broadcasting are shown in red and blue respectively, while the PSD for a measurement containing only background noise is shown in black; in this case the measurements were obtained when the ASV and beacons were in close proximity, giving SNRs of about 25 dB.

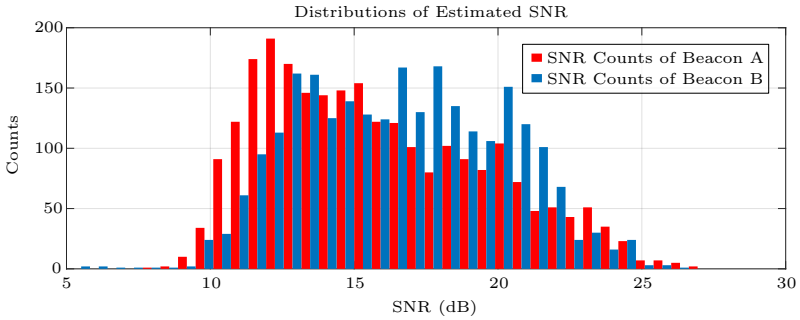

Figure 5.13: Distribution of the estimated SNR for both beacons – the signal-to-noise ratio (SNR) distribution estimated from experimental measurements for beacon A is shown as a red histogram, and the distribution for beacon B as a blue histogram; it is apparent that the majority of SNR values fall within 10–25 dB, meaning the signals broadcast by the beacons are on the order of 10 to 300 times more powerful than the noise.

5.4.4 Range-Only Position from Passive LBL

Given that our experiments make use of two independent piUSBL beacons, we can use the MLE values from the range distributions of both beacons to estimate the position of the ASV using the intersections of range circles in a process called multilateration. This process is commonly used in long baseline (LBL) positioning, but since the piUSBL receiver passively receives the acoustic signals, we term this approach passive LBL. Since we only use two beacons, their range circles produce two intersections – however, since the vehicle only operates in the lower half-plane of the line that connects the two beacons, we can disambiguate the

correct intersection. We estimate range-only passive LBL position by root-finding of the following non-linear equation:

$$(x_{pLBL}^{llf,MLE}, y_{pLBL}^{llf,MLE}) = \min_{x,y} \sum_{i=A,B} (x - x_{beacon,i}^{llf})^2 + (y - y_{beacon,i}^{llf})^2 - (r_i^{MLE})^2 \tag{5.11}$$

This equation is optimized using the Powell hybrid method [188], where we use the ASVs DGPS position as the initial guess for the solver. Position estimates from passive LBL are shown on the right of Fig. 5.8.

5.4.5 Position Error

We have shown the error statistics gathered for range and azimuth from piUSBL in Fig. 5.10, and listed them on the left in table 5.1; In addition to these statistics, error statistics for position can also be obtained. These error distributions are illustrated in Fig. 5.11, which illustrate the 2D positional error in x and y for piUSBL to beacon A on the left, piUSBL to beacon B in the center, and range-only passive LBL on the right; the marginal error distributions are also shown as histograms projected onto the 'walls' of these plots. Gaussian fits to these distributions provide us with the mean and standard deviation values for positional error, which we list on the right in table 5.1; the standard deviations of the major and minor axes of the 2D Gaussian fits are listed (σ_{major} and σ_{minor}), as well as those of the marginal distributions (σ_x and σ_y).

The values from table 5.1 and the distributions illustrated in Fig. 5.11 indicate that raw piUSBL measurements calculated by projecting MLE range and azimuth values can provide an accurate estimate of position. For beacon A, the error in position at these ranges is $\mu \pm \sigma = (0.10, -0.44) \pm (8.33, 3.84)$ m in x and y, and for beacon B, the corresponding error in position is $\mu \pm \sigma = (-0.54, -0.31) \pm (5.18, 3.08)$ m. It is interesting to note that, as expected, the distributions for piUSBL positional error have their major axis of 'spread' in the direction perpendicular to the angle between the respective beacon and the majority of position measurements – this is entirely due to the fact that there is a larger uncertainty in the angle measurement than in the range measurement, and is made obvious by the rotation of the two distributions. As such, for piUSBL the error in position is less informative than the errors in range and angle measurements, since positional error will tend to increase with range from the beacon due to the uncertainty in angle; at these ranges however (< 350 m), the positional accuracy of piUSBL is on par to that of passive LBL.

The values indicated in table 5.1 demonstrate that passive LBL provides highly accurate estimates of vehicle position, similar in accuracy to that of conventional GPS. The error in position from passive LBL in x and y is about $\mu \pm \sigma = (-1.60, -0.36) \pm (3.89, 1.44)$ m.

The larger spread in the x axis is expected, since the majority of the position measurements occur 'below' the two acoustic beacons rather than 'in-between' them – as such, any error in one range will cause the circular intersection to slide along the other range circle, causing a larger deviation in the x direction than the y direction. As expected, passive LBL provides both an accurate and highly precise estimate of position, since it does not make use of angle measurements which have a higher variability than range - precision for passive LBL should therefore not degrade as badly with range as that for piUSBL.

The statistics for passive LBL provide us with an additional comfort – the level of accuracy demonstrated by passive LBL positioning ensures that it can be used with confidence as a measure of ground-truth for piUSBL positioning. In later chapters of this thesis, in which multiple autonomous underwater vehicles are deployed using piUSBL, ground-truth position from passive LBL can be viewed with some assurance, on a level similar to that of GPS ground-truth – and this can be done simply by the addition of extra piUSBL beacons for passive LBL measurements within our existing acoustic system.

5.5 Bayesian Filtering with the Sequential Monte-Carlo Beamformer

As is standard in our piUSBL processing pipeline, Bayesian filtering is used to improve localization by incorporating our acoustic measurement distributions with inertial measurements. As part of our evaluation of the capabilities of piUSBL positioning, we are interested not only in the achievable improvement in localization afforded by Bayesian filtering, but also in characterizing how well the filtered piUSBL system performs with increasing degradation in the accuracy of inertial measurements. By doing so, we can determine how robust the system is to bias errors in both vehicle heading and speed.

To perform Bayesian filtering, we make use of the sequential Monte-Carlo beamformer (SMCB), described in chapter 3 – this approach essentially closely-couples the particle filter introduced in chapter 2 subsection 2.5.2 and beamforming. To briefly recap, this filter tracks the position of the beacon relative to the vehicle; the predict step propagates particles using vehicle speed and heading; these particles are duplicated into two additional sets of particles: one set of particles is transformed into the range-domain by calculating their magnitude, and their weights updated using the range measurement distribution – the other set of particles is transformed into the body-fixed frame (BFF) by first calculating their range-normalized vectors and then transforming these vectors into spherical (θ, ϕ) coordinates, where the beamformed output power at these coordinates are used to update their weights; finally, these two sets of particles are transformed back into the vehicle-carried frame (VCF) by element-wise multiplication. Refer to subsection 2.5.2 for an in-depth explanation of the reference frames

and operating principles of the particle filter, and to chapter 3 for an explanation of the SMCB.

Since we wish to use this filter not only for piUSBL localization to a single beacon, but for piUSBL and range-only passive LBL localization using both beacons, we make a few minor changes to our state representation. The modified state vector is given by:

$$\boldsymbol{x}(t) = \begin{bmatrix} x^{llf}_{asv}(t) \\ y^{llf}_{asv}(t) \\ z^{llf}_{asv} \\ x^{vcf}_{beacon,A}(t) \\ y^{vcf}_{beacon,A}(t) \\ x^{vcf}_{beacon,B}(t) \\ y^{vcf}_{beacon,B}(t) \\ x^{llf}_{beacon,A} \\ y^{llf}_{beacon,A} \\ x^{llf}_{beacon,B} \\ y^{llf}_{beacon,B} \\ \alpha_{asv}(t) \\ v_{asv}(t) \\ \psi_{asv}(t) \end{bmatrix} \text{ WHERE } \begin{bmatrix} x^{llf}_{asv}(t) & : & \text{x–pos}^{\text{n}} \text{ of ASV in the local-level frame} \\ y^{llf}_{asv}(t) & : & \text{y–pos}^{\text{n}} \text{ of ASV in the local-level frame} \\ z^{llf}_{asv} & : & \text{z–pos}^{\text{n}} \text{ of ASV in the local-level frame} \\ x^{vcf}_{beacon,A}(t) & : & \text{x–pos}^{\text{n}} \text{ of beacon } A \text{ relative to the ASV} \\ y^{vcf}_{beacon,A}(t) & : & \text{y–pos}^{\text{n}} \text{ of beacon } A \text{ relative to the ASV} \\ x^{vcf}_{beacon,B}(t) & : & \text{x–pos}^{\text{n}} \text{ of beacon } B \text{ relative to the ASV} \\ y^{vcf}_{beacon,B}(t) & : & \text{y–pos}^{\text{n}} \text{ of beacon } B \text{ relative to the ASV} \\ x^{llf}_{beacon,A} & : & \text{x–pos}^{\text{n}} \text{ of beacon } A \text{ in the local-level frame} \\ y^{llf}_{beacon,A} & : & \text{y–pos}^{\text{n}} \text{ of beacon } A \text{ in the local-level frame} \\ x^{llf}_{beacon,B} & : & \text{x–pos}^{\text{n}} \text{ of beacon } B \text{ in the local-level frame} \\ y^{llf}_{beacon,B} & : & \text{y–pos}^{\text{n}} \text{ of beacon } B \text{ in the local-level frame} \\ \alpha_{asv}(t) & : & \text{course of ASV} \\ v_{asv}(t) & : & \text{speed of ASV} \\ \psi_{asv}(t) & : & \text{yaw of ASV, rot}^{\text{n}} \text{ around body–fixed z–axis} \end{bmatrix} \tag{5.12}$$

where the first two elements are tracked using the filter. Unlike the original state representation, this single state vector maintains estimates of the relative positions of both beacons, and assumes that vehicle pitch and roll are zero. Most importantly, particles are propagated in the local-level frame (LLF) common to both beacons, giving the following predict step:

$$\boldsymbol{x}(t) = \begin{bmatrix} x^{llf}_{i,asv}(t) \\ y^{llf}_{i,asv}(t) \\ \vdots \\ x^{vcf}_{i,beacon,A}(t) \\ y^{vcf}_{i,beacon,A}(t) \\ x^{vcf}_{i,beacon,B}(t) \\ y^{vcf}_{i,beacon,B}(t) \\ \vdots \\ \alpha_{i,asv}(t) \\ v_{i,asv}(t) \\ \psi_{i,asv}(t) \end{bmatrix} = \begin{bmatrix} x^{llf}_{i,asv}(t-\Delta t) + v_{i,asv}(t-\Delta t) \cdot \Delta t \cdot \cos(\alpha_{i,asv}(t-\Delta t)) \\ y^{llf}_{i,asv}(t-\Delta t) + v_{i,asv}(t-\Delta t) \cdot \Delta t \cdot \sin(\alpha_{i,asv}(t-\Delta t)) \\ \vdots \\ x^{llf}_{i,beacon,A} - x^{llf}_{i,asv}(t) \\ y^{llf}_{i,beacon,A} - y^{llf}_{i,asv}(t) \\ x^{llf}_{i,beacon,B} - x^{llf}_{i,asv}(t) \\ y^{llf}_{i,beacon,B} - y^{llf}_{i,asv}(t) \\ \vdots \\ \arctan(\Delta y^{dgps}(t)/\Delta x^{dgps}(t)) + \epsilon_\alpha(t) + \mathcal{N}(0,\sigma^2_\alpha) \\ \sqrt{(\Delta x^{dgps}(t))^2 + (\Delta y^{dgps}(t))^2}/\Delta t + \epsilon_v(t) + \mathcal{N}(0,\sigma^2_v) \\ \psi^{dgps}(t) + \epsilon_\alpha(t) + \mathcal{N}(0,\sigma^2_\alpha) \end{bmatrix} \tag{5.13}$$

WHERE

$$\Delta x^{dgps}(t) = x^{dgps}(t) - x^{dgps}(t - \Delta t) \tag{5.14}$$

$$\Delta y^{dgps}(t) = y^{dgps}(t) - y^{dgps}(t - \Delta t) \tag{5.15}$$

By propagating the particles in the LLF rather than in the VCF as is done in our standard particle filter, we are able to incorporate acoustic range and angle measurement distributions from both beacon *A* and beacon *B*. This is performed in a serial manner – the weight of each particle is first updated using the acoustic measurements of one beacon, followed by those of the other beacon. This approach also enables the filter to incorporate only the range measurement distributions of both beacons by effectively ignoring the angle measurement distributions, allowing us to obtain filtered range-only passive LBL.

Notice that our state vector includes true velocity v_{asv} and course α_{asv} extracted at every timestep using DGPS position measurements:

$$\alpha_{asv}(t) = \arctan(\Delta y^{dgps}(t)/\Delta x^{dgps}(t)) + \epsilon_\alpha(t) \tag{5.16}$$

$$v_{asv}(t) = \sqrt{(\Delta x^{dgps}(t))^2 + (\Delta y^{dgps}(t))^2}/\Delta t + \epsilon_v(t) \tag{5.17}$$

$$\psi_{asv}(t) = \psi^{dgps}(t) + \epsilon_\alpha(t) \tag{5.18}$$

where ψ_{asv} is the vehicle heading as measured by the Vector V102 DGPS receiver. Velocity and course are extracted from DGPS position in order to simulate inertial measurements – doing so allows us to 'perfectly' recreate the DGPS trajectory, if the bias terms $\epsilon_\alpha(t), \epsilon_v(t)$ are set to zero. Particles are propagated during the predict step using simulated velocity and course, while heading is used to transform the particles into the BFF to incorporate angle measurements distributions; this is necessary because the vehicle 'crabs' due to the effects of wind on the surface, meaning that vehicle course and heading deviate from each other. The bias terms $\epsilon_\alpha(t), \epsilon_v(t)$ allow us to inject noise into the simulated inertial measurements, enabling us to examine how robust the piUSBL system is to inertial sensor offset and noise. These bias terms vary as Gaussian random walks to simulate inertial sensor noise:

$$\epsilon_\alpha(t) = b_\alpha + \mathcal{N}(0, \sigma^2_{\epsilon_\alpha})\Delta t \ : \ b_\alpha \in \{0.0, 0.3, 1.0, 3.0, 6.0\} \tag{5.19}$$

$$\epsilon_v(t) = b_v + \mathcal{N}(0, \sigma^2_{\epsilon_v})\Delta t \ : \ b_v \in \{0.0, 0.3, 1.0, 3.0, 6.0\} \times 10^{-1} \tag{5.20}$$

where we set $\sigma_{\epsilon_v} := 1 \text{ cm s}^{-1}$ and $\sigma_{\epsilon_\alpha} := 6^\circ \text{ h}^{-1}$, which are typical values for the speed precision of a standard DVL and bias instability for a MEMS IMU.

The filter update step follows the standard approach that we outlined in subsection 2.5.2. The only difference is that the duplication of the particles into the range and angle particle sets ($\overline{\overline{\text{rng}}}\boldsymbol{S}$ and $\overline{\overline{\text{agl}}}\boldsymbol{S}$) is performed twice – once for each beacon, using the state elements corresponding to the relative position of each beacon to the vehicle, $(x^{vcf}_{i,beacon,A}, y^{vcf}_{i,beacon,A})$ and $(x^{vcf}_{i,beacon,B}, y^{vcf}_{i,beacon,B})$. The serialized ordering of which beacon's measurements are incorporated first has a negligible effect on the localization output, due to the stochastic nature of particle filtering, and is increasingly negligible as the number of particles increases.

Finally, initialization of the filter is performed using DGPS position at the very first timestep:

$$\boldsymbol{x}(0) = \begin{bmatrix} x^{llf}_{asv}(0) \\ y^{llf}_{asv}(0) \\ z^{llf}_{asv} \\ \vdots \\ x^{llf}_{beacon,A} \\ y^{llf}_{beacon,A} \\ x^{llf}_{beacon,B} \\ y^{llf}_{beacon,B} \\ \vdots \end{bmatrix} = \begin{bmatrix} x^{dgps}(0) + \mathcal{N}(0, \sigma^2_{dgps}) \\ y^{dgps}(0) + \mathcal{N}(0, \sigma^2_{dgps}) \\ 0 + \mathcal{N}(0, \sigma^2_{dgps}) \\ \vdots \\ 40 \\ 0 \\ -40 \\ 0 \\ \vdots \end{bmatrix} \tag{5.21}$$

Illustrative examples of the filtered localization output from the incorporating of acoustic measurement distributions and simulated inertial measurements are shown in Fig. 5.14.

5.6 Filtered Localization Accuracy

In order to investigate the impact of Bayesian filtering on the localization accuracy of piUSBL, we ran several realizations of simulated ASV inertial measurements with increasing levels of bias in speed and course/heading, while fusing different sets of acoustic measurements. Three classes of acoustic measurements were used: in the first, only range and angle measurement distributions to a single beacon (beacon B) were fused with inertial measurements – this represents the accuracy and robustness of our standard, single-beacon piUSBL system stack; in the second, range and angle measurement distributions from *both* beacons (beacon A and beacon B) were incorporated – this represents the gain in accuracy and robustness from the addition of a second beacon to our piUSBL system; and in the third, *only range* measurements from both beacons were used – this represents the accuracy and robustness of a filtered LBL system. The initial bias terms were set with increasing values such that $(b_v, b_\alpha) := \{(0.00, 0.0), (0.03, 0.3), (0.1, 1.0), (0.3, 3.0), (0.6, 6.0)\}$; for each pair of bias terms, we ran 10 realizations for each class of acoustic measurements, using 2500 particles in the filter.

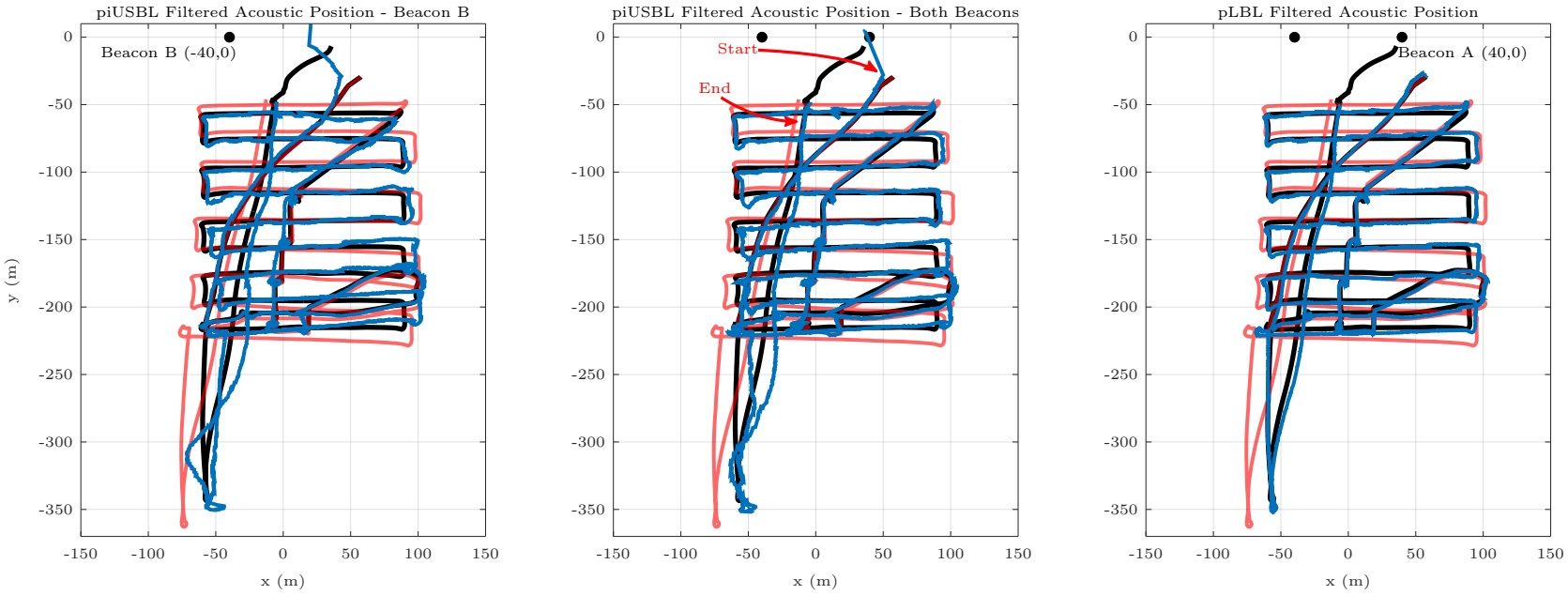

Figure 5.14: Plots of filtered trajectories of the WAM-V ASV from fusion of simulated inertial measurements and various sets of acoustic measurement distributions using a modified sequential Monte-Carlo beamformer (SMCB) – initial bias values for speed and course/heading were set to $b_v := 0.03$ and $b_\alpha := 0.3$ for these realizations; ground-truth DGPS is in black, simulated inertial dead-reckoning in red, and filtered position in blue. *Left*: filtered piUSBL using range and angle measurement distributions to a single beacon *B*. *Center*: filtered piUSBL using range and angle measurement distributions to both beacons. *Right*: filtered passive LBL using only range measurement distributions to both beacons.

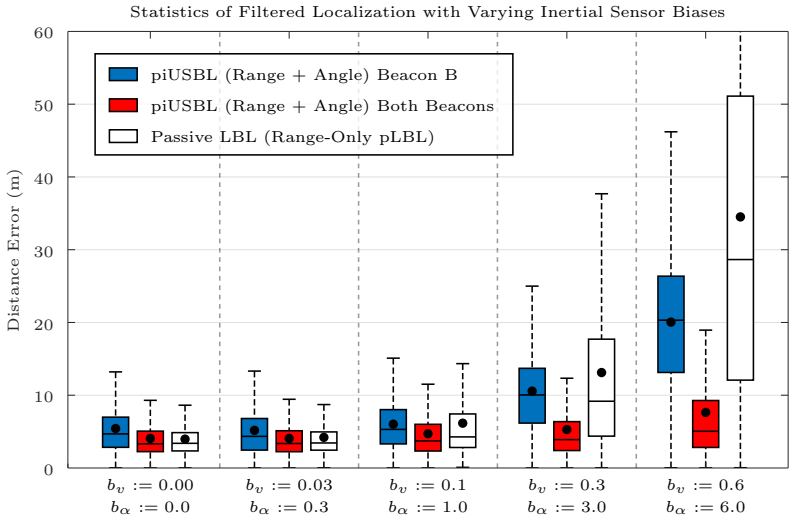

Figure 5.15: Error statistics of 2-norm distance between DGPS and filtered localization with increasing bias error in simulated inertial measurements – solid lines indicate median values and dots indicate mean values; filtered piUSBL using range and angle measurements to beacon B and to both beacons are shown as blue and red boxplots respectively, while passive LBL using range-only measurements to both beacons are shown as white boxplots; the initial speed bias term, b_v, is increased from 0.0 to 0.6, and the initial course/heading bias term, b_α, is increased from 0.0 to 6.0; statistics were gathered over 10 realizations of simulated inertial measurements for each pair of initial bias term.

	piUSBL B			**piUSBL Both**			**pLBL range-only**		
(b_v, b_α)	μ	$\tilde{\mu}$	P_{75}	μ	$\tilde{\mu}$	P_{75}	μ	$\tilde{\mu}$	P_{75}
(0.00, 0.0)	5.42	4.68	6.99	4.07	3.31	5.07	3.97	3.39	4.86
(0.03, 0.3)	5.18	4.35	6.80	4.07	3.36	5.12	4.21	3.44	4.96
(0.1, 1.0)	6.02	5.30	8.02	4.71	3.72	6.00	6.16	4.27	7.43
(0.3, 3.0)	10.6	10.0	13.7	5.28	3.90	6.37	13.1	9.17	17.7
(0.6, 6.0)	20.0	20.3	26.4	7.64	5.05	9.28	34.5	28.6	51.1

Table 5.2: Error statistics corresponding to Fig. 5.15 of 2-norm distance in meters between DGPS and filtered localization with increasing bias error in simulated inertial measurements – μ indicates the mean distance error, $\tilde{\mu}$ indicates the median distance error, and P_{75} indicates the 75^{th} percentile distance error.

5.6.1 Localization Error

The localization error in terms of the 2-norm distance in meters between ground-truth DGPS and the solutions from Bayesian filtering are illustrated with boxplots in Fig. 5.15. The corresponding mean, median, and 75^{th} percentile errors are enumerated in table 5.2. As expected, as the initial bias values in speed and course/heading are increased, inertial measurements become increasingly inaccurate, and so both accuracy and precision of the filtered localization solution degrades regardless of measurement class (single-beacon piUSBL with beacon B,

dual beacon piUSBL with both beacons, or range-only passive LBL). Interestingly, although passive LBL is highly accurate and superior in terms of precision at low levels of inertial measurement bias, its accuracy and precision degrades the fastest with increasing bias error in the simulated inertial measurements; we suspect that this is due to the serialized nature of incorporating acoustic measurements - the particles in the filter may become overconfident and cluster around the maximum range of one beacon without 'correcting' themselves sufficiently using the range measurement of the second beacon. We see also that the accuracy and precision of piUSBL using acoustic range and angle measurement distributions to both beacons is highly robust to noise and bias in the inertial measurements; we suspect that this is due to the fact that heading bias affects the angle measurement of both beacons equally - thus the error due to bias in heading tends to 'average' out when using multiple beacons, a distinct advantage of multi-beacon piUSBL as compared to single-beacon piUSBL. However, single-beacon piUSBL can be seen to be quite robust, and is fairly accurate and precise if the bias in inertial measurements is well calibrated.

5.6.2 Acoustic Heading Estimation

The use of multiple acoustic beacons within the piUSBL framework enables an interesting additional feature – we can now estimate vehicle heading directly using acoustic range and angle measurements. By examining Eq. 5.9, notice that it can be rearranged for heading ψ:

$$\psi(t) = (\phi(t) - \arctan((y(t) - y^{llf}_{beacon})/(x(t) - x^{llf}_{beacon})) - \frac{\pi}{2}) \mod 2\pi \tag{5.22}$$

Thus, given an estimate of vehicle position (x, y), and an estimate of the acoustic azimuth from an angle measurement distribution ϕ, we can directly calculate an estimate of vehicle heading ψ. Note that given vehicle position from DGPS, the error statistics of vehicle heading from Eq. 5.22 for beacon A and beacon B are identical (though mirrored) to those of azimuth shown in Fig. 5.10, if we use MLE to estimate the acoustic azimuth.

To demonstrate the utility of heading estimation from acoustics, we extend our Bayesian filter to incorporate parallel particle filters for tracking the acoustic azimuth of each beacon. For each beacon, a set of particles is maintained that reside in the domain in which beamforming occurs, and the particles are propagated at every timestep using the *change* in heading from inertial measurements (these particles only vary in azimuth, since we assume that the ASV does not experience any pitch or roll). The weights of these particles are updated using the output of beamforming, and the spherical mean is used as the azimuth estimate. Eq. 5.22 is then used, along with the (x, y) estimate from range-only passive LBL, to estimate heading. This hybrid system that combines passive LBL and piUSBL can thus be used to estimate both vehicle position and heading, entirely using acoustic measurements.

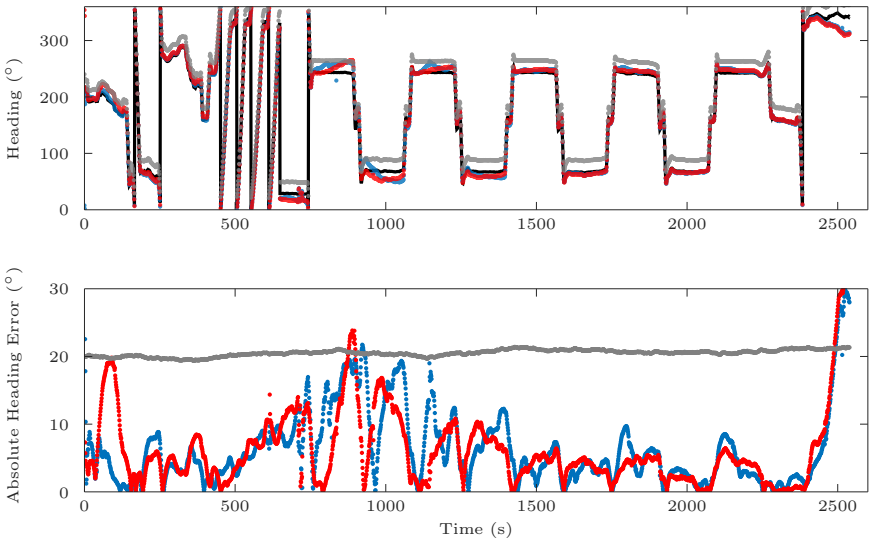

Figure 5.16: Plots of ASV heading estimation using multi-beacon piUSBL – *Top*: ASV heading from DGPS in black, as estimated from beacon A acoustic measurements in blue, as estimated from beacon B acoustic measurements in red, and from noisy simulated inertial measurements in gray with an initial bias of $b_\alpha := 20°$. *Bottom*: Error in ASV heading from acoustic measurements using beacons A and B in blue and red respectively, and from noisy simulated inertial measurements in gray.

We simulate a realization of noisy odometry with this hybrid acoustic system, using initial bias values in speed and course/heading of $b_v := 0\ \mathrm{m\,s^{-1}}$ and $b_\alpha := 20°$ respectively, resulting in the heading estimates and errors shown in Fig. 5.16. These plots illustrate that estimating heading from acoustics is especially useful when large biases are present in a system's inertial heading measurement, which can frequently occur on underwater vehicles that use low-cost inertial measurement units (IMUs) with magnetometers that are especially vulnerable to magnetic anomalies. In contrast to a magnetic compass and IMU, this hybrid acoustic system provides a drift-free and fairly accurate estimate of heading, as long as it is able to determine a good estimate of position. Such a system may be particularly useful for low-cost underwater vehicles that operate in the vicinity of underwater infrastructure that possess magnetic field effects, such as metallic ship hulls and pilings.

5.7 Summary of piUSBL Accuracy using the WAM-V ASV

By deploying our final piUSBL system on the WAM-V autonomous surface vehicle, we were able to gather comprehensive statistics on the accuracy of piUSBL positioning, both in terms of maximum likelihood estimates from acoustic range and angle measurement distributions, as well as in terms of filtered localization through the fusion of these acoustic distributions with

simulated inertial measurements. MLE values from our acoustic distributions demonstrate accuracies in range and azimuth MLE values of about $\mu \pm \sigma = 0.03 \pm 1.49$ m and $\mu \pm \sigma = -0.11 \pm 3.16°$, with 68% of range measurements falling below an absolute error of 1.25 m and 68% of azimuth measurements falling below an absolute error of 2.15°. At ranges less than 350 m, these range and angle values correspond to a precision in position characterized by a standard deviation in x of 5.2–8.3 m, and a standard deviation in y of 3.1–3.8 m.

Bayesian filtering of acoustic and inertial measurements via the sequential Monte-Carlo beamformer (SMCB) allows us to generate temporally-consistent vehicle trajectories in a principled manner. If inertial measurements are highly accurate, we demonstrated that they can also improve positioning accuracy of single-beacon piUSBL, with the filtered position of the vehicle less than 7 m from ground-truth 75% of the time for low levels of inertial bias error. It is difficult to provide a direct comparison of the accuracy of raw MLE position to filtered position, since this comparison is dependent on a number of factors, including noise and bias in inertial measurements, range to the beacon, as well as the weighted balance between 'trusting' inertial measurements versus acoustic measurements; instead we note the important results of this evaluation and characterization of the piUSBL system:

- The most important statistical information that has emerged from this evaluation is the characterization of the MLE accuracy of acoustic range and angle measurement distributions. These values inform positional accuracy from the use of only acoustic range and angle, regardless of the range of operation. Temporal filtering of MLE outliers of the acoustic distributions provide us with a good idea of the positional accuracy achievable using a well-tuned Bayesian filter.

- Through Bayesian filtering, we can produce temporally-consistent trajectories for various moderate levels of bias error in inertial measurements. piUSBL localization using a single beacon is particularly sensitive to bias in heading, since heading is needed to project range and angle to determine the relative position of the beacon in the local-level frame; results demonstrate that its performance remains fairly good even with a bias in heading of up to 3°, but accurate calibration of vehicle heading is essential to produce good positioning results using single-beacon piUSBL.

- As the range of operation increases, Bayesian filtering becomes an increasingly critical component of the piUSBL system by constraining acoustic outliers caused by error in the angle measurement distribution.

- The addition of a second piUSBL beacon provides a significant improvement in accuracy, precision and robustness in vehicle localization. Results indicate that multi-beacon piUSBL is able to provide accurate positioning even at high levels of bias error in inertial measurements, and is more robust than LBL to these types of error.

- Multi-beacon piUSBL enables a novel hybrid system that combines passive LBL and piUSBL, which allows us to estimate both vehicle position and heading entirely using acoustics. Such an approach may be useful for magnetometer-equipped underwater vehicles, whose heading estimate may become erroneous in the presence of magnetic anomalies.

5.8 The Commercial Bluefin SandShark AUV

The implementation of our final piUSBL system on the WAM-V ASV has provided us with a characterization of its achievable positioning accuracy. Now that all the necessary components have been laid out in detail, including the hardware implementation on the prototype SandShark AUV in chapter 2, the filtering stack that enables closed-loop navigation in chapter 3, the efficient beamforming approach that improves acoustic measurement precision in chapter 4, and the characterization of piUSBL accuracy using a five-element pyramidal array in this chapter 5, we describe here the implementation of the system on a fleet of three *commercial Bluefin SandShark AUVs*[5] which we named *Platypus*, *Quokka*, and *Wombat*, and which we used to perform novel multi-AUV experiments detailed in the following chapter.

5.8.1 Hardware Configuration

Like its prototype predecessor, the commercial Bluefin SandShark [189] is a low-cost, miniature AUV, and is in many respects identical to the prototype – again, the rear $\frac{1}{2}$ of the vehicle, the tail, is the standardized SandShark platform provided by Bluefin; the front $\frac{1}{2}$ of the vehicle contains our final piUSBL system. We describe the platform here, pictured in Fig. 5.17, paying particular attention to the key differences between this commercial vehicle and the prototype SandShark.

Platform

A number of changes were made to the commercial incarnation of the SandShark AUV. Like the prototype, the tail section is equipped with the same sensors, as labeled at the top of Fig. 5.17: (i) it is propelled using a single magnetically-coupled thruster, which provides a measure of vehicle speed via a mapping of rotations-per-minute (RPM); (ii) instead of two stepper motors for rudder and elevator control, it makes use of *three* stepper motors to individually actuate three control fins laid out in a triangle configuration – this design decision was made to enable active roll control, in addition to pitch and heading control [189]; (iii) its mast contains GPS and WiFi receivers for global positioning and communications when the

[5] https://gdmissionsystems.com/en/products/underwater-vehicles/bluefin-sandshark-autonomous-underwater-vehicle/

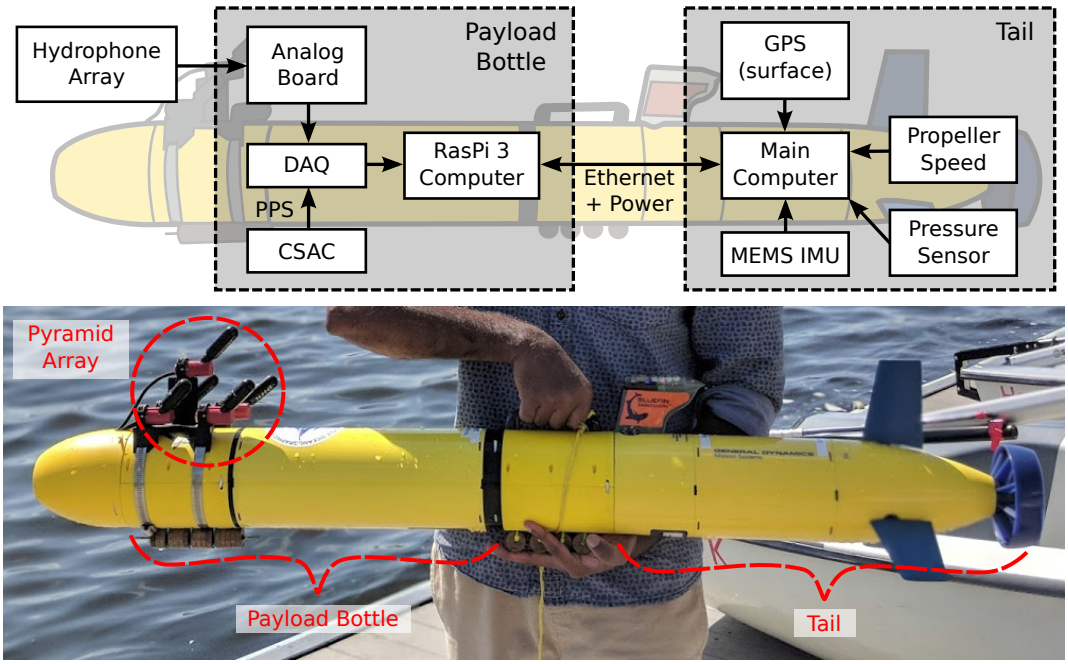

Figure 5.17: Commercial Bluefin SandShark AUV (*Quokka*) outfitted with our final piUSBL payload – the rear $\frac{1}{2}$ of the AUV is the standard commercial SandShark platform (the tail), which consists of a single magnetically-coupled propeller and motor, three stepper motors for fin control, a pressure sensor for depth, a GPS and WiFi receiver, and a 9-axis MEMS IMU with magnetometer; the front $\frac{1}{2}$ of the vehicle is our final piUSBL payload, with system components labeled in the upper diagram.

vehicle is surfaced; (iv) it makes use of an improved 9–axis MEMS IMU with magnetometer from Sparton[6] as an attitude and heading reference system (AHRS) for attitude and heading estimation; (v) it uses a pressure sensor to estimate depth; (vi) finally, a Linux main vehicle computer is used to receive sensor data and send control commands. Note that the tail is also equipped with an Imagenex Model 852 ultra-miniature echo sounder[7] as standard to estimate vehicle altitude above the seafloor, but in our fleet of three SandShark AUVs, only one vehicle had an operational echo sounder. As with the prototype SandShark, a pitch-compensated linear mapping between motor RPM and speed-over-ground is used to estimate AUV velocity:

$$v_{sog} = RPM \cdot 1.25 \times 10^{-3} \cdot \cos(\beta) \tag{5.23}$$

Again, this mapping was experimentally obtained by Bluefin, and is a significant source of localization error for dead-reckoning, which increases on the the order of 3 $\mathrm{m\,min^{-1}}$. Unlike the prototype vehicle, in which it was necessary to include a battery and power management

[6] https://www.spartonnavex.com/
[7] https://imagenex.com/

circuitry in the payload, the commercial vehicle's battery has enough capacity to reliably power the payload via an underwater SubConn cable[8].

As with the prototype vehicle, we make use of the *frontseat–backseat* paradigm, which cedes all low-level vehicle control to the main vehicle computer in the tail section. The platform makes use of proportional-integral-derivative (PID) controllers to reach desired set-points in depth, heading, and roll. Control in the vertical plane is achieved using a nested PID controller in depth and pitch, while separate PID controllers are used to achieved desired heading and stable roll:

- A PID controller for pitch takes as input a desired pitch set-point, and the current pitch and pitch rate from the AHRS, and outputs a pitch moment command to modify the current pitch.
- The desired pitch set-point is output by a PID controller for depth, which takes as input a desired depth set-point from the user, as well as the current depth and depth rate from the pressure sensor, and outputs the desired pitch set-point for consumption by the pitch controller.
- Control in the horizontal plane is achieved using a separate heading PID controller; this controller takes as input a desired heading from the user, as well as the current heading and heading rate from the AHRS, and outputs a heading moment.
- A separate PID controller is used by the vehicle to always attempt to maintain zero roll; the current roll and roll rate from the AHRS are used to output a roll moment to try and counteract any roll experienced by the AUV.

Pitch, heading and roll moments output by these PID controllers are mixed to command the angles on the three control fins. Because the behavior of the vehicle is highly sensitive to the payload and its mass characteristics, PID tuning was done by performing a number of experimental deployments of the vehicle with the final piUSBL payload attached. The resulting PID gains for reasonable vehicle control performance are listed in table 5.3.

	Pitch Cont.	**Roll Cont.**	**Heading Cont.**	**Depth Cont.**
P Gain	$0.3\ \mathrm{N\,m\,rad^{-1}}$	$0.3\ \mathrm{N\,m\,rad^{-1}}$	$0.2\ \mathrm{N\,m\,rad^{-1}}$	$0.1\ \mathrm{rad\,m^{-1}}$
I Gain	$0.15\ \mathrm{N\,m\,rad^{-1}\,s^{-1}}$	$0.15\ \mathrm{N\,m\,rad^{-1}\,s^{-1}}$	$0.1\ \mathrm{N\,m\,rad^{-1}\,s^{-1}}$	$0.01\ \mathrm{rad\,m^{-1}\,s^{-1}}$
D Gain	$0.0\ \mathrm{N\,m\,rad^{-1}\,s}$	$0.0\ \mathrm{N\,m\,rad^{-1}\,s}$	$0.01\ \mathrm{N\,m\,rad^{-1}\,s}$	$0.0\ \mathrm{rad\,m^{-1}\,s}$
I Max	$0.15\ \mathrm{N\,m}$	$0.15\ \mathrm{N\,m}$	$0.15\ \mathrm{N\,m}$	$5°$

Table 5.3: The proportional-integral-derivative (PID) gains for the commercial SandShark AUVs outfitted with our final piUSBL payload.

[8] https://www.macartney.com/what-we-offer/systems-and-products/connectors/subconn/

One significant difference from the prototype vehicle is that the software on the commercial SandShark AUV uses a proprietary Bluefin system, rather than ROS; a critical outcome of this change is that we are no longer able to transmit positional information from the backseat computer on the payload back to the frontseat main vehicle computer in the tail section. As a result, closed-loop control on the commercial AUVs is performed entirely on their payloads, with positional information from piUSBL being used as feedback to modify MOOS-IvP [156] behaviors running on the backseat; these behaviors transmit desired depth, speed and heading set-points to the main vehicle computer to perform closed-loop control, but the main vehicle computer maintains its own estimate of vehicle position. In practice this has little effect on vehicle behavior, since the frontseat position estimate is not used by the payload; the only exception are mission aborts due to the violation of safety parameters – even if the vehicle is within the confines of the safety region, erroneous frontseat estimates of vehicle position may cause the AUV to consider itself outside the safety region. The vehicle diameter is 12.4 cm, with an overall length slightly below that of the prototype vehicle at around 105 cm.

Payload

The piUSBL payload on the commercial SandSharks is in most respects identical to the payload on the prototype AUV (chapter 2, subsection 2.3.3), and makes up the front $\frac{1}{2}$ of the vehicle. Unlike the prototype configuration, the USBL array consists of *five* rather than four hydrophone elements, configured as a regular pyramid with edge length of 8 cm as was the case for the WAM-V ASV and detailed in subsection 5.2.3 earlier in the chapter. This pyramidal USBL array is also mounted above the nose of the vehicle (as shown in Fig. 5.17), rather than at the front of the vehicle as was the case with the tetrahedral array on the prototype AUV; this mounting configuration allows us to avoid the issues of acoustic self-occlusion that were present on the prototype vehicle – the array is able to receive acoustic energy from all azimuthal directions, as long as the source is above the vehicle. As illustrated at the top of Fig. 5.17, the same circuitry is used to synchronously trigger and collect acoustic energy measured by the USBL array, including the use of a CSAC. As was the case with the prototype SandShark, 8000 data samples are collected by the DAQ from each element of the array at a rate of 37.5 kS/s, giving the piUSBL system an effective range of about 316 m. Unlike the prototype system, the payload receives power from the tail and does not contain separate power circuitry; it also does not house a WHOI Micromodem.

Receiver USBL Array and Source Signal

Since the design of the USBL array on the commercial SandShark payload was informed by the experiments performed with the WAM-V ASV, its geometry is an identical 8 cm regular pyramid. As such, the analysis for this array geometry provided earlier in this chapter in

subsection 5.2.4 still holds – source signals lower than 12 kHz work well with this array in practice.

5.9 piUSBL Calibration for the Commercial Bluefin SandShark AUV

During the experimental evaluation of the piUSBL system using the WAM-V ASV, we demonstrated that local acoustic effects can have a significant impact on the accuracy of the azimuth obtained from our acoustic angle measurement distribution (subsection 5.3.2). In addition, results from Bayesian filtering with increasing levels of simulated bias in inertial measurements indicated that single-beacon piUSBL is particularly sensitive to bias errors in heading (subsection 5.6.1). As a result, it is important to calibrate the piUSBL system on the commercial SandShark AUVs to compensate for local acoustic effects and bias heading.

To perform this calibration, we make use of the cradle setup pictured in Fig. 5.18. This setup consists of a cradle which 'clamps' around the body of the SandShark AUV, and is attached to the end of a 2.10 m length of 80/20[9] T-slot aluminum bar; at the top of this length of 80/20 we mount the Vector V102 DGPS receiver, which is powered by a Lithium Ion battery, and logs DGPS data to a Raspberry Pi 3 computer housed in the small enclosure shown at the bottom right of the picture. This cradle rigidly attaches the DGPS receiver to the AUV, and allows us to manually lower the vehicle into the water to a maximum depth of 1.85 m. Once lowered into the water, we can manually rotate the vehicle in place at a known DGPS position while keeping pitch and roll as level as possible. Heading from DGPS provides a ground-truth reference (with an accuracy of 0.75°) against which we can directly compare the AUVs MEMS AHRS heading; true azimuth to the beacon can be calculated from DGPS position (as was done for

Figure 5.18: Photo of the cradle setup used for SandShark calibration.

[9] https://www.8020.net/

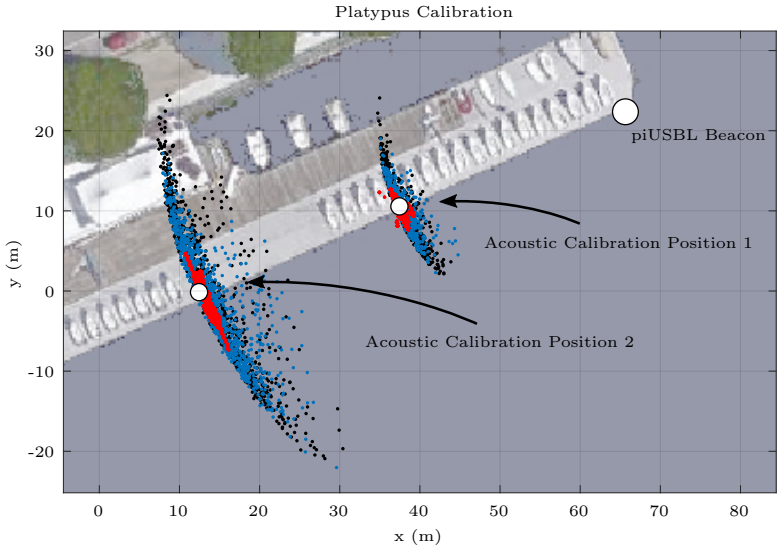

Figure 5.19: Visualization of experimental setup for SandShark (*Platypus*) calibration – the vehicle is manually lowered into the water at two calibration positions, and rotated in place to capture acoustic data from all azimuths; Uncalibrated positions from projection of MLE values of acoustic measurements are shown in black, after azimuthal calibration to mitigate local acoustic effects in blue, and after Bayesian filtering in red.

the WAM-V ASV), allowing us to compensate for local acoustic effects that bias the acoustic angle measurement distribution.

Using this cradle setup we performed a calibration experiment for each of the three commercial SandShark vehicles. First, a piUSBL beacon transmitting a $7 - 9$ kHz, 20 ms LFM up-chirp was deployed at the end of the MIT Sailing Pavilion dock at a depth of 1 m, as shown in Fig. 5.19. For each AUV, the vehicle was manually lowered into the water at two separate calibration positions, shown as white dots in Fig. 5.19, and slowly rotated in place by hand in order to capture acoustic angle measurements from all possible azimuths. Logged acoustic and navigation data from the vehicles were then used along with ground-truth DGPS logs to estimate heading bias of each vehicle and to perform acoustic calibration.

5.9.1 Estimating Heading Bias

Our cradle setup allows us to directly characterize the heading accuracy of the internal MEMS IMU used by each of the commercial SandShark AUVs. The distribution of differences in heading computed by the SandSharks *Platypus*, *Quokka*, and *Wombat*, and of that captured by the DGPS receiver, are shown as histograms in Fig. 5.20. It is apparent that the heading provided by the MEMS IMU on-board each vehicle is fairly accurate, with *Platypus*, *Quokka*, and *Wombat* having biases of $\mu \pm \sigma = -1.92 \pm 2.21^\circ$, $\mu \pm \sigma = -1.37 \pm 3.98^\circ$, and $\mu \pm \sigma = -1.47 \pm 3.02^\circ$ respectively.

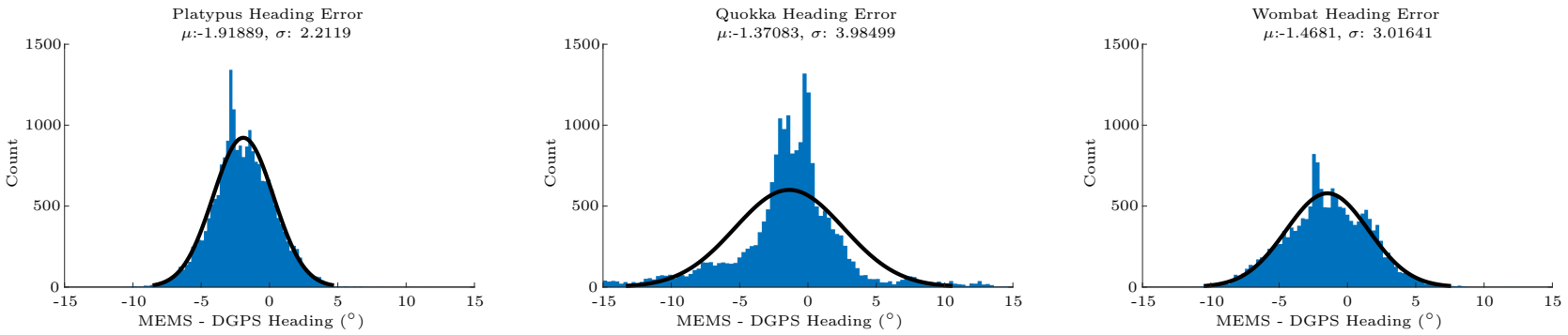

Figure 5.20: Heading difference between SandShark MEMS IMU and Vector V102 DGPS receiver – the DGPS heading is treated as ground-truth and has a stated accuracy of 0.75°. *Left*: MEMS IMU heading error for *Platypus*. *Center*: MEMS IMU heading error for *Quokka*. *Right*: MEMS IMU heading error for *Wombat*.

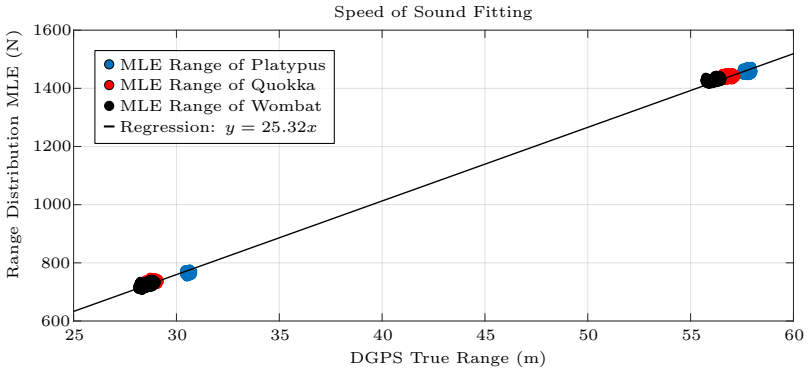

Figure 5.21: Plot of regression to determine speed of sound using MLE values from the collection of range distributions collected during calibration of the three SandShark AUVs – the MLE sample offset values for *Platypus* are shown in blue, for *Quokka* in red, and for *Wombat* in black; the linear best fit using iteratively re-weighted least squares over all data is shown as the black line.

5.9.2 Estimating Speed of Sound

Since we performed the calibration experiments at two different positions along the dock, we can make use of the same linear regression approach we used previously to estimate the speed of sound in the Charles River during this period of time. Performing the optimization of Eq. 5.2 using iteratively re-weighted least squares over the data gathered by all three vehicles, we calculate speed of sound as:

$$\chi^* \approx \frac{F_s}{c} = 25.316 \qquad c = \frac{F_s}{\chi^*} = \tfrac{37500}{25.316} \approx 1481.28 \text{ m s}^{-1} \tag{5.24}$$

This speed of sound estimate is used for acoustic position estimation in all subsequent experiments, which took place within the three months following this calibration procedure (July–September 2018). This speed of sound regression is shown in Fig. 5.21.

5.9.3 Angle Biases and Calibration

Since the USBL array is mounted above the dry payload bottle, local acoustic effects caused by the interaction of the broadcast signal with the body of the vehicle result in a bias of the beamformed angle measurement, shifting their maxima from the true values. These effects are systemic, and are likely somewhat dependent on the operating frequency of the beacon. The resulting offset between the maximum azimuth to the beacon as measured by the beamformed output, and the true azimuth as estimated by DGPS, varies depending on azimuth as illustrated at the top of Fig. 5.22; this plot shows these azimuth-dependent biases for the 7−9 kHz, 20 ms LFM up-chirp, as measured by each of the three commercial SandShark AUVs – the MLE values from the angle measurements from *Platypus*, *Quokka*, and *Wombat* are plotted using blue, red, and black dots respectively.

What is interesting about the angle biases illustrated in Fig. 5.22 is that they are remarkably consistent across all three vehicles, as well as across the two different ranges to the beacon that each vehicle was calibrated at – this strongly indicates that these biases are caused by acoustic effects local to the AUV, and are not due to environmental acoustic effects. Since each vehicle is outfitted in the same way, and each have their USBL arrays mounted at the same location on their bodies, the obvious conclusion is that it is due to the interaction between the incoming acoustic plane wave broadcast by the beacon and the body of the vehicle. The nature of this interaction is very difficult to determine – it may be due to reverberation or resonance within the dry payload bottle causing acoustic energy from the incident wave to be reflected by the body of the AUV. Examining Fig. 5.22, we see that at angles off the bow ($0 - 100^\circ$ is port and $360 - 250^\circ$ is starboard) the angle measurement tends to be 'pulled' further toward either side of the vehicle; similarly for angles off the stern ($180 - 100^\circ$ is port and $180 - 250^\circ$ is starboard) the angle measurement also tends to be 'pulled' further toward

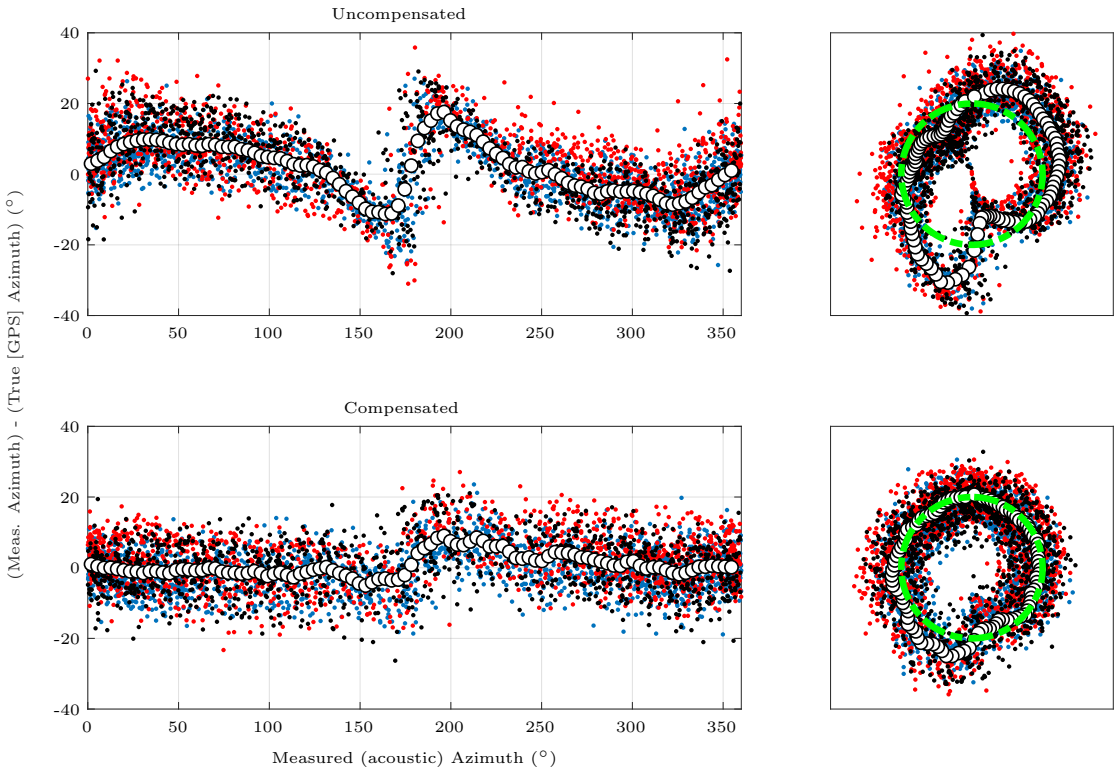

Figure 5.22: Plot of MLE azimuthal values from the angle distributions from all three SandShark AUVs versus the difference between these values and azimuths from DGPS ground-truth – angle measurements from *Platypus* are in blue, from *Quokka* in red, and from *Wombat* in black *Top*: uncompensated difference in azimuth between acoustic measurements and true (DGPS) azimuth, as well as medians of 100 bins (white); *Bottom*: after compensation by subtracting the median 'offset' of the nearest bin; *Right*: projections of biases onto a circle demonstrate that they are circularly consistent and azimuth-dependent.

either side of the vehicle. It is apparent that the azimuth measurement is biased toward the port and starboard sides of the vehicle, unless the acoustic signal arrives from dead-ahead or dead-behind the vehicle – in fact, it appears that the azimuth is 'pulled' towards the 120° mark off the bow in both directions. This suggests that a lot of acoustic energy is being reflected by the body of the vehicle, the majority of which sits behind the USBL array.

The best approach to mitigate these local acoustic effects would be to prevent reflections by using an acoustic 'baffle' on the vehicle body to absorb the incoming acoustic energy; however, since we lacked the time to implement such measures, we instead made use of the same compensation strategy that we used for the WAM-V ASV in subsection 5.3.2. The median values of these measured azimuthal biases across all three vehicles and divided into 100 equal bins are shown as the white circles in Fig. 5.22. By subtracting these median values from the azimuthal measurement after beamforming, we can compensate for these local

acoustic effects (for this particular operating frequency), resulting in the 'flattened' response in azimuth shown at the bottom of Fig. 5.22. These local acoustic effects have an enormous impact on the accuracy of the piUSBL system, and should be further investigated in future work.

5.9.4 Acoustic Measurement Accuracy

As was done with the WAM-V ASV in section 5.4 of this chapter, we use the same method to generate statistics on the range, azimuth and position accuracy of the piUSBL system for the three SandShark AUVs. DGPS measurements provide our estimate of ground-truth range and azimuth using Eqs. 5.8 and 5.9 respectively, as well as for position.

Error distributions in range and azimuth of the final piUSBL system are illustrated using histograms in Fig. 5.23 for each of the three vehicles. The corresponding statistics from Gaussian fitting of these distributions are listed in table 5.4. These values indicate that the piUSBL system on *Platypus*, *Quokka*, and *Wombat* can provide range accuracies of $\mu \pm \sigma = 0.22 \pm 0.27$ m, $\mu \pm \sigma = 0.68 \pm 0.32$ m, and $\mu \pm \sigma = 0.73 \pm 0.26$ m respectively; as well as angle accuracies of $\mu \pm \sigma = -0.43 \pm 7.81°$, $\mu \pm \sigma = 2.02 \pm 8.09°$, and $\mu \pm \sigma = 0.35 \pm 8.83°$ respectively. Interestingly, the standard deviations for range are significantly smaller than those measured by the WAM-V ASV, and is similar to the theoretical range resolution of a 2 kHz bandwidth signal – this is likely due to code changes that were made to the piUSBL beacon to reduce jitter. Unfortunately, the standard deviations for angle are significantly worse than those measured by the WAM-V – this is most likely due to the more prominent local acoustic interactions that the system experiences with the SandShark platform. The rightmost plots in Fig. 5.23 illustrate the cumulative distribution functions (CDFs) of absolute error in range and azimuth using the combined data from all three AUVs; these plots indicate that 68% of raw range measurements have an absolute error of less than 0.7 m, and 68% of raw azimuth measurements fall below an absolute error of 7.0° – recall that the corresponding values for the WAM-V ASV were 1.25 m and 2.15° respectively.

For the sake of interest, we also project the MLE values from the range and angle distributions so as to estimate vehicle position in the LLF, and compare these positions against DGPS. These error distributions for position are illustrated in Fig. 5.24 for each of the three vehicles, which also shows their marginal distributions in x and y. Statistics for the Gaussian fits to these distributions are also listed in table 5.4. As previously mentioned, note that these statistics are only valid for the ranges at which the calibration experiments were performed, which were less than 60 m to the beacon, and so they are not particularly informative.

Finally, we note that Fig. 5.19 illustrates the improvement in positioning accuracy obtained by first compensating azimuth measurements for local acoustic effects, and then by

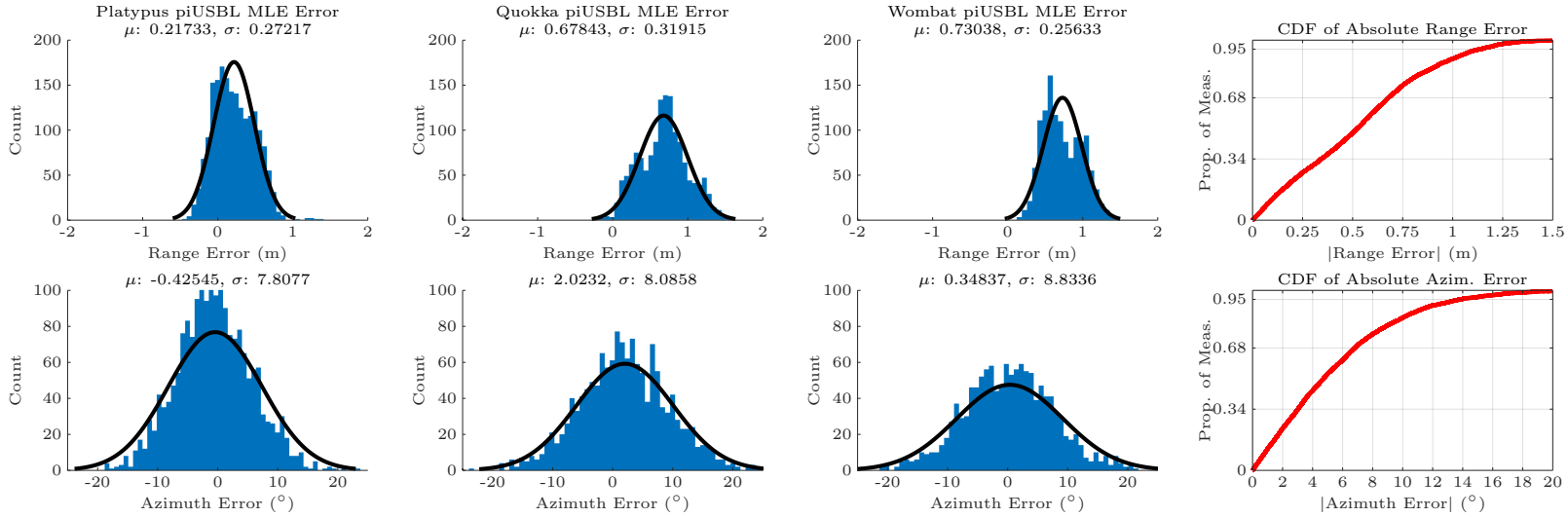

Figure 5.23: Range and azimuth differences in value between MLE from acoustic measurement distributions and from DGPS for the three SandShark AUVs – range and azimuth error histograms between MLE values from piUSBL acoustic measurement distributions against DGPS are shown for *Platypus*, *Quokka*, and *Wombat*, along with Gaussian fits. *Left*: error histograms for *Platypus*. *Center Left*: error histograms for *Quokka*. *Center Right*: error histograms for *Wombat*. *Right*: empirical cumulative distribution functions (CDFs) for absolute range and azimuth errors using combined data from all three vehicles.

	Range Error (m)		**Azimuth Error (°)**		**Position Error (m) at ranges < 60 m**				
	$\boldsymbol{\mu}_r$	$\boldsymbol{\sigma}_r$	$\boldsymbol{\mu}_\phi$	$\boldsymbol{\sigma}_\phi$	$\boldsymbol{\mu}_{x,y}$	$\boldsymbol{\sigma}_{major}$	$\boldsymbol{\sigma}_{minor}$	$\boldsymbol{\sigma}_x$	$\boldsymbol{\sigma}_y$
***Platypus* AUV**	0.217	0.272	-0.425	7.808	(-0.277, 0.999)	5.657	2.649	5.652	2.660
***Quokka* AUV**	0.678	0.319	2.023	8.086	(1.252, 0.273)	5.638	1.197	5.635	1.212
***Wombat* AUV**	0.730	0.256	0.348	8.834	(-0.109, 0.598)	5.838	2.143	5.838	2.144

Table 5.4: Error statistics from MLE using range and angle measurement distributions against DGPS for range, azimuth and position for all three SandShark AUVs.

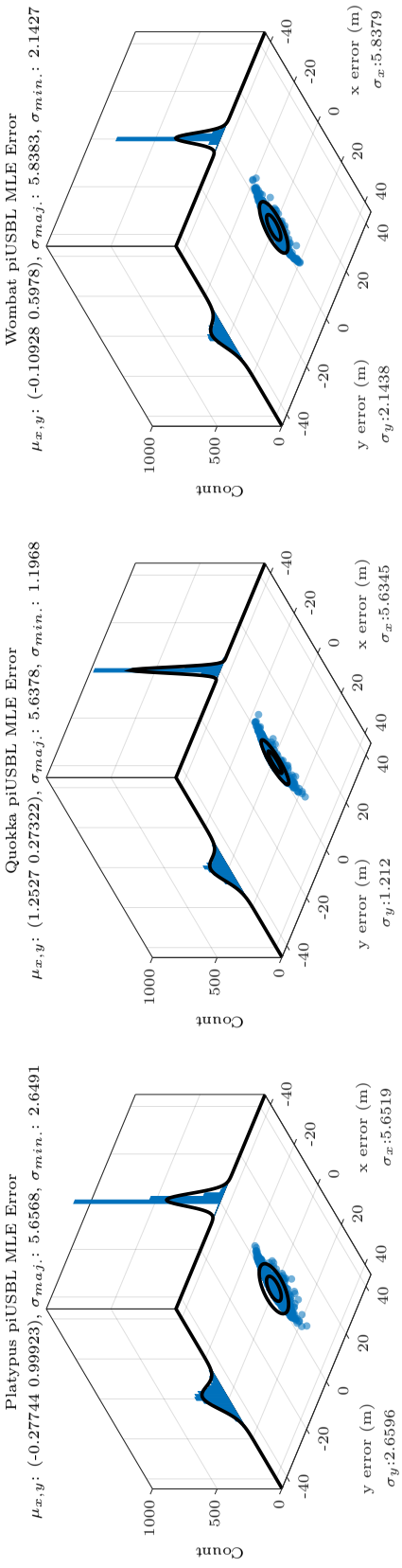

Figure 5.24: Position differences in value between MLE from acoustic measurement distributions and from DGPS for the three SandShark AUVs – the distributions of position error between MLE values from acoustic measurements gathered by each vehicle against DGPS are shown as blue circles, with 1σ and 2σ ellipses in black; the associated marginal error histograms in the x and y dimensions are projected on the 'walls', with Gaussian fits in black. *Left*: position errors for piUSBL on *Platypus*. *Center*: position errors for piUSBL on *Quokka*. *Right*: position errors for piUSBL on *Wombat*.

Bayesian filtering of these acoustic measurements with inertial measurements using the sequential Monte-Carlo beamformer (SMCB). The positions obtained from projection of range and uncompensated azimuth MLE values in black are worse than those obtained from projection of MLE values with azimuth compensation in blue, which are still worse than those obtained after Bayesian filtering in red – Bayesian filtering substantially improves localization.

5.10 Summary of the Commercial Bluefin SandShark AUV

We have introduced the three commercial Bluefin SandShark AUVs *Platypus*, *Quokka*, and *Wombat*, and the implementation of the final piUSBL positioning system on these vehicles, whose design was informed by the experimental results obtained by first fielding the system on the WAM-V ASV. In these preceding series of sections, we described the hardware of our SandShark fleet, and detailed the calibration procedure we undertook to characterize the vehicles' MEMS AHRS errors in heading, as well as to compensate for systemic local acoustic interactions experienced by the piUSBL system on all three AUVs. In addition, statistics characterizing the accuracy of raw MLE range and angle measurement distributions obtained by the piUSBL system as implemented on these commercial SandShark vehicles were provided, which demonstrated an improvement in range accuracy as compared to piUSBL on the WAM-V; importantly, these statistics also demonstrated a reduction in angle accuracy as compared to the WAM-V, which is most likely a consequence of the stronger influence of local acoustic interactions of the broadcast acoustic signal and the torpedo-shaped body of the SandShark vehicle. These biases in the beamformed output have a significant impact on the accuracy of our piUSBL angle measurements, and consequently reduce the positioning accuracy of our system – methods for alleviating these biases should be a priority for future work.

This chapter and the preceding three chapters have laid out in detail the entire piUSBL system stack as used by our fleet of SandShark AUVs: this chapter described the hardware and calibration procedures required to operate the fleet using piUSBL positioning; chapter 2 explained the general processing pipeline of the piUSBL system; chapter 3 described the modifications to the piUSBL particle filter that enable closed-loop performance; and chapter 4 detailed the beamforming method that improves the resolution and precision of piUSBL angle measurements. In the following chapter we describe the behaviors and novel operating principles that allow us to command and control this fleet of AUVs using our piUSBL system for *relative navigation*.

Chapter 6

Experiments with Multiple Autonomous Underwater Vehicles: *Relative Navigation using piUSBL*

6.1 Introduction

Passive Inverted Ultra-Short Baseline (piUSBL) navigation has three significant benefits that imbue it with the real possibility of democratizing underwater vehicle technology and for making multi-AUV deployments more common: (i) the passive nature of the piUSBL receiver allows an arbitrary number of vehicles to self-localize; (ii) it enables the *relative navigation* operating paradigm, providing a method for fleet-wide control among other advantages; and (iii) its features of low-cost and low-power make it an ideal navigation solution for inexpensive and miniature autonomous underwater vehicles (AUVs), which is a necessity in allowing operators to move away from an expensive, complex and large single vehicle toward multiple vehicles at a similar level of cost.

To demonstrate these advantages, this chapter presents results of relative navigation experiments with a fleet of three commercial Bluefin SandShark AUVs, each equipped with our final piUSBL payload. The preceding four chapters have paved the way toward enabling these multi-AUV deployments, providing these vehicles with a navigation system that is robust and accurate, and providing the fleet operator with an operating paradigm that is intuitive and straightforward to use. In this chapter we describe the *relative navigation* operating paradigm for multi-AUV deployments, detailing the custom autonomous behaviors that allow the vehicles to operate under this paradigm, and we explain how this approach enables the operator to perform fleet-wide command and control of all vehicles simultaneously. We provide experimental results from these multi-AUV deployments, with a total of 12 deployments performed with three vehicles, representing a combined total vehicle runtime of more than 36 hours across

all AUVs. Finally, we provide some proof-of-concept demonstrations of useful applications for multi-AUV deployments.

6.2 The piUSBL Relative Navigation Operating Paradigm

We first introduced the concept of *relative navigation* for AUVs in chapter 3, where we demonstrated preliminary experiments of this operational paradigm using the prototype SandShark vehicle in subsection 3.5.2, as well as with a conventional Bluefin-21 AUV in section 3.7. In both these experiments only a single vehicle was used, both of which made use of a single type of relative behavior – the dynamic loiter, in which the AUV always attempts to maintain a standoff distance to the beacon as it circles around it. In this section we describe how this relative navigation concept scales to a fleet of vehicles, and introduce additional relative behaviors as well as system functionality that allows the operator to command the fleet to switch between different behaviors in-situ.

6.2.1 The Commercial Bluefin SandShark AUV Fleet

Our fleet of AUVs is comprised of three commercial SandShark vehicles, which we introduced at the end of the previous chapter in section 5.8. Each of these miniature, low-cost AUVs is outfitted with our final piUSBL receiver, using a five element pyramidal array mounted above the nose of each vehicle – the processing stack makes use of our innovations in filtering and beamforming detailed in chapters 3 and 4. These vehicles are named *Platypus*, *Quokka*, and *Wombat*, and can be seen in the photograph of Fig. 6.1.

6.2.2 AUV Fleet Behaviors and Relative Autonomy

Typically, AUV behaviors such as waypoints, racetracks, loiters, and lawnmower paths are defined within an absolute frame of reference. As demonstrated in our closed-loop absolute navigation experiments in chapter 3, subsection 3.5.1, when the piUSBL beacon is fixed at a known location, the vehicle is able to carry out such absolute behaviors fairly accurately (especially in comparison to dead-reckoning) using piUSBL navigation. However, when operating in a frame of reference that is defined *relative* to an object in the environment, if the absolute position of the object is *unknown* to the vehicle, then there is no possible way for the vehicle to undertake absolute behaviors – this is the case that we encounter with moving beacon *relative navigation* using piUSBL. In this case, we must define new vehicle behaviors that are specifically designed for use with a relative reference frame that moves dynamically within the absolute frame of reference. We refer to the operational use of such relative behaviors as *relative autonomy*, where these behaviors are defined in a beacon-centric coordinate system in which the beacon is always assumed to be at the origin. If the position of the beacon in the

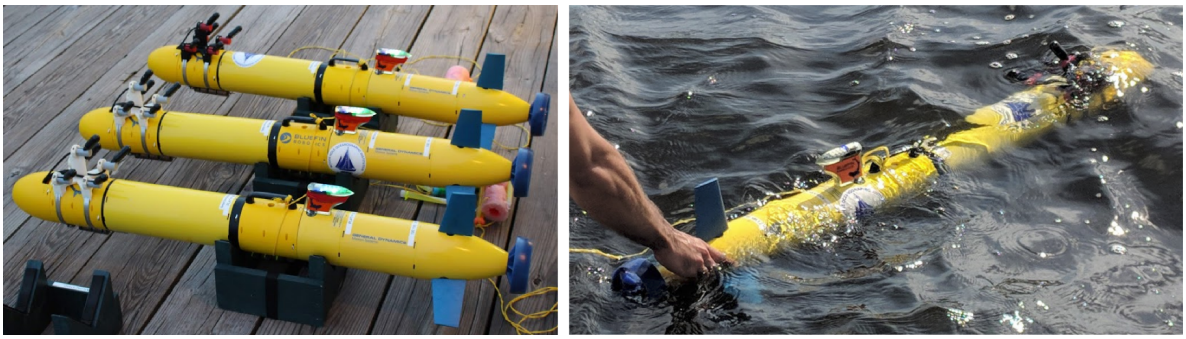

Figure 6.1: Photograph of our fleet of three commercial Bluefin SandShark AUVs, *Platypus*, *Quokka*, and *Wombat* – *Right*: *Quokka* being deployed.

absolute coordinate system is known, offsetting of the relative navigation solution by beacon position allows us to calculate the absolute position of each vehicle in post-processing. We implement these relative behaviors using the MOOS-IvP autonomy framework [156].

The Relative Loiter Behavior

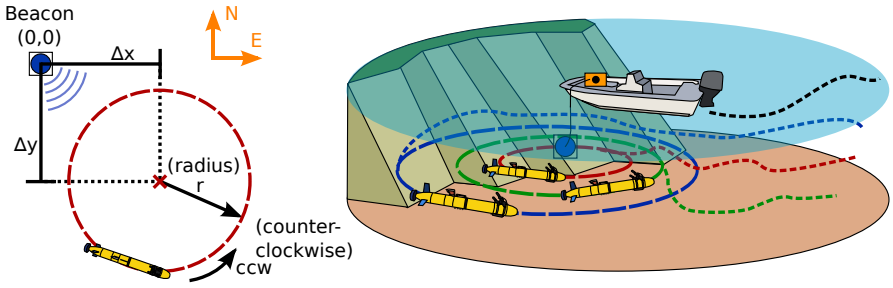

Figure 6.2: Diagram of the relative loiter behavior – by setting different loiter radii for each AUV, the fleet of vehicles can loiter around the beacon with a very low risk of collision.

	Relative Loiter Parameters	
	Domain	**Description**
Speed	$\mathbb{R}_{\geq 0}$	speed-over-ground during loiter ($\mathrm{m\,s^{-1}}$)
Radius (r)	$\mathbb{R}_{\geq 0}$	loiter radius (m)
Counter-Clockwise (ccw)	True/False	loiter counter-clockwise (True)
West-East Offset (Δx)	$\mathbb{R}$	West-East offset from beacon (m)
North-South Offset (Δy)	$\mathbb{R}$	North-South offset from beacon (m)

Table 6.1: Important vehicle-specific parameters for the relative loiter behavior.

The dynamic loiter behavior that we demonstrated with the Bluefin-21 AUV in section 3.7 is the simplest example of such a relative behavior. From the perspective of the vehicle, it always attempts to maintain a relative standoff distance to the beacon – if the beacon moves

in the absolute reference frame, to the vehicle this appears as a shift in its relative position and it compensates accordingly; from the perspective of the operator, this dynamic loiter appears as a circular track that is centered at the beacon and moves with it in the absolute frame of reference. In fact we generalize this behavior further for our final *relative loiter* behavior – we parameterize an offset of the loiter along the West-East and North-South axes, allowing the operator to offset the circular track relative to the beacon. By setting these parameters to zero, we revert back to the original *relative loiter* behavior demonstrated by the Bluefin-21. An illustration of the *relative loiter* behavior is shown in Fig. 6.2, with its parameters listed in table 6.1.

The Relative Line Behavior

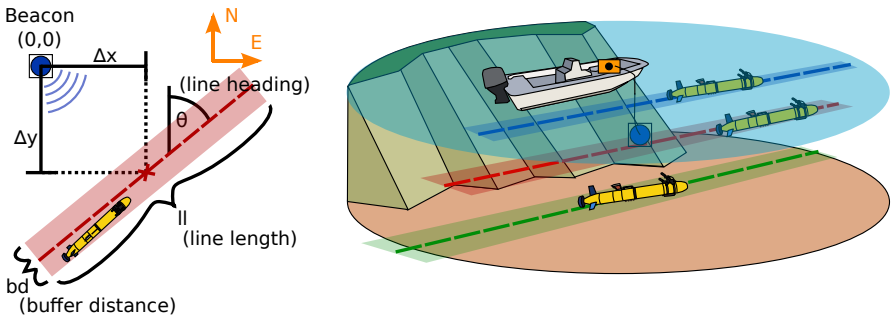

Figure 6.3: Diagram of the relative line behavior – by setting different line offsets for each AUV, the fleet of vehicles can sample and survey along parallel tracks.

	Relative Line Parameters	
	Domain	**Description**
Speed	$\mathbb{R}_{\geq 0}$	speed-over-ground during line ($\mathrm{m\,s^{-1}}$)
Line Length (ll)	$\mathbb{R}_{\geq 0}$	length of line transect (m)
Line Heading (θ)	$\mathbb{R}$	heading of line transect (°)
Buffer Distance (bd)	$\mathbb{R}_{\geq 0}$	width of buffer distance (m)
West-East Offset (Δx)	$\mathbb{R}$	West-East offset from beacon (m)
North-South Offset (Δy)	$\mathbb{R}$	North-South offset from beacon (m)

Table 6.2: Important vehicle-specific parameters for the relative line behavior.

The *relative line* behavior is designed to allow a vehicle to sample along a finite line transect at a specified heading – the center of this line is positioned at a specified relative offset from the beacon. If the beacon is static, the result is that the AUV continuously runs back-and-forth along this finite line; however, if the beacon travels in a direction perpendicular to the transect, the operator is able to 'sweep' the transect over an area in the absolute frame of reference, causing the AUV to naturally survey that area – from the perspective of the vehicle,

the relative distance between itself and the line transect increases, and so it closes the distance as it attempts to converge to and track the line. An illustration of the *relative line* behavior is shown in Fig. 6.3, with its most relevant parameters listed in table 6.2.

In the original implementation of the *relative line* behavior in MOOS-IvP, we did not include the 'buffer distance' illustrated in Fig. 6.3 – this caused difficulties in the ability for the vehicle to track the line, resulting in oscillatory behavior. These oscillations were due to the fact that small movements in the beacon, as well as uncertainty in the state estimate of the relative beacon position, caused the line transect to shift laterally; without reaching a stable state, this movement in the line prompted continuous control adjustments in the vehicle. Initial multi-AUV experiments reflect this instability in this behavior. The buffer distance significantly reduces this effect. It does so by forcing a control response in the vehicle only when the vehicle breaches the bounds of this distance – when this occurs, the AUV attempts to center itself on the transect line. If the vehicle has placed itself on the line, it simply attempts to maintain constant heading until it breaches the buffer bounds again.

The Offset-Follow Behavior

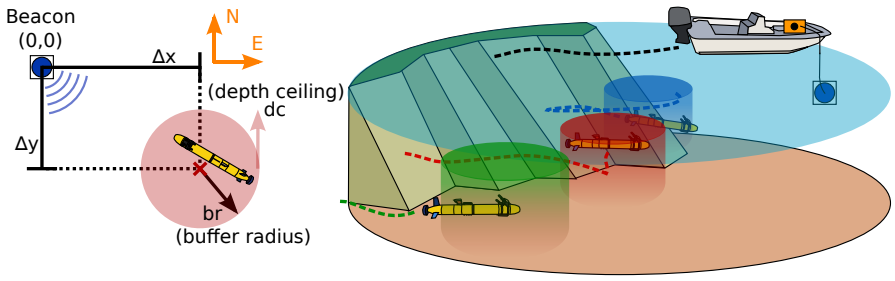

Figure 6.4: Diagram of the offset-follow behavior – by setting different offsets for each AUV, the fleet of vehicles attempt to maintain a geometry of positions relative to the beacon.

	Offset-Follow Parameters	
	Domain	**Description**
Speed	$\mathbb{R}_{\geq 0}$	speed-over-ground during follow ($\mathrm{m\,s^{-1}}$)
Buffer Radius (br)	$\mathbb{R}_{\geq 0}$	radius of buffer circle (m)
Depth Ceiling (dc)	$\mathbb{R}_{\geq 0}$	minimum depth ceiling (m)
West-East Offset (Δx)	$\mathbb{R}$	West-East offset from beacon (m)
North-South Offset (Δy)	$\mathbb{R}$	North-South offset from beacon (m)

Table 6.3: Important vehicle-specific parameters for the offset-follow behavior.

The *offset-follow* behavior is a simple waypoint-like behavior, in which the vehicle always attempts to drive towards a position defined by a West-East and North-South $(\Delta x, \Delta y)$ offset from the beacon. Once it has reached this position, the vehicle completely stops its propeller

and slowly floats toward the surface[1]. The behavior has a defined 'buffer radius' centered at this position, within which the vehicle will not thrust once it has reached its position – if the vehicle exits the bounds of this buffer radius, which usually occurs if the beacon moves and the offset position moves along with it, then the vehicle restarts its propeller to drive itself toward the offset position again. In addition, the behavior defines a 'depth ceiling' – if the vehicle floats above a depth shallower than this ceiling, then the AUV also restarts its propeller to drive itself back to the specified depth at the offset position, essentially circling back to its original position if the beacon has not moved. From the perspective of the vehicle, it essentially always works to keep the beacon at a standoff position from itself, using a sprint-and-stop mechanic to prevent itself from floating above a certain depth. From the perspective of the operator, this appears as a kind of following behavior (or formation keeping behavior), where the vehicle always attempts to maintain its position relative to the beacon as it moves. An illustration of the *offset-follow* behavior is shown in Fig. 6.4, with its most relevant parameters listed in table 6.3.

The Return and Surface Behavior

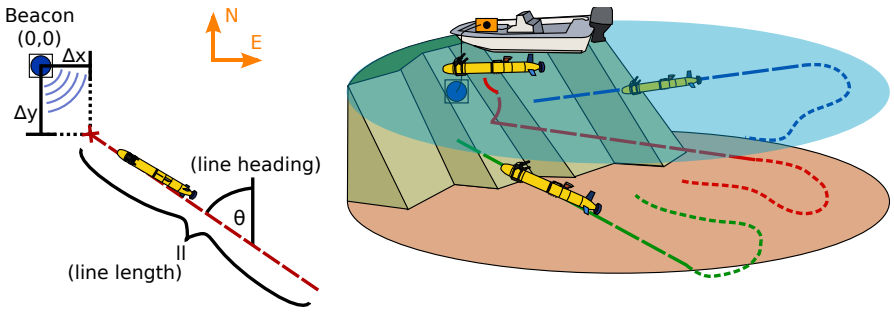

Figure 6.5: Diagram of the return and surface behavior – by setting different offsets and line headings for each AUV, the fleet of vehicles return and surface close to the beacon.

	Return and Surface Parameters	
	Domain	**Description**
Speed	$\mathbb{R}_{\geq 0}$	speed-over-ground during line ($m\,s^{-1}$)
Line Length (ll)	$\mathbb{R}_{\geq 0}$	length of return line (m)
Line Heading (θ)	$\mathbb{R}$	heading of return line (°)
West-East Offset (Δx)	$\mathbb{R}$	West-East surfacing offset from beacon (m)
North-South Offset (Δy)	$\mathbb{R}$	North-South surfacing offset from beacon (m)

Table 6.4: Important vehicle-specific parameters for the return and surface behavior.

The *return and surface behavior* behavior is formulated as a convenient method of recalling

[1]The AUVs are ballasted to be just slightly positively buoyant, allowing them to float very slowly at a rate of less than 10 $cm\,s^{-1}$ toward the surface.

all the vehicles in the fleet back to the operator at the beacon – it defines a finite return line whose end is offset from the beacon by relative offsets in the West-East and North-South axes, and which the vehicle attempts to follow from the start to the end. Upon reaching the end of this return line, the vehicle stops its propeller and freely floats to the surface, at which point it has deemed its deployment to be complete. The return line in this behavior is rotated by a specified heading, allowing the vehicle to return to the beacon from any direction as desired by the operator. Using this behavior allows the operator to command all vehicles to return to the beacon for easy retrieval and end the mission. An illustration of the *return and surface behavior* is shown in Fig. 6.5, with its most relevant parameters listed in table 6.4.

The Abort Behavior

Finally, the *abort behavior* is extremely simple – when this behavior is triggered, the vehicle stops all action and freely floats to the surface.

6.2.3 AUV Fleet Command with Signal and Mode Switching

In chapter 4 we introduced the element pair decomposition (EPD) beamformer, a novel beamforming approach that proved to be very computationally efficient for arrays with a small number of elements. The EPD beamformer also had a secondary advantage in terms of memory usage – recall that to speed up the beamforming process, it is necessary to precompute and store the phase shifts associated with each look-angle. Consider a regular grid of look-angles, equally spaced in the azimuth and inclination axes such that there are 360 azimuths and 180 inclinations – the total number of look-angles is $360 \times 180 = 64800$. Let us compare the memory consumption again between the conventional beamformer (CBF) and the EPD beamformer for our five-element pyramidal array, assuming $NFFT = 1024$ to represent the frequency range in our chirp Z-transform (CZT):

- **conventional beamformer (CBF)**: Total memory usage is $64800 \cdot 16 \cdot 1024 \approx 1062$ MB ≈ 1 GB.
- **element pair decomposition (EPD) Beamformer**: Let us assume that we have 360 coning angles for each pair of elements; for our pyramidal array we have two *unique pair lengths* – the distance between each pair of elements is 8 cm, except for the pairs across the diagonal of the pyramid base, which are 11.3 cm apart. For each unique pair length we have to store 360 phase shifts for each coning angle, which are then mapped to the grid of look-angles via simple trigonometry. Thus, the total memory usage is $2 \cdot 360 \cdot 16 \cdot 1024 \approx 11.8$ MB. In fact, if we stored the phase shifts for each unique pair of elements, of which there are 10 for the pyramidal array, the total memory usage will only increase by a factor of five to 59 MB. Similarly, increasing the number of coning angles

by a factor of two to 720 would only double the memory usage to 118 MB. The EPD beamformer enables an enormous reduction in memory usage, reducing the amount of memory required by *one to two orders of magnitude* compared to the CBF.

The enormous memory savings afforded to us by EPD beamforming allow us to implement a simple centralized command structure, in which the operator can command different fleet 'modes' by switching between different signals broadcast by the piUSBL beacon. Notice that in both cases memory usage is tied to the frequency domain resolution, $NFFT$; this means that the wider the frequency range of the signals used in our piUSBL system, the more memory is required for a given 'fineness' of frequency. In order to more easily distinguish between different broadcast signals, it is advantageous to use signals in different frequency ranges; however, we cannot afford the corresponding increase in memory usage if we use the CBF – but with the EPD beamformer, the memory usage is so small that enough memory is available for the increase in the frequency range and related memory usage.

For fleet-wide command, we use four different linear frequency modulation (LFM) chirps, each corresponding to different fleet 'modes':

- **Mode 0**: No signal broadcast into the water (occurs only during initial deployment).
- **Mode 1**: 7–9 kHz, 20 ms LFM up-chirp.
- **Mode 2**: 10–8 kHz, 20 ms LFM down-chirp.
- **Mode 3**: 8–6 kHz, 20 ms LFM down-chirp.
- **Mode 4**: 9–11 kHz, 20 ms LFM up-chirp.

The operator switches between modes by broadcasting these different beacon signals, and does so by turning a physical rotary dial installed on the piUSBL beacon box. For each mode there is a corresponding relative behavior defined on each AUV, with behavior parameters uniquely set for each vehicle. For example, *mode 1* can be set to command a *relative line* behavior on each vehicle, *mode 2* can be set to command a *relative loiter*, *mode 3* to command a *return and surface*, and *mode 4* an abort; by switching between different modes, the fleet of AUVs will switch their behavior accordingly to undertake the corresponding relative behavior assigned to that mode. This provides the operator with an intuitive method of *commanding* all vehicles in the fleet *simultaneously*.

Note that because each AUV in the fleet essentially operates independently without any knowledge of the state of the other vehicles, it is vitally important that behavior parameters are carefully set for each vehicle such that AUVs are not at risk of colliding with one another. For example, with the relative loiter behavior, radii or offsets can be set so that the circular paths of each vehicle do not intersect; or with the relative line behavior, the line heading

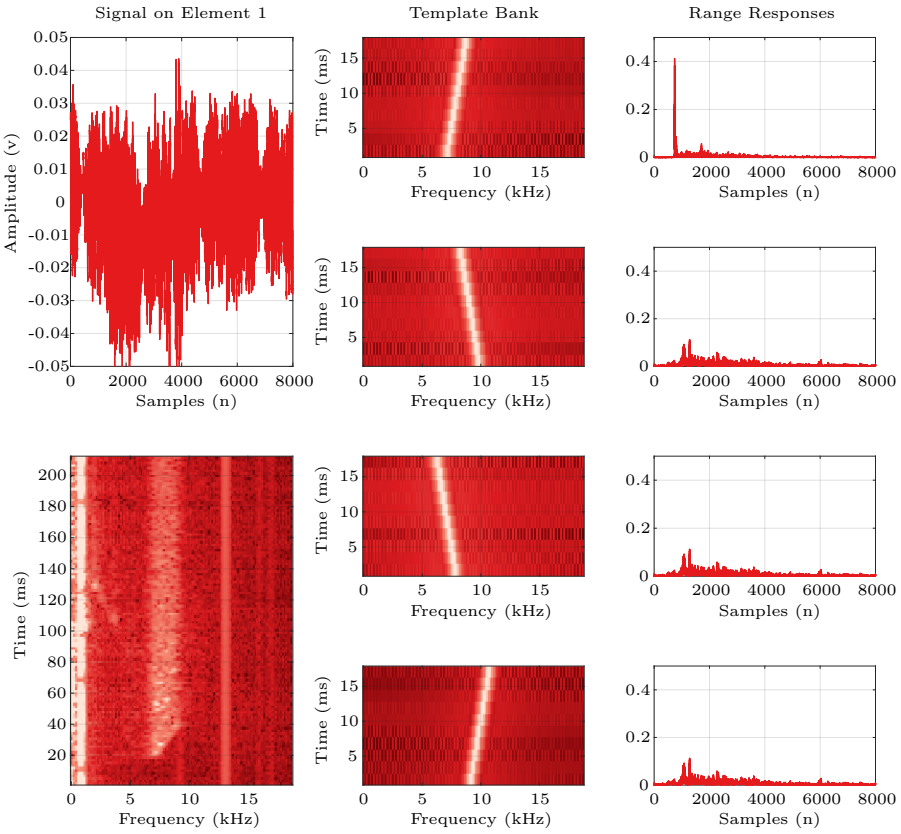

Figure 6.6: Visualization of the filter bank to detect the broadcast beacon signal and to detect the commanded mode – *Left Column*: the in-water signal recorded by the first element of the USBL array, with its spectrogram. *Middle Column*: spectrograms of each template in the bank of templates. *Right Column*: the range measurement distribution response for each template. The rows in the middle and right columns represent the templates for *mode 1*, *mode 2*, *mode 3*, and *mode 4*; it is apparent that the first template, with the 7–9 kHz, 20 ms LFM up-chirp, elicits the largest response, indicating that *mode 1* has been selected by the operator.

for all vehicles should be equal, and there should be a large enough offset between lines to generate parallel transect lines across the fleet.

To detect the correct mode, the piUSBL receiver on each vehicle simply generates a range measurement distribution using a bank of templates containing the four possible signals that the beacon can broadcast. The template that generates the largest response is then selected as the most likely broadcast signal; if the same template generates the largest response 3 *times in a row*, then the vehicle deems that the beacon has commanded the corresponding mode, and the AUV enters the relative behavior associated with that mode. As always, the acoustic measurements are only valid if they pass the matched filtering validity check. An example of this mode detection scheme is illustrated in Fig. 6.6, in which *mode 1* has been commanded.

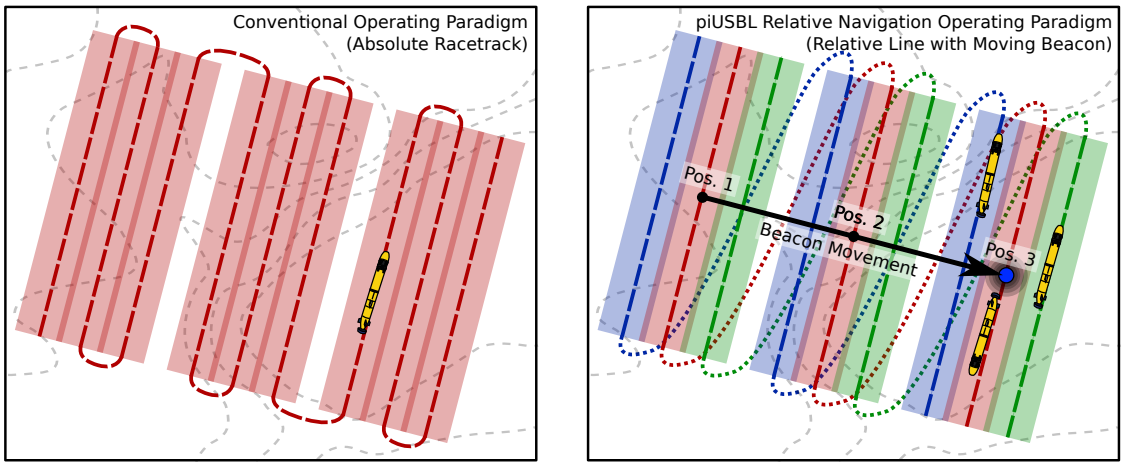

Figure 6.7: Conceptual illustration of sidescan survey using conventional and relative navigation operating paradigms – *Left*: in the conventional operating paradigm, a racetrack must be defined in the absolute reference frame for the AUV to follow. *Right*: in the piUSBL relative navigation operating paradigm, the AUVs always navigate relative to the beacon, which can move; movement of the beacon perpendicularly to the relative line behavior of the vehicles allows the operator to perform the same sidescan survey using multiple vehicles.

6.2.4 AUV Fleet Control with Beacon Movement

Our relative navigation approach enables centralized fleet-wide command via mode-switching by broadcasting different signals using the piUSBL beacon. It also implicitly enables fleet-wide *control* since relative behaviors are defined in a beacon-centric frame of reference – *movement of the beacon itself allows the operator to control the position of the fleet in the absolute frame of reference.*

Although conceptually very simple, this approach is powerful. Consider for example the scenario illustrated in Fig. 6.7; using a conventional operating paradigm with a single AUV, performing a sidescan survey of a patch of seafloor would require the operator to program a lawnmower path over the desired area, as shown to the left of Fig. 6.7; alternatively, using multiple low-cost AUVs and the relative line behavior, the same patch can be surveyed via the movement of the beacon itself as shown to the right of Fig. 6.7 – as the operator repositions the beacon at the three different positions, the fleet of vehicles reposition themselves automatically, resulting in the same coverage of the desired area. By recording the position of the beacon with a high accuracy, the relative positions of the three AUVs can be accurately transformed into the absolute frame of reference.

This combination of designing relative behaviors in a beacon-centric reference frame, fleet-wide command via mode-switching, and fleet-wide control via beacon movement, collectively represent a powerful command-and-control approach that we call the *piUSBL relative navigation operating paradigm.* This operating paradigm is intuitive, simple to use, and easy

to configure, and we believe that it is a significant first step toward simplifying multi-AUV operations. By making these deployments more approachable and less complex, we hope to make multi vehicle experiments more common.

6.3 Experimental Results

The entire piUSBL navigation system in conjunction with the relative navigation operating paradigm was tested with multi-AUV deployments in the Charles River next to the MIT Sailing Pavilion in August and September of 2018. These deployments made use of three commercial (production-level) Bluefin SandShark AUVs named *Platypus*, *Quokka*, and *Wombat*, which we have previously described at the end of chapter 5 in section 5.8. The piUSBL system on each vehicle runs our sequential Monte-Carlo beamformer (SMCB) with 500 particles in order to ensure the cycle time of the filter was below 1 s – with 500 particles an iteration of the filter takes about 500 ms on the Raspberry Pi 3. The system also uses our element pair decomposition (EPD) beamformer for angle estimation, with each pair of elements precomputing the phase shifts of 360 coning angles. Each vehicle runs under the frontseat-backseat paradigm, with the relative MOOS-IvP [156] behaviors detailed in subsection 6.2.2 sending desired heading, depth and speed commands from the payload Raspberry Pi 3 to the main vehicle computer in the tail to carry out. As previously explained, navigation information from piUSBL is not fed back into the main vehicle computer, but is only used by the backseat for closed-loop control by continuously sending desired values to the frontseat. This has little effect on vehicle behavior, although it would sometimes cause the vehicle to abort and surface during missions due to its belief that it traveled outside the confines of the safety area of the mission – when this happened, GPS placed it back within the safety area and it then continued the mission. Enabling proper navigational feedback to the main vehicle computer is the subject of ongoing work to be done with Bluefin Robotics.

As mentioned before, a total of 12 deployments were carried out during the course of the month, using all three AUVs. With each mission lasting for approximately an hour, this represents about 36 hours of total vehicle runtime across all three vehicles. In this section we present results from the last 6 of these deployments, which represent the final performance attained using our piUSBL system. Data from the initial 6 deployments is still relevant, but represent deployments that were used to fine-tune the system and make it more robust – they allowed us to correct bugs in behaviors and the associated state-machine, as well as improve the practical operational procedures of the multi-AUV relative navigation paradigm.

In each of the following missions all vehicles were programmed to dive to and maintain a desired depth of 2.5 m and to maintain a desired speed-over-ground of 1 $\mathrm{m\,s^{-1}}$. For every mission the piUSBL navigation beacon was secured by rope to a motorboat at a depth of approximately 1 m; this motorboat was operated by one or two people, who would drive the

motorboat to move the beacon to position the fleet in the absolute frame of reference, and who would manually rotate the beacon dial to select different modes and command all vehicles in the fleet to switch their behavior. The motorboat was equipped with the Hemisphere V102 DGPS introduced in chapter 5, which allowed us to record and timestamp the position of the piUSBL beacon with decimeter-level accuracy.

For each experiment, two additional acoustic beacons were fastened to the dock at the positions of (17.05 m, 1.78 m) and (−60.56 m, −34.97 m) in our absolute frame of reference, and at a depth of approximately 1 m. These beacons were not used by the vehicles for the purposes of navigation – they fire in sync with the moving piUSBL beacon on the motorboat, and their signals as recorded by the vehicles were processed *offline* to obtain reference trajectories of the AUVs for comparison of the piUSBL navigation solution; we essentially use these beacons as 'ground-truth' position as calculated from passive LBL (chapter 5, subsection 5.4.4), which we demonstrated to have accuracy on the same level as that of consumer GPS in chapter 5, subsection 5.4.5. These two beacons broadcast a 5–2 kHz, 20 ms LFM down-chirp, and a 250–1500 Hz, 20 ms LFM up-chirp respectively. There is an exception to their use as a navigational aid, however – the beacon fixed at (17.05 m, 1.78 m) is sometimes commanded to broadcast the signal for *mode 3*, and the motorboat beacon is switched off, in order to command all vehicles into the *return and surface* behavior in order to have all vehicles return to the dock.

6.3.1 Description of Figures Showing Experimental Results

Since each mission makes use of three vehicles, and each vehicle operates in a relative frame of reference centered around the boat-based moving beacon using multiple different behaviors, the data from these missions is *dense*, and surfacing this data to make it easily understandable is not simple. Before presenting our results, we walk through and explain what the plots in Fig. 6.8 and Fig. 6.10 are displaying, in order not to repeat the exercise for each mission.

In these figures, data relating to *Platypus* is always plotted in red, data relating to *Quokka* in blue, and data relating to *Wombat* in green. Let us begin with Fig. 6.8:

- Bottom Row: These three plots show the piUSBL navigation trajectory of individual vehicles over the course of the entire mission in red, blue or green, with darker shades indicating early parts of the trajectory and lighter shades indicating later parts. The position of the motorboat and piUSBL beacon is shown as connected white dots. Finally, the position solution from the intersection of passive LBL range circles from the two dockside beacons are shown as black dots.
- Top Row: These six plots show sections of the mission, and plot piUSBL vehicle trajectories for all three vehicles together. The absolute position plots are the vehicle trajectories transformed into the absolute frame of reference by offsetting (in post-processing) the

trajectories in the relative position plots by the piUSBL beacon position – the absolute and relative position plots are coupled to show the same section of time within the mission. Again, dark shades of color indicate early mission times, while light shades indicate later mission times, and the position of the piUSBL beacon is again shown using connected white dots – notice in the relative position plots that the beacon is always at the origin.

- Middle Row: This single long plot indicates the mode detected by each vehicle over the course of the entire mission, with colors indicating vehicle and shade of color indicating time. As such, this plot can be used to determine which shade of color in all plots correspond to which mission time, as well as the mode that each vehicle is in during each moment of the mission. The dashed vertical black lines delineate how the mission has been sectioned into three parts for individual display in the top row – for example, the section from 0–1800 s is displayed in the first two plots in the top row, the section from 1800–3200 s is displayed in the second two plots in the top row, and 3200+ s in the final two plots in the top row.

Recall from chapter 3 section 3.4 that the particles in the sequential Monte-Carlo beamformer (SMCB) are randomly uniformly initialized when the vehicle operates in the relative navigation paradigm – as such, the trajectories in these plots only show the piUSBL navigation solution when the standard deviation of the particles in both the major and minor axes is less than 15 m, indicating that the system has confidence in its navigation solution.

The plots in Fig. 6.10 illustrate the following:

- Top: This plot shows the Euclidean (2-norm) distance between the position solution as estimated by our piUSBL system and the position solution estimated by the intersection of range circles from passive LBL. This piUSBL position error referenced against passive LBL is shown for *Platypus*, *Quokka*, and *Wombat* as red, blue, and green solid lines respectively. We also plot the error for dead-reckoning without any acoustic correction from piUSBL as the semi-transparent dashed lines with the same colors.

- Bottom: This plot shows the empirical cumulative distribution function (CDF) of the 2–norm error of the piUSBL positioning solution as referenced against passive LBL; the CDFs for all three vehicles are shown individually as the colored lines, as well as using data from all three vehicles as the black line.

We reproduce these plots for all 6 deployments in these results, to demonstrate the robustness of our piUSBL system for navigation, and to demonstrate exactly how the relative navigation paradigm works in practice using both mode switching and beacon movement. All piUSBL trajectories plotted here were generated using the online, real-time relative position of the beacon as calculated by the AUVs.

6.3.2 Relative Loiter Experiments (10 Sep 2018)

For the two multi-AUV deployments on the 10^{th} of September 2018, the missions were set up with the following relative behaviors:

- **Mode 1**: *Relative Loiter*, with offset of $(0, 0)$ for all vehicles, and radii of 18, 36, and 48 meters for *Platypus*, *Quokka*, and *Wombat* respectively.
- **Mode 2**: *Relative Loiter*, with radii of 18 meters for all vehicles, and offsets of $(7.5, -26)$, $(-7.5, 26)$, and $(22.5, -78)$ for *Platypus*, *Quokka*, and *Wombat* respectively.
- **Mode 3**: *Return and Surface*, with headings of $340°$, $300°$, and $20°$ for *Platypus*, *Quokka*, and *Wombat* respectively.

Data from the first mission is illustrated in Fig. 6.8, which shows that all three vehicles began the mission in *mode 1*, with all three vehicles circling around the beacon at their respective radii; this is apparent in the first two plots of the top row, in which we see that the fleet continued to loiter around the moving beacon as it slowly moved from East to West. At around 1770 s, the operator switched to *mode 2*, which all vehicles successfully detected, switching their behavior into a second *relative loiter*; in this mode, all three vehicles loitered in a circle of the same radius offset from each other in a line, and followed the vehicle as it moved back from West to East. Finally, at around 3200 s, the beacon on the boat was switched off, and the one on the dock was switched to *mode 3*, causing all three vehicles to switch to a *return and surface* behavior, and the fleet returned to the dock.

The experiment is repeated in mission 2, illustrated in Fig. 6.9 – the fleet started out by loitering around the boat-based beacon at different radii as it moved from East to West in *mode 1*; at around 1700 s, the operator switched to *mode 2* and the fleet switched their behavior to the second *relative loiter*, loitering in 18 m circles at different offsets to the beacon as it moved from West to East; at around 1900 s, the fleet was commanded again into *mode 1*, causing them to circle around the boat at different radii; and finally at around 3450 s, the dock-based beacon was switched to *mode 3*, causing the fleet to return to the dock and surface. Comparing the trajectories from the piUSBL system offset by beacon position against the position solution from passive LBL in the lower three plots of both these figures, we see that they qualitatively match fairly well.

The upper plots in Figs. 6.10 and 6.11 illustrate the distance error of piUSBL positioning for each vehicle for the two deployments, as referenced against passive LBL. We see that for both deployments, the position error quickly reduces to around 10 m within the first 60 s or so, as the SMCB particles converge. The error stays around this level for the remainder of the deployments, with occasional large 'spikes' occurring. There are three reasons for these error 'spikes': first, passive LBL is subject to outliers as well, so it may be due to outliers in this reference signal; second, the vehicles breach the surface occasionally, resulting in a GPS fix – when this occurs, the SMCB is reinitialized, and the particles are randomly uniformly

distributed; finally, the multimodal nature of our acoustic range and angle measurement distributions may also occasionally cause the filter to 'spread' its particles over multiple maxima, causing its estimate to be inaccurate. These figures also plot the error in dead-reckoning for each vehicle referenced against passive LBL. We see that these error signals grow unbounded, but occasionally drop close to zero when the vehicle surfaces and receives GPS – these drops in dead-reckoning error occur simultaneously with large increases in piUSBL error, since the SMCB is randomly reinitialized whenever it receives a GPS fix.

The lower plots in Figs. 6.10 and 6.11 illustrate the the empirical CDFs of the distance error for piUSBL positioning as referenced against passive LBL. The CDFs for each AUV for each mission are shown as colored lines, and the CDF using data from all three vehicles is shown with black lines. Examining these plots, we see that the navigational accuracy of *Platypus* is better than that of *Quokka*, which in turn is better than that of *Wombat*. Using data from all three AUVs, we see that the data from these missions indicate 68% of piUSBL position measurements have a 2-norm error with respect to passive LBL of less than 9 m. These error statistics agree well with the accuracy analysis we performed previously in the system evaluation chapter, chapter 5, section 5.6, which showed that the piUSBL system achieved a 75^{th} percentile distance error in the range of 8 m to 14 m, when inertial measurements had initial biases of 0.1–0.3 $\mathrm{m\,s^{-1}}$ in speed and 1°–3° in heading. The superior navigational performance of *Platypus* is likely due to its better heading precision, as previously shown in chapter 5, Fig. 5.20; this plot illustrates that the MEMS IMU heading performance of *Platypus* is better than that of *Quokka*, and also shows that *Wombat* has the worst heading performance of all three vehicles. Although all three vehicles are equipped with the same MEMS IMU type, the effects of sensor calibration have likely caused this inconsistency in performance across the fleet. Heading accuracy has a significant effect on the performance of piUSBL positioning, as we demonstrated in section 5.6 of the previous chapter.

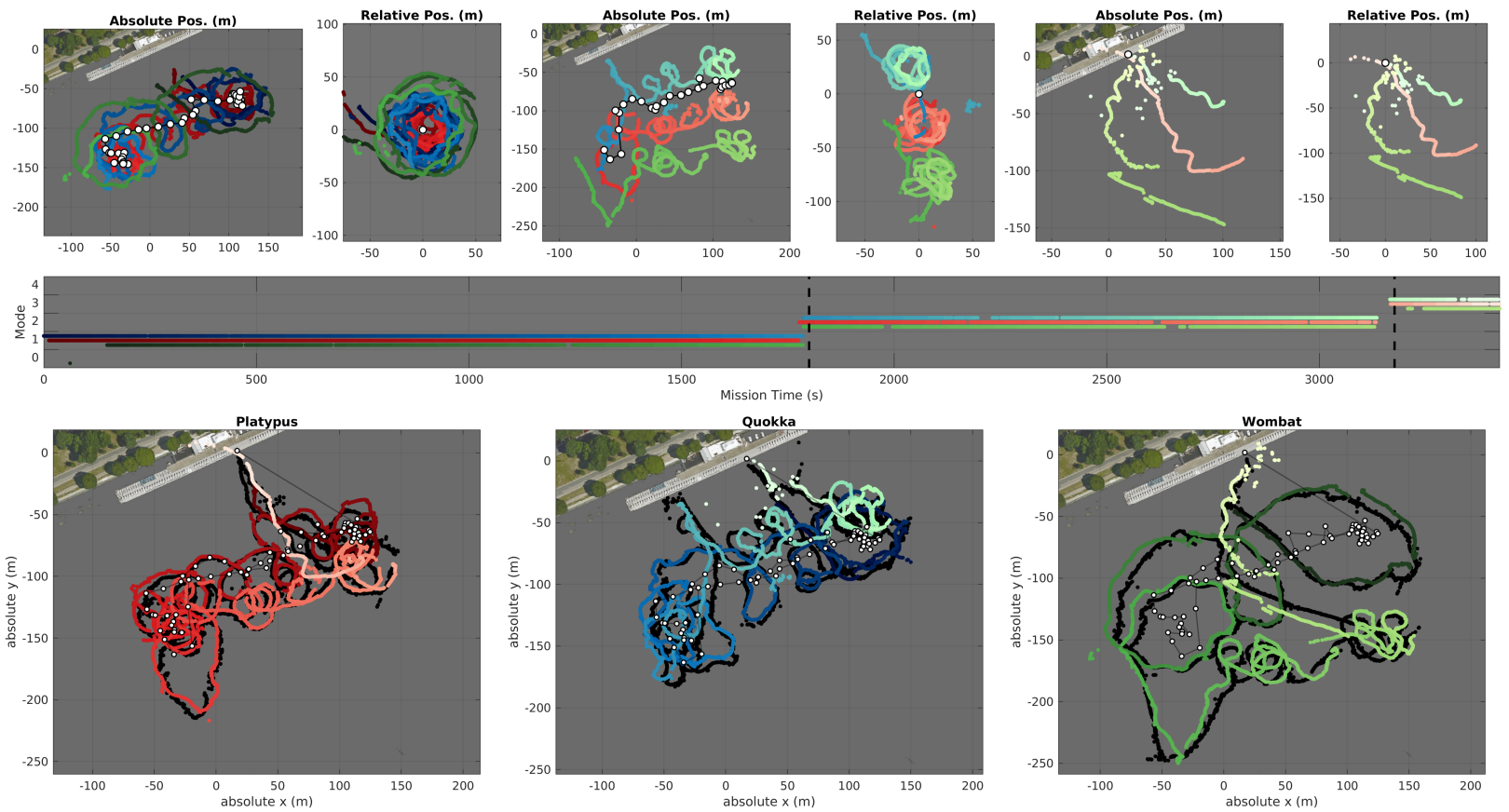

Figure 6.8: Mission 1 September 10 2018, plots of trajectories of the SandShark fleet operating under the piUSBL relative navigation operating paradigm – *Platypus* in red, *Quokka* in blue, and *Wombat* in green, with darker shades indicating earlier mission times, and lighter shades indicating later mission times. *Middle*: plot indicating the mode detected by each vehicle at each moment during the entire mission. *Top*: first two plots show the absolute and relative (beacon-centric) trajectories of all vehicles up to the mission time indicated by the first dashed black line in the middle plot, with the vehicles performing a *relative loiter* behavior centered at the beacon with differing radii; the second two plots show the trajectories between the first and second dashed black lines in the middle plot, with the vehicles performing a *relative loiter* of the same radius at different offsets from the beacon; the last two plots show the trajectories between the last dashed black line and the end of the mission, with the vehicles performing a *return and surface* behavior to return to the dock. *Bottom*: plots illustrating the trajectories of *Platypus*, *Quokka*, and *Wombat* over the entire mission as estimated by piUSBL in color, and by passive LBL in black.

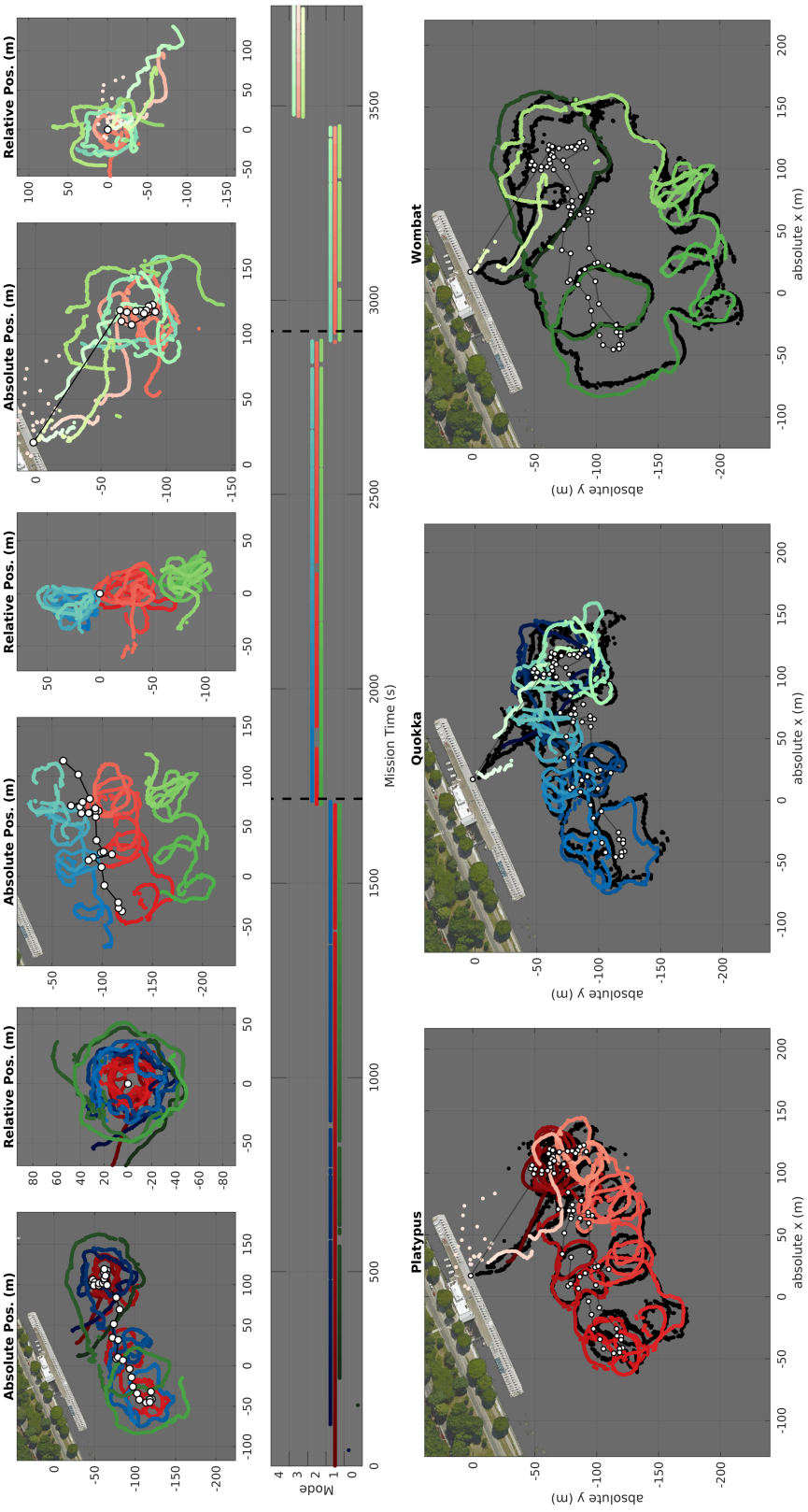

Figure 6.9: Mission 2 September 10 2018, plots of trajectories of the SandShark fleet operating under the piUSBL relative navigation operating paradigm – *Platypus* in red, *Quokka* in blue, and *Wombat* in green, with darker shades indicating earlier mission times, and lighter shades indicating later mission times. *Middle:* plot indicating the mode detected by each vehicle at each moment during the entire mission. *Top:* first two plots show the absolute and relative (beacon-centric) trajectories of all vehicles up to the mission time indicated by the first dashed black line in the middle plot, with the vehicles performing a *relative loiter* behavior centered at the beacon with differing radii; the second two plots show the trajectories between the first and second dashed black lines in the middle plot, with the vehicles performing a *relative loiter* of the same radius at different offsets from the beacon; the last two plots show the trajectories between the last dashed black line and the end of the mission, with the vehicles performing a *return and surface* behavior to return to the dock. *Bottom:* plots illustrating the trajectories of *Platypus*, *Quokka*, and *Wombat* over the entire mission as estimated by piUSBL in color, and by passive LBL in black.

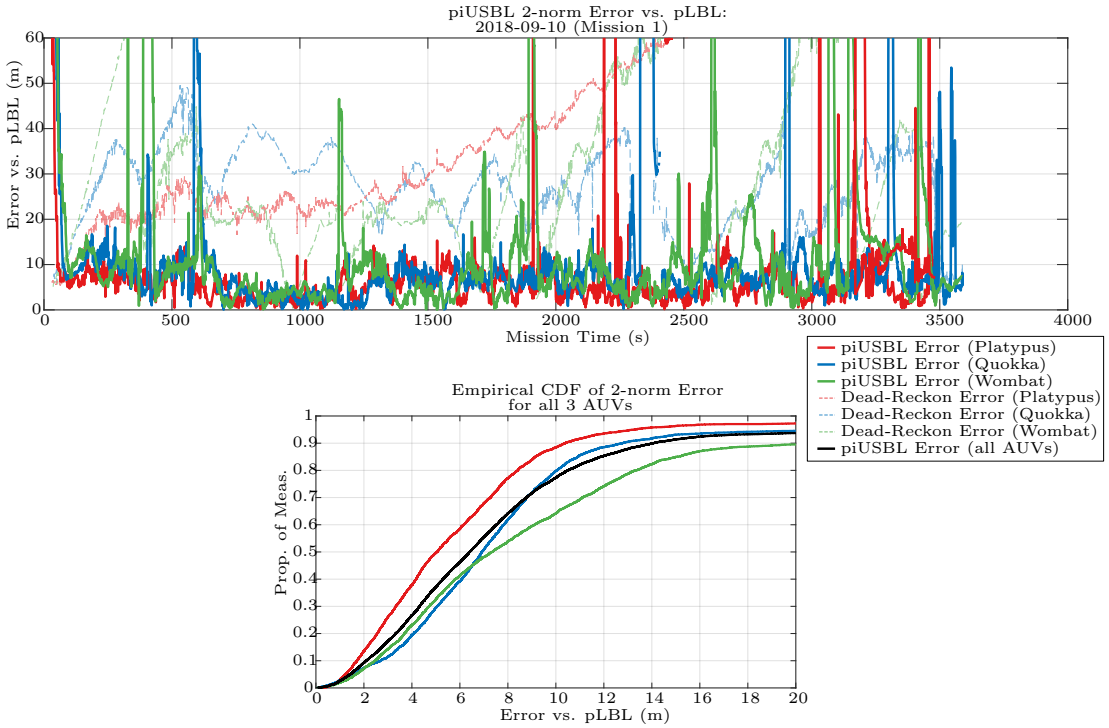

Figure 6.10: Mission 1 September 10 2018, plots of 2-norm distance error of piUSBL navigation referenced against passive LBL – *Bottom*: empirical CDFs of piUSBL error.

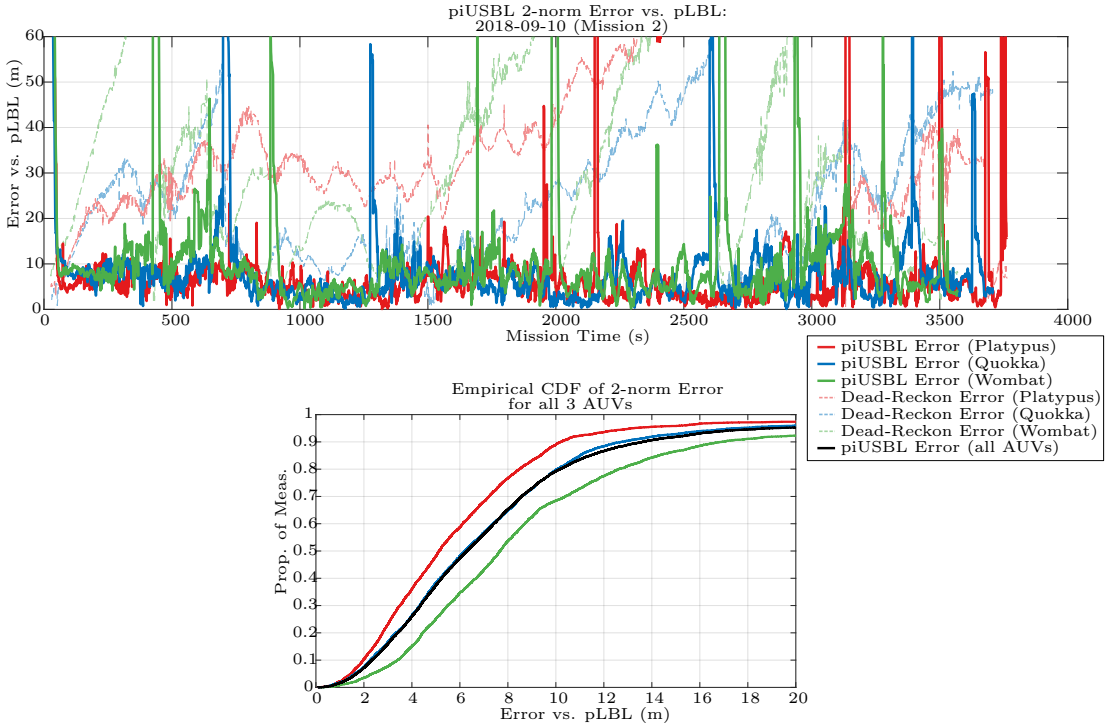

Figure 6.11: Mission 2 September 10 2018, plots of 2-norm distance error of piUSBL navigation referenced against passive LBL – *Bottom*: empirical CDFs of piUSBL error.

6.3.3 Relative Line and Relative Loiter Experiments (12 Sep 2018)

For the two multi-AUV deployments on the 12^{th} of September 2018, the missions were set up with the following relative behaviors:

- **Mode 1**: *Relative Line*, with offsets of $(-14.1, -5.1)$, $(18.8, 6.8)$, and $(-37.6, -13.7)$ for *Platypus*, *Quokka*, and *Wombat* respectively, and line headings of 160°, line lengths of 120 meters and a buffer distance of 10 meters for all vehicles.
- **Mode 2**: *Relative Loiter*, with radii of 15 meters for all vehicles, and offsets of $(7.5, -26)$, $(-7.5, 26)$, and $(22.5, -78)$ for *Platypus*, *Quokka*, and *Wombat* respectively.
- **Mode 3**: *Return and Surface*, with line headings of 340°, 300°, and 20° for *Platypus*, *Quokka*, and *Wombat* respectively.

The results of the first mission are shown in Fig. 6.12, which illustrate that all three AUVs began the mission in *mode 1*, with the three vehicles performing a *relative line* behavior in order to form parallel tracks with one another, at a distance of 35 m between *Platypus* and *Quokka*, and a distance of 25 m between *Platypus* and *Wombat*. As the boat-based beacon was shifted from the North-East to the South-West, the three vehicles maintained these parallel tracks as they shifted their absolute position along with it; this allowed the fleet to survey a large area, but from their point of view, they simply continued to perform the same back and forth track in the beacon-centric frame of reference, as clearly shown in the first and second relative position plots. The vehicles continued to perform this behavior until around 2250 s, where the operator switched to *mode 2*, and the fleet began to perform the *relative loiter* behavior at different offsets from the beacon. As the beacon was moved from the West to the East, the fleet followed along with it while continuing to perform their circular patterns of constant radius, as shown in the third relative position plot. The dock-based beacon then switched to *mode 3* to request that the vehicles return to the dock, but only *Quokka* was able to detect and perform the *return and surface* behavior. The other two vehicles were manually retrieved from the water, having been unable to switch into this behavior due to a bug in the behavioral state machine. After fixing this bug, the fleet was deployed again for a second mission, shown in Fig. 6.13 – the vehicles performed a similar deployment as the first mission, but this time all three AUVs were able to detect *mode 3* and return to the dock. Again, comparison of the piUSBL navigation solution to the passive LBL position solution shows good qualitative agreement.

The upper plots of Figs. 6.14 and 6.15 show the piUSBL position error for each AUV against passive LBL, and they show similar convergence properties and error levels as in the September 10 missions. The lower plots illustrate the CDFs of these errors, as well as for the error using data from all vehicles combined – these plots are very consistent with the corresponding CDF plots of the September 10 missions, with 68% of piUSBL error measurements falling below 9 m in mission 1, and below 8 m in mission 2 (using data from all vehicles).

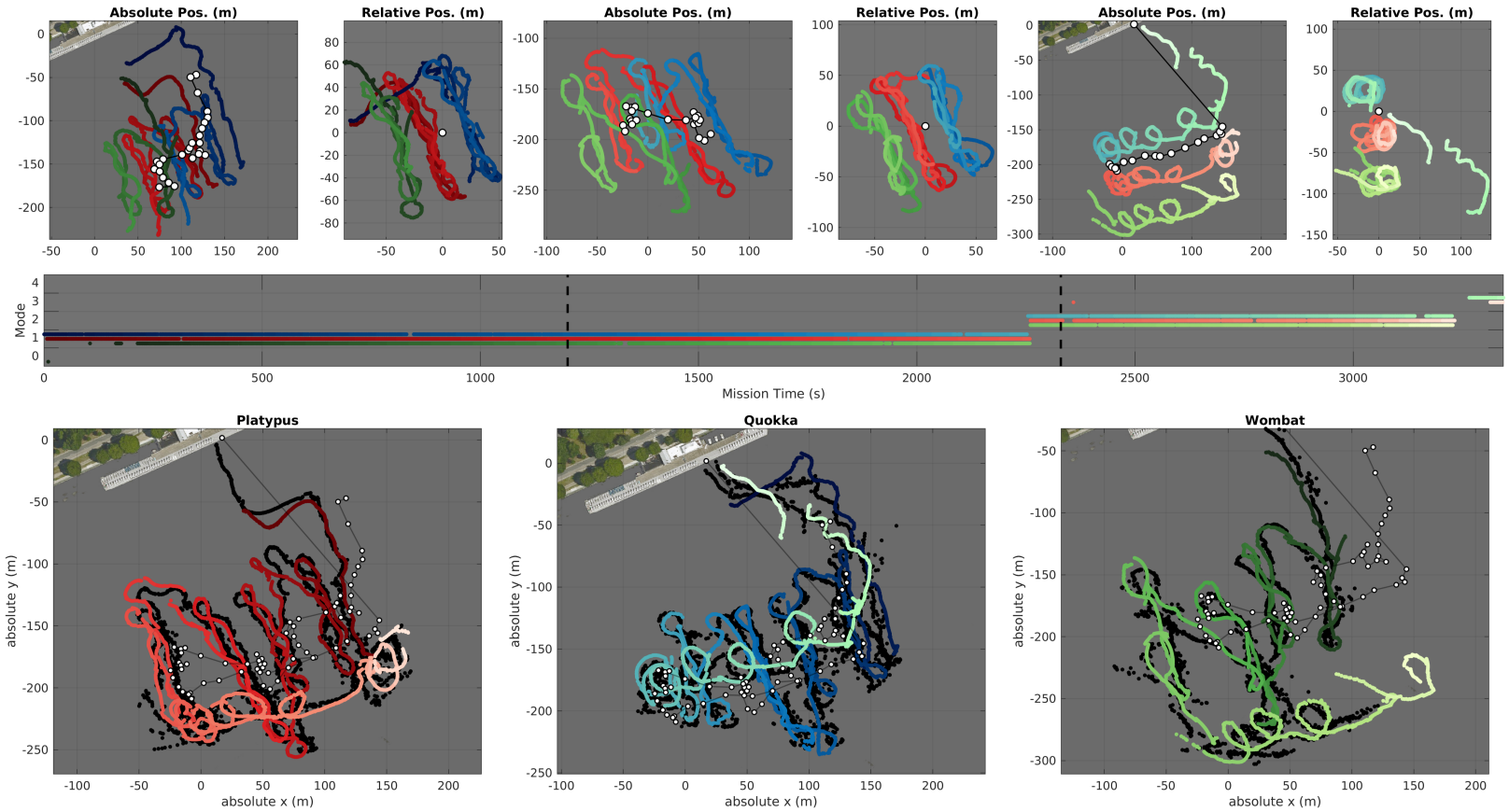

Figure 6.12: Mission 1 September 12 2018, plots of trajectories of the SandShark fleet operating under the piUSBL relative navigation operating paradigm – *Platypus* in red, *Quokka* in blue, and *Wombat* in green, with darker shades indicating earlier mission times, and lighter shades indicating later mission times. *Middle*: plot indicating the mode detected by each vehicle at each moment during the entire mission. *Top*: first two plots show the absolute and relative (beacon-centric) trajectories of all vehicles up to the mission time indicated by the first dashed black line in the middle plot, with the vehicles performing a *relative line* behavior offset from the beacon to perform parallel tracks; the second two plots show the trajectories between the first and second dashed black lines in the middle plot, with the vehicles still performing a *relative line*; the last two plots show the trajectories between the last dashed black line and the end of the mission, with the vehicles performing a *relative loiter* behavior of the same radius offset from the beacon. *Bottom*: plots illustrating the trajectories of *Platypus*, *Quokka*, and *Wombat* over the entire mission as estimated by piUSBL in color, and by passive LBL in black.

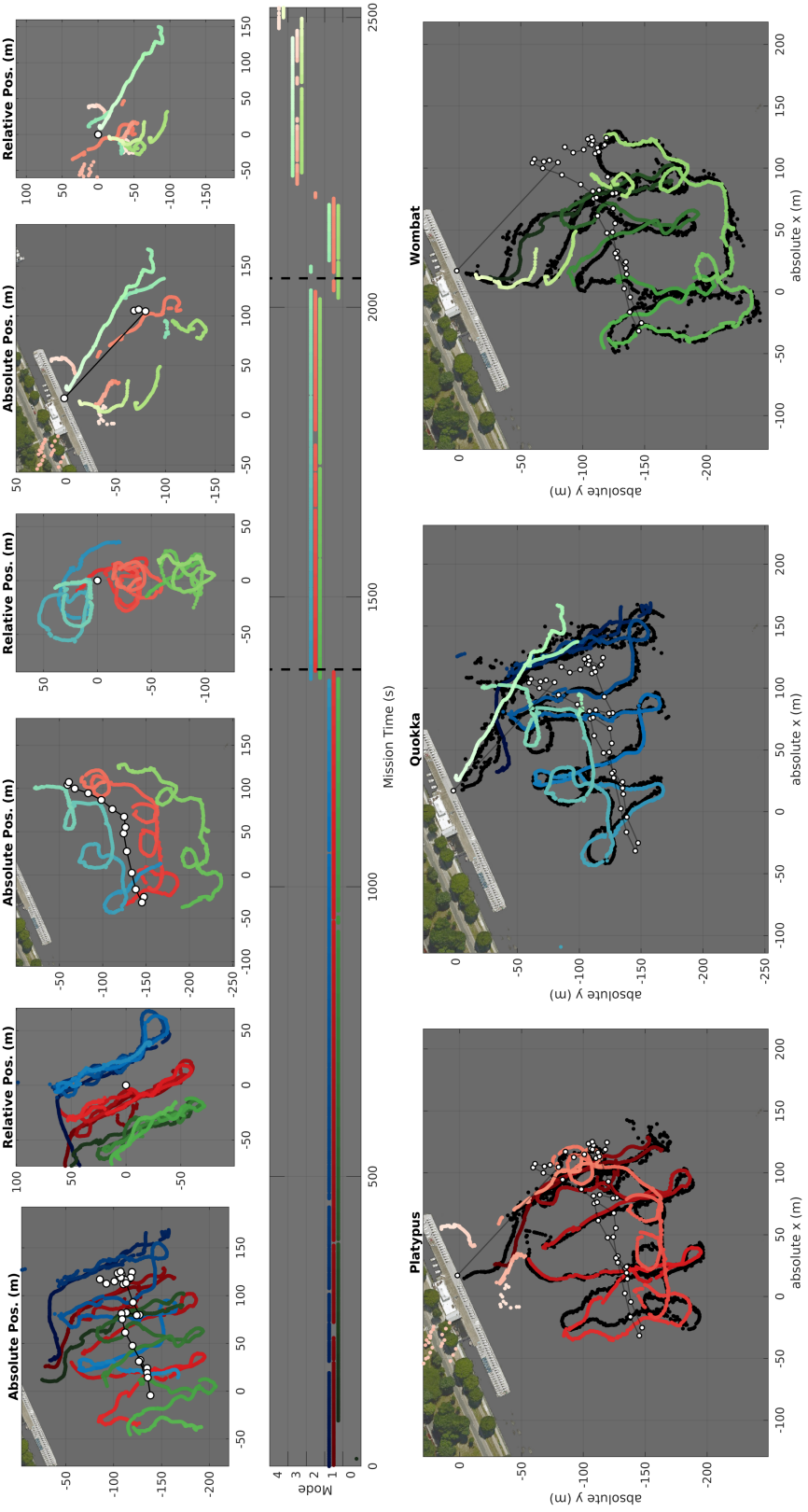

Figure 6.13: Mission 2 September 12 2018, plots of trajectories of the SandShark fleet operating under the piUSBL relative navigation operating paradigm – *Platypus* in red, *Quokka* in blue, and *Wombat* in green, with darker shades indicating earlier mission times, and lighter shades indicating later mission times. *Middle:* plot indicating the mode detected by each vehicle at each moment during the entire mission. *Top:* first two plots show the absolute and relative (beacon-centric) trajectories of all vehicles up to the mission time indicated by the first dashed black line in the middle plot, with the vehicles performing a *relative line* behavior offset from the beacon to perform parallel tracks; the second two plots show the trajectories between the first and second dashed black lines in the middle plot, with the vehicles performing a *relative loiter* of the same radius offset from the beacon; the last two plots show the trajectories between the last dashed black line and the end of the mission, with the vehicles performing a *return and surface* behavior to return to the dock. *Bottom:* plots illustrating the trajectories of *Platypus*, *Quokka*, and *Wombat* over the entire mission as estimated by piUSBL in color, and by passive LBL in black.

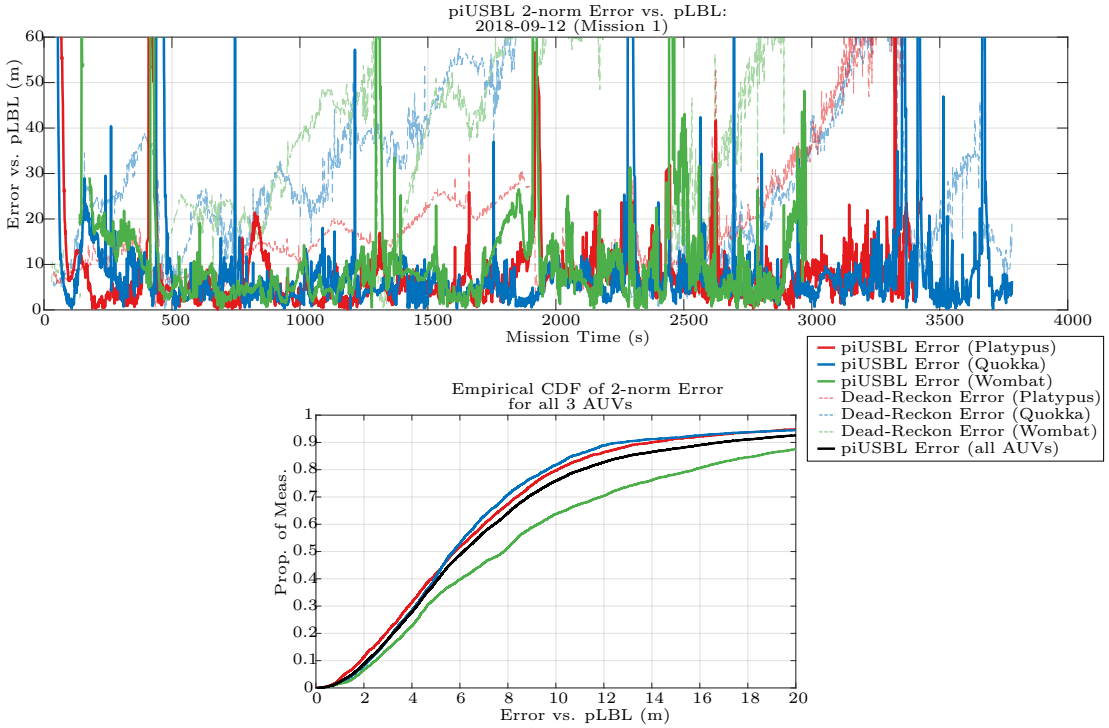

Figure 6.14: Mission 1 September 12 2018, plots of 2-norm distance error of piUSBL navigation referenced against passive LBL – *Bottom*: empirical CDFs of piUSBL error.

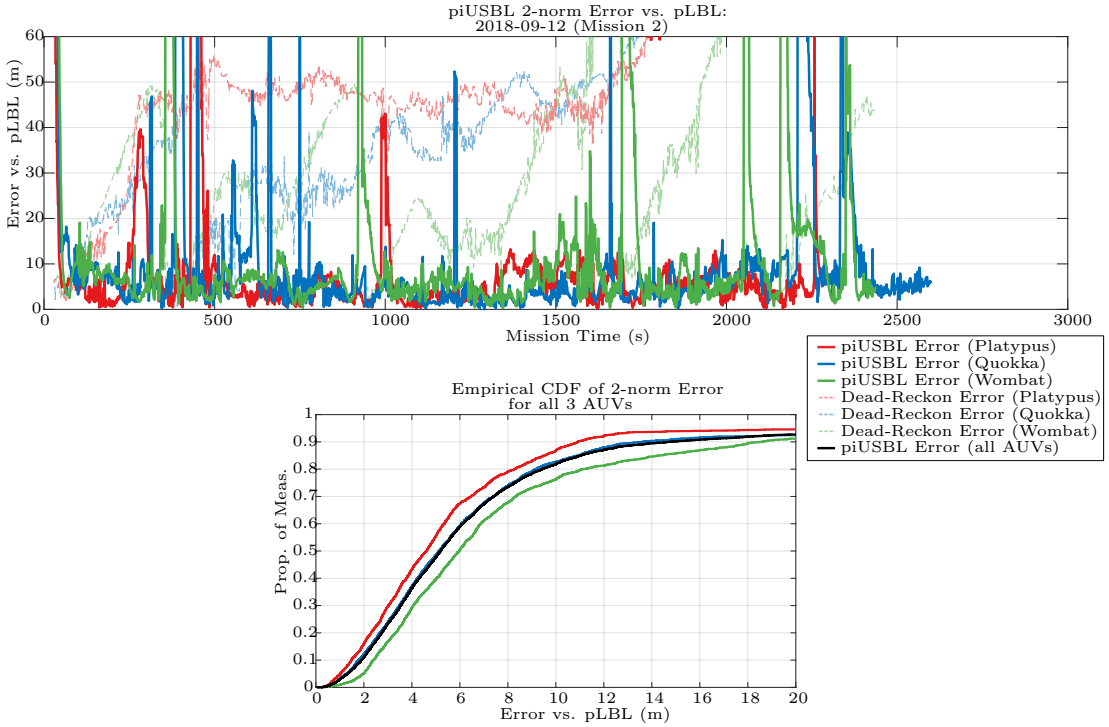

Figure 6.15: Mission 2 September 12 2018, plots of 2-norm distance error of piUSBL navigation referenced against passive LBL – *Bottom*: empirical CDFs of piUSBL error.

6.3.4 Relative Line and Offset-Follow Experiments (14 Sep 2018)

For the two multi-AUV deployments on the 14^{th} of September 2018, the missions were set up with the following relative behaviors:

- **Mode 1**: *Relative Line*, with offsets of $(-14.1, -5.1)$, $(18.8, 6.8)$, and $(-37.6, -13.7)$ for *Platypus*, *Quokka*, and *Wombat* respectively, and line headings of 160°, line lengths of 120 meters and a buffer distance of 14 meters for all vehicles.
- **Mode 2**: *Offset-Follow*, with and offsets of $(7.5, -26)$, $(-7.5, 26)$, and $(22.5, -78)$ for *Platypus*, *Quokka*, and *Wombat* respectively, and a buffer radius of 15 meters and depth ceiling of 1 meter for all vehicles.
- **Mode 3**: *Return and Surface*, with line headings of 340°, 300°, and 20° for *Platypus*, *Quokka*, and *Wombat* respectively.

Data from the first mission is illustrated in Fig. 6.16. The three AUVs detected the initially selected *mode 1* broadcast by the boat-based beacon, and the fleet performed the *relative line* behavior, with a spacing of 35 m and 25 m between *Platypus* and *Quokka*, and *Wombat*, forming parallel line transects 160 m long at a heading of 160°. As the beacon slowly moved East-West, it stopped at three different positions, allowing the fleet to survey a large area – the tracks formed by this section of the mission represent a near-ideal performance of the sidescan survey multi-AUV concept shown in Fig. 6.7. At the Western-most end of the dock, at around 2000 s, the operator selected *mode 2*, causing the vehicles to switch into the *offset-follow* behavior, where *Platypus* and *Quokka* attempted to maintain position at points 27 m away from the boat to the South and North respectively; *Wombat* tried to maintain position 81 m in the South direction from the beacon. As a result, the three AUVs attempted to maintain a *line formation*, with a spacing between each other of 54 m. As the boat-based beacon moved in a large 'P' maneuver, we see that the fleet was able to successfully formation-keep, maintaining this line throughout the maneuver – this result represents one of the very few times that multi-AUV formation keeping has been demonstrated experimentally. At approximately 3400 s, the fleet is commanded to *mode 3* by the dock-based beacon, causing the vehicles to *return and surface* dockside. Notice the LBL outliers for *Wombat*.

Data from the second mission is shown in Fig. 6.17, where the fleet performed a similar deployment. *Relative lines* were again performed at three different points using the East-West movement of the boat-based beacon, and the *offset-follow* behavior was selected using *mode 2* at around 1800 s; the fleet was again able to maintain formation while the boat performed a shallow 'U' maneuver. The fleet then switched back into the *relative line* behavior at around 2550 s, and into the *return and surface* behavior at around 2900 s to return dockside.

The piUSBL position error plots against passive LBL of Figs. 6.18 and 6.19 show a similar level of accuracy as the missions from the previous days; performance is slightly improved as evidenced by the lower CDF plots, with 68% of piUSBL error measurements falling below 7.5 m in mission 1, and below 7 m in mission 2, when using data from all vehicles.

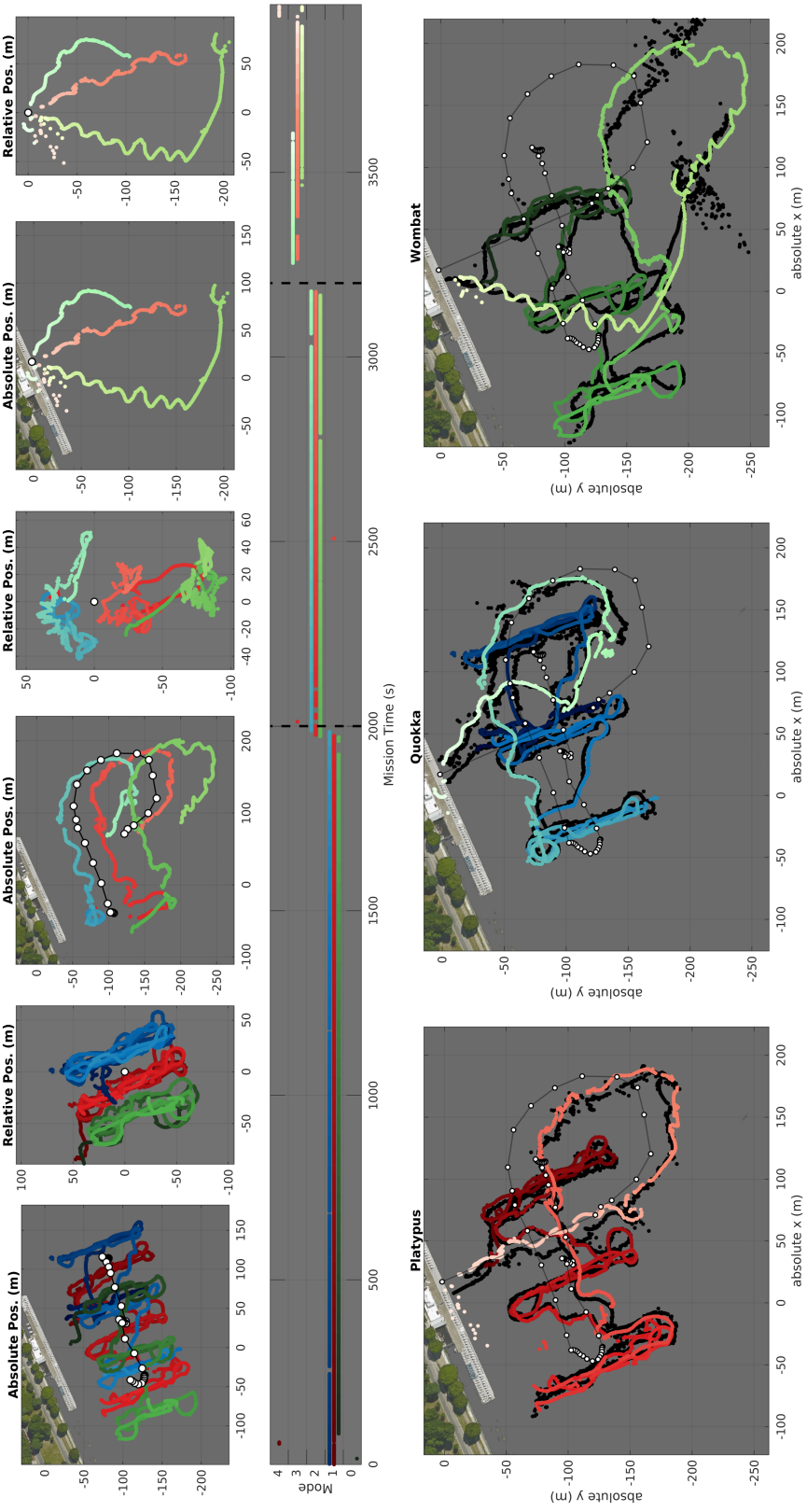

Figure 6.16: Mission 1 September 14 2018, plots of trajectories of the SandShark fleet operating under the piUSBL relative navigation operating paradigm – *Platypus* in red, *Quokka* in blue, and *Wombat* in green, with darker shades indicating earlier mission times, and lighter shades indicating later mission times. *Middle:* plot indicating the mode detected by each vehicle at each moment during the entire mission. *Top:* first two plots show the absolute and relative (beacon-centric) trajectories of all vehicles up to the mission time indicated by the first dashed black line in the middle plot, with the vehicles performing a *relative line* behavior offset from the beacon to perform parallel tracks; the second two plots show the trajectories between the first and second dashed black lines in the middle plot, with the vehicles performing an *offset-follow* behavior to follow the beacon in a line formation; the last two plots show the trajectories between the last dashed black line and the end of the mission, with the vehicles performing a *return and surface* behavior to return to the dock. *Bottom:* plots illustrating the trajectories of *Platypus*, *Quokka*, and *Wombat* over the entire mission as estimated by piUSBL in color, and by passive LBL in black.

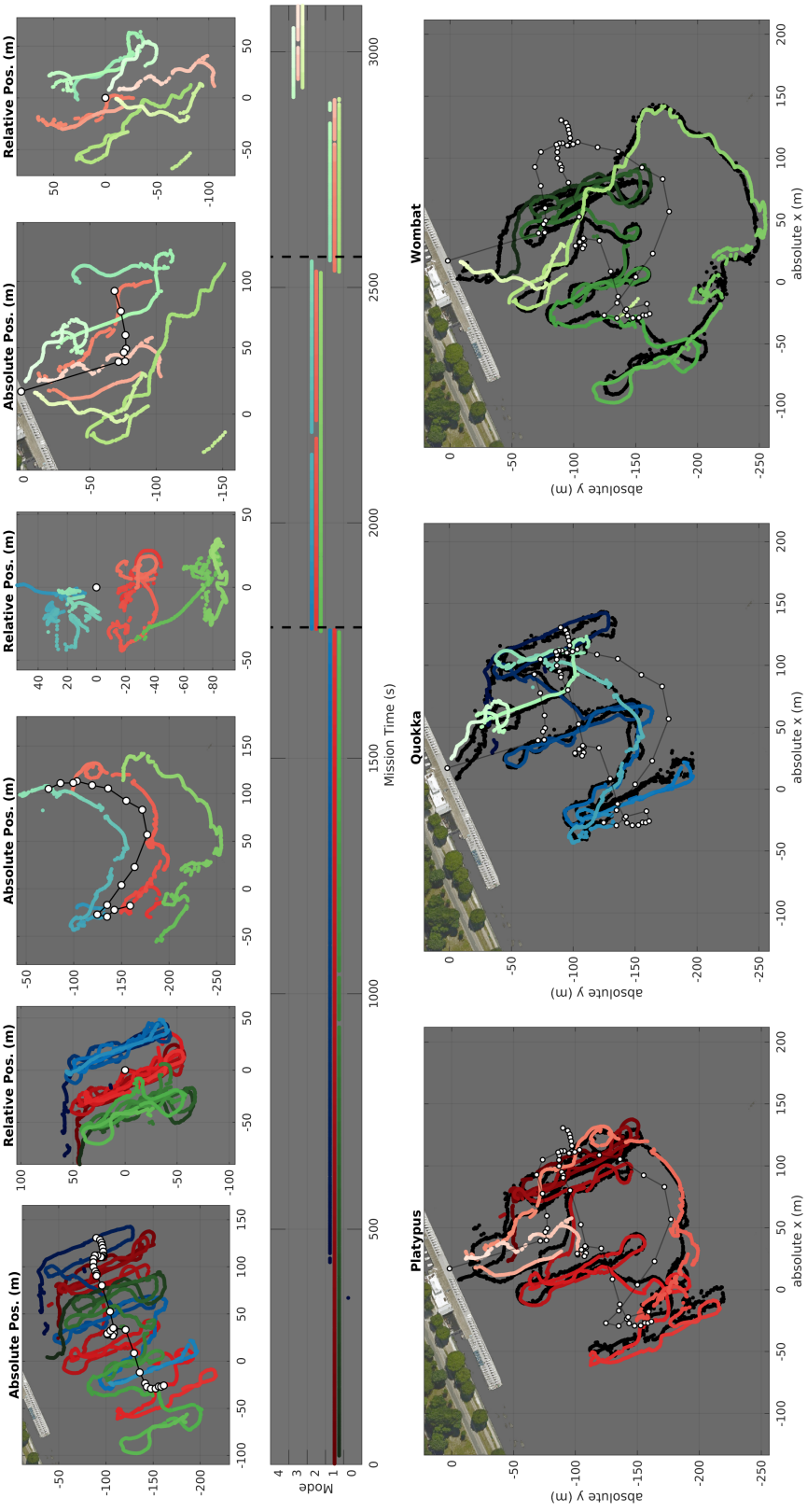

Figure 6.17: Mission 2 September 14 2018, plots of trajectories of the SandShark fleet operating under the piUSBL relative navigation operating paradigm – *Platypus* in red, *Quokka* in blue, and *Wombat* in green, with darker shades indicating earlier mission times, and lighter shades indicating later mission times. *Middle:* plot indicating the mode detected by each vehicle at each moment during the entire mission. *Top:* first two plots show the absolute and relative (beacon-centric) trajectories of all vehicles up to the mission time indicated by the first dashed black line in the middle plot, with the vehicles performing a *relative line* behavior offset from the beacon to perform parallel tracks; the second two plots show the trajectories between the first and second dashed black lines in the middle plot, with the vehicles performing an *offset-follow* behavior to follow the beacon in a line formation; the last two plots show the trajectories between the last dashed black line and the end of the mission, with the vehicles performing a *return and surface* behavior to return to the dock. *Bottom:* plots illustrating the trajectories of *Platypus*, *Quokka*, and *Wombat* over the entire mission as estimated by piUSBL in color, and by passive LBL in black.

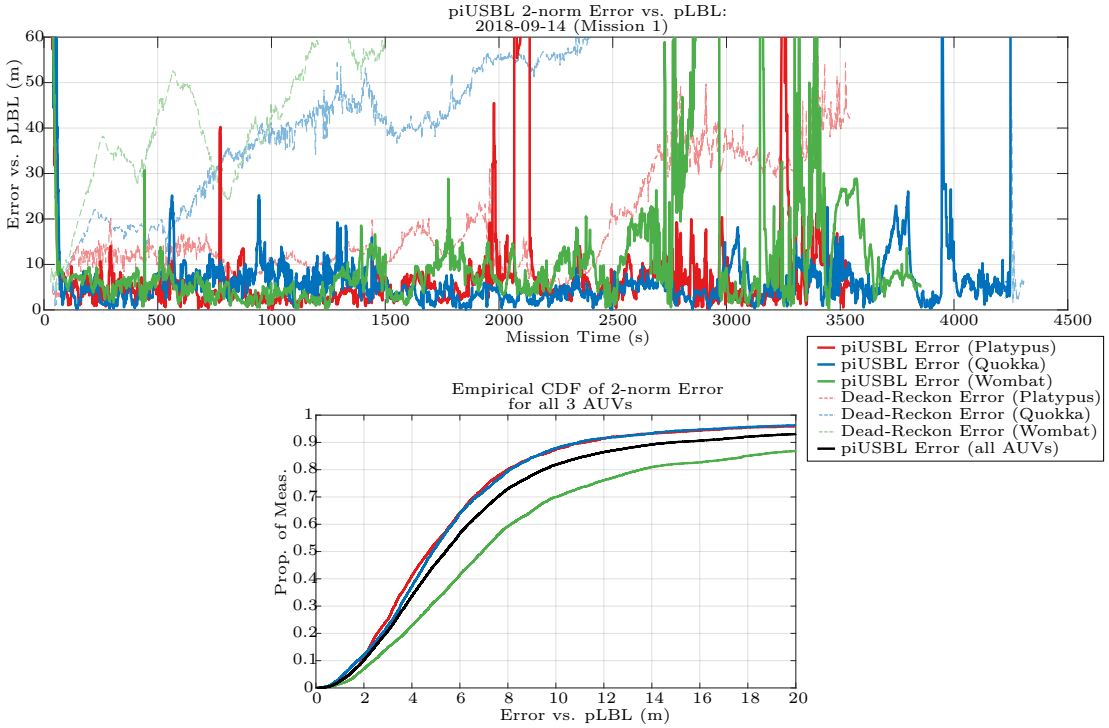

Figure 6.18: Mission 1 September 14 2018, plots of 2-norm distance error of piUSBL navigation referenced against passive LBL – *Bottom*: empirical CDFs of piUSBL error.

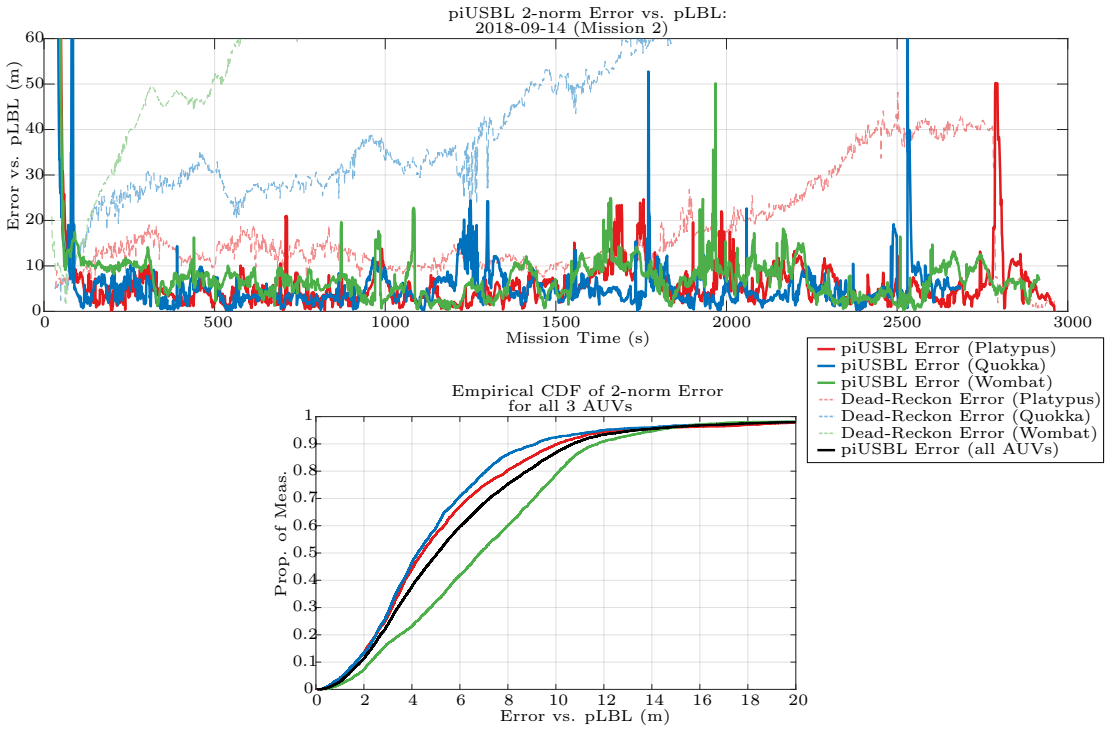

Figure 6.19: Mission 2 September 14 2018, plots of 2-norm distance error of piUSBL navigation referenced against passive LBL – *Bottom*: empirical CDFs of piUSBL error.

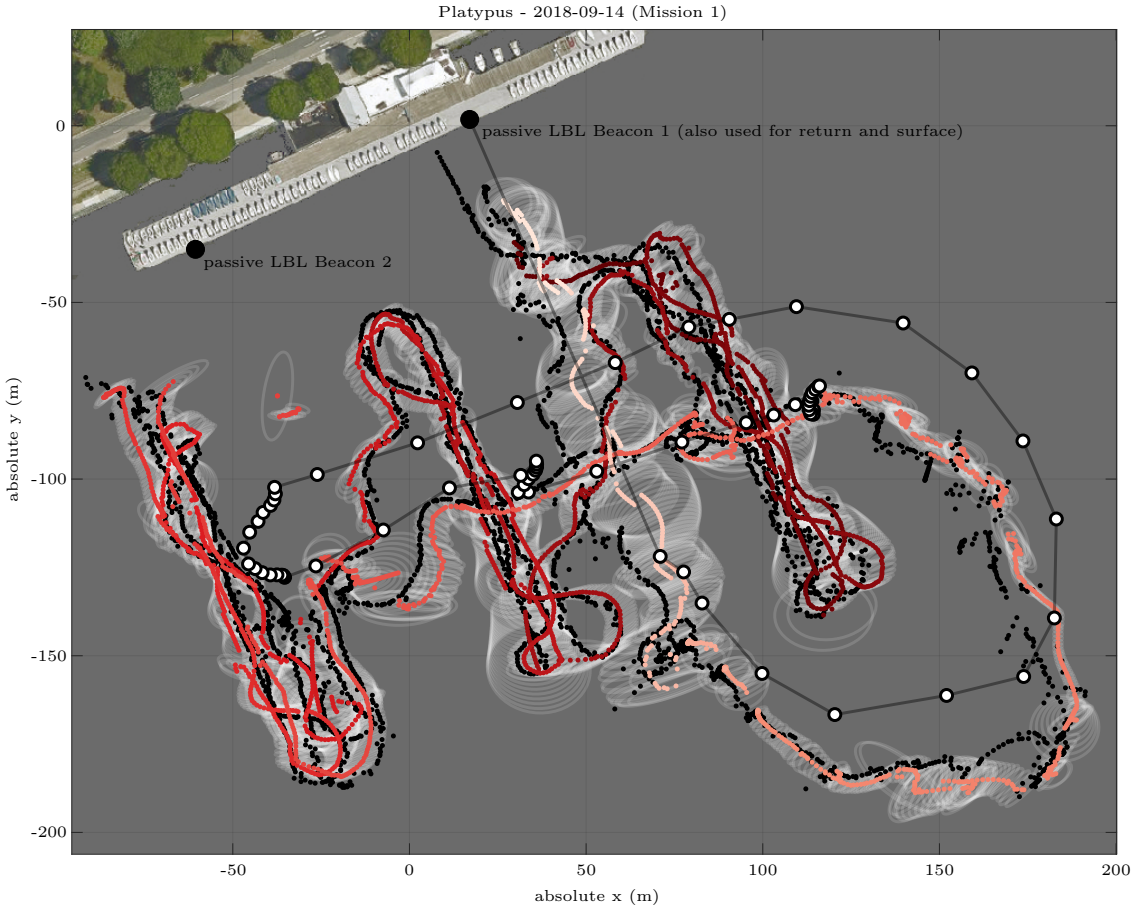

Figure 6.20: Trajectory of *Platypus* during mission 1 on September 14 2018, as estimated by piUSBL and passive LBL – the piUSBL trajectory estimate is shown as the shaded red dots, with darker-to-lighter shades indicated an increase in mission time; passive LBL position estimates are shown as black dots, which are estimated using the intersection of range circles from the two LBL beacons shown as black dots on the dock; the position of the motorboat and beacon over time is shown as the white dots connected by black lines; and 1σ covariance ellipses as estimated using multivariate Gaussian fits to the particles in the sequential Monte-Carlo beamformer (SMCB) are shown along the entire trajectory using semi-transparent white ellipses. Only piUSBL positions with a 1σ value less than 15 m in both the minor and major axes are shown.

To more clearly illustrate the piUSBL relative navigation operating paradigm, we reproduce an enlarged copy of the entire trajectory of *Platypus* from mission 1 on the 14^{th} of September 2018 (shown to the lower left of Fig. 6.16) in Fig. 6.20. In terms of vehicle behavior, mode-switching, and vehicle command-and-control, we clearly see that *Platypus* started the mission in *mode 1*, maintaining a 120 m long transect line offset from the beacon to the South-West in the beacon-centric frame of reference; as the beacon moved to the South-West, stopping at two different points at $(30, -100)$ m and $(-30, -125)$ m, *Platypus automatically* shifted its

transect line with it, following the beacon and always attempting to maintain its track relative to the beacon. The vehicle clearly detected the operator-commanded switch to *mode 2*, which triggered it to change to the *offset-follow behavior*, in which it always attempted to station-keep at a point offset from the beacon by 27 m in a South-Easterly direction; as the boat performed a large 'P-shaped' maneuver, *Platypus* was clearly able to maintain this constant offset position relative to the beacon, reproducing the boat maneuver faithfully at its offset. Finally, the dock-based beacon was used to command *mode 3*, and the vehicle successfully detected this request to return to the dock, approaching the beacon at a constant heading of 340° and surfacing 15 m from the dock using the *return and surface* behavior. This set of behaviors commanded by the operator, and the control of the vehicle in the absolute frame of reference through the movement of the beacon by the operator, provide a highly convincing demonstration of the utility and power of the *piUSBL relative navigation operating paradigm* as a method of coherently managing a fleet of AUVs using very few operators – this approach provides an intuitive method of simultaneously navigating, commanding, and controlling a large fleet of vehicles.

In terms of piUSBL navigational accuracy, Fig. 6.20 also plots the 1σ covariance ellipses from a multivariate Gaussian fit to the particles in the SMCB. Note that we only show the ellipse and associated piUSBL position estimate if the 1σ deviation is less than 15 m in both axes of the ellipse, since this represents a convergence of the filter to a strong confidence of the relative position of the piUSBL beacon. The figure also plots the position of *Platypus* as estimated by the range-only intersection of passive LBL using the two additional dockside beacons (as black dots) – this passive LBL estimate is used to validate the accuracy of piUSBL, since we demonstrated in chapter 5 that it has an accuracy comparable to that of a consumer GPS receiver. We see that for much of the mission, the estimate from piUSBL and the estimate from passive LBL shows very good agreement, with the passive LBL estimate often falling within the 1σ covariance ellipse of the piUSBL estimate. The difference in estimates are due to a multitude of factors: error in the piUSBL position solution itself; error in the GPS position estimate of the beacon; and error in the passive LBL solution; all these factors combine to contribute to the disagreement between the two solutions.

Finally, let us take a closer look at Fig. 6.19 – during this mission (the second mission on September 14), all AUVs essentially remained underwater for the entire duration of the deployment. As a result, this plot provides us with a clear insight of how the position error evolves with time, since the piUSBL filter is never reinitialized due to surfacing, and the dead-reckon solution never resets to GPS. We clearly see that the piUSBL navigation solution for all three vehicles quickly falls below 10 m within the first 300 s of the mission, and remains more or less below this level for the entire deployment – the position error is clearly bounded. On the other hand, we can see just how quickly the position error from dead-reckoning grows without bound, and how much this rate of growth varies across all three vehicles. *Platypus*

dead-reckoning error grows the slowest, at a maximum rate of approximately 4 $\mathrm{cm\,s^{-1}}$; dead-reckoning for *Quokka* grows more rapidly at a maximum rate just above 4 $\mathrm{cm\,s^{-1}}$; and finally, the dead-reckoning performance of *Quokka* is the worst, at a maximum rate of approximately 8 $\mathrm{cm\,s^{-1}}$. This means that after only 10 minutes of operation using dead-reckoning for navigation, the error in position can grow to about 36 m. With each vehicle traveling at a nominal 1 $\mathrm{m\,s^{-1}}$ speed, these error rates represent a navigational drift in dead-reckoning of 4%–8% of distance traveled.

6.3.5 Position Error Referenced Against Passive LBL

To quantify the accuracy of our piUSBL system when using the relative navigation operating paradigm, we plot the position error using the passive LBL estimate as a 'ground-truth' reference. These statistics for *Platypus*, *Quokka*, and *Wombat* are shown in Fig. 6.21, which show these errors along with their Gaussian fits, as well as the mean and standard deviations associated with these fits; these statistics are also summarized in table 6.5. These accuracies are slightly inferior to those demonstrated using the WAM-V ASV in chapter 5, sections 5.4 and 5.6, but are still impressive – the lower accuracy is to be expected given the additional complication and uncertainty associated with a moving beacon, as well as the stronger local acoustic interactions that effect the angle measurements for the SandShark AUVs as shown in section 5.9.

The empirical CDFs of the 2-norm error in piUSBL position for each AUV is shown in Fig. 6.22, along with the empirical CDFs estimated using dead-reckoning without using piUSBL acoustic corrections. We see from these plots that *Platypus* outperforms *Quokka*, which in turn outperforms *Wombat* – it is interesting to note that the dead-reckoning performance follows this same trend; this suggests that the performance of dead-reckoning has a significant impact on the performance of piUSBL. This intuitively makes sense, since error in dead-reckoning is largely due to error in heading and speed-over-ground estimates, both of which have an impact on piUSBL accuracy (heading error especially), as we demonstrated in chapter 5, section 5.6. This suggests that regardless of inertial navigation accuracy, our piUSBL system can be used to bound the growth in error of dead-reckoning and significantly improve navigational performance – including that of both high-end conventional AUVs, as well as low-cost, miniature AUVs. Statistics related to these CDF plots are also listed in table 6.5.

By using all the data from all three AUVs over all six missions presented here, we can generate the error statistics shown in Fig. 6.23. The plot on the left shows the position error of piUSBL against passive LBL using all this data, with the Gaussian fit standard deviations listed in table 6.5. These statistics suggest an accuracy of $\mu \pm \sigma = (0.294, 1.874) \pm (6.99, 6.48)$ m in x and y for our piUSBL system operating under relative navigation across all vehicles. The

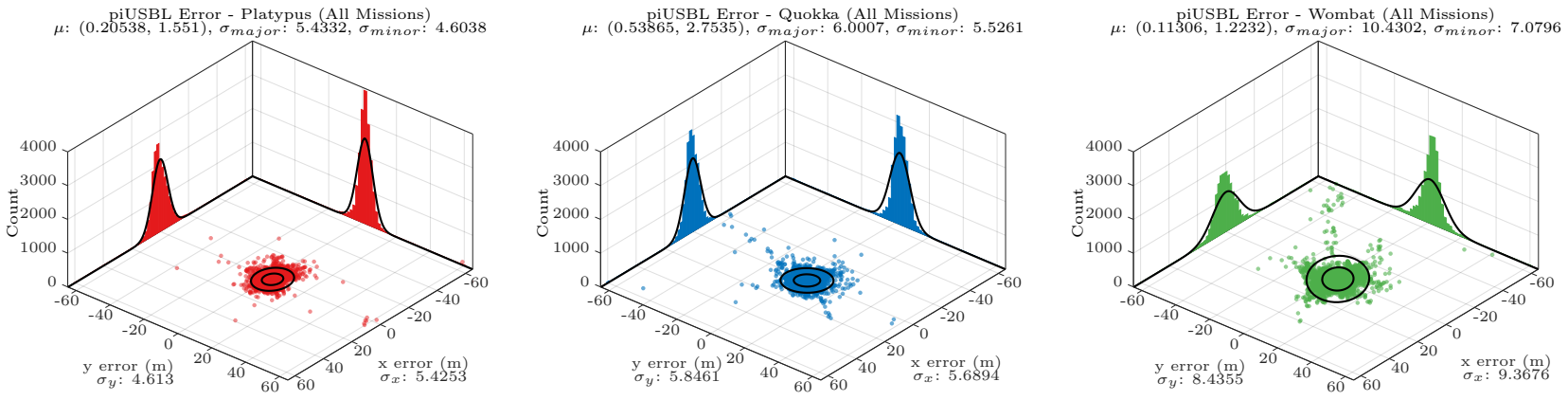

Figure 6.21: The piUSBL position error referenced against passive LBL, for *Platypus*, *Quokka*, and *Wombat*, using data from all six multi-AUV deployments – piUSBL errors are plotted as dots in the x-y plane, with 1σ and 2σ covariance ellipses from multivariate Gaussian fitting; the marginal error distributions in x and y are plotted as histograms on the 'walls' of the plot, with Gaussian fits. *Left*: *Platypus* piUSBL position error. *Middle*: *Quokka* piUSBL position error. *Right*: *Wombat* piUSBL position error.

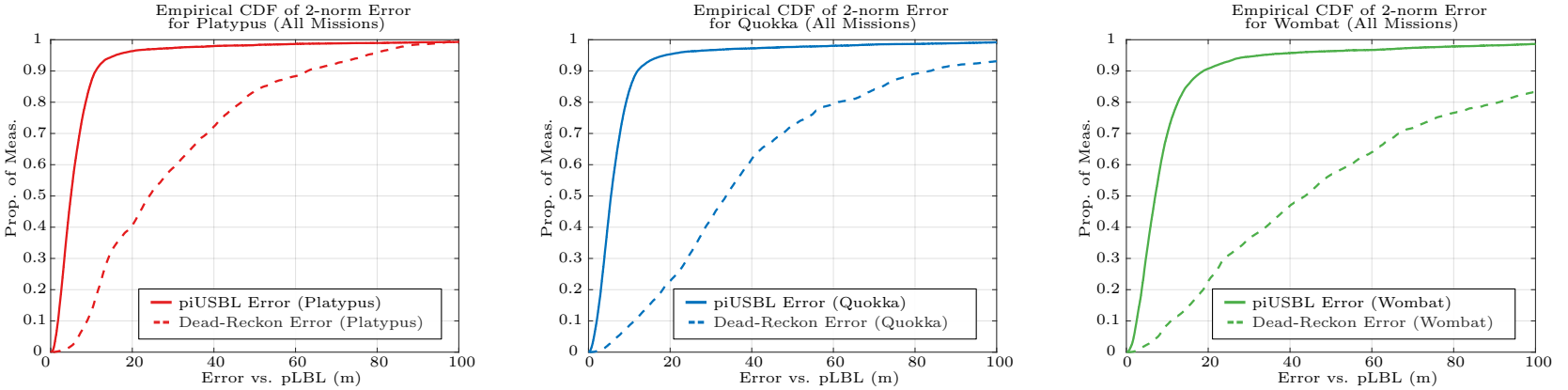

Figure 6.22: The piUSBL and dead-reckoning position error referenced against passive LBL plotted as empirical CDFs, for *Platypus*, *Quokka*, and *Wombat*, using data from all six multi-AUV deployments – solid lines are used for the CDFs of piUSBL error, and dashed lines are used for the CDFs of dead-reckoning error. *Left*: *Platypus*. *Middle*: *Quokka*. *Right*: *Wombat*.

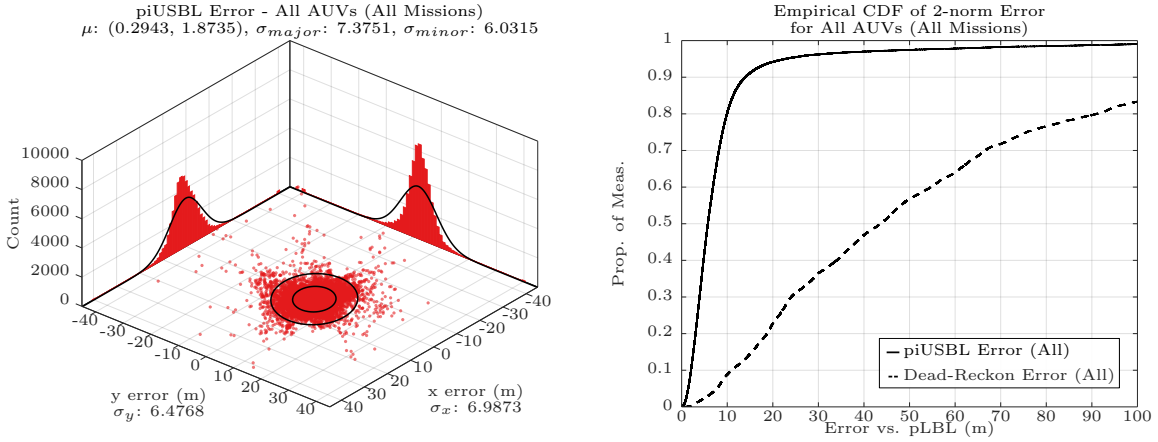

Figure 6.23: The piUSBL position error referenced against passive LBL, using data from the entire fleet from all six multi-AUV deployments – *Left*: piUSBL errors are plotted as dots in the x-y plane, with 1σ and 2σ covariance ellipses from multivariate Gaussian fitting; the marginal error distributions in x and y are plotted as histograms on the 'walls' of the plot, with Gaussian fits. *Right*: the empirical CDFs of piUSBL and dead-reckoning position error shown as solid and dashed lines respectively, using data from the entire fleet over all six deployments.

	piUSBL Position Error vs. Passive LBL (m)						
	(x, y) Error					**2-norm Error**	
	$\boldsymbol{\mu_{x,y}}$	$\boldsymbol{\sigma_{major}}$	$\boldsymbol{\sigma_{minor}}$	$\boldsymbol{\sigma_x}$	$\boldsymbol{\sigma_y}$	**68%-tile**	**95%-tile**
Platypus	(0.205, 1.551)	5.433	4.604	5.425	4.613	6.85	15.81
Quokka	(0.539, 2.754)	6.001	5.526	5.689	5.846	7.45	18.52
Wombat	(0.113, 1.223)	10.430	7.080	9.368	8.436	9.66	32.75
***All* AUVs**	(0.294, 1.874)	7.375	6.032	6.987	6.477	7.89	22.74

Table 6.5: Statistics of piUSBL navigation error referenced against passive LBL for the six multi-AUV missions operating under the relative navigation operating paradigm.

plot on the right of Fig. 6.23 shows the empirical CDF calculated using all this data, for both piUSBL and dead-reckoning position errors. These plots indicate that 68% of position errors from our piUSBL system operating under relative navigation falls below a 2-norm distance of 7.89 m across all vehicles. These statistics provide convincing evidence of the accuracy of piUSBL positioning when operating under the *relative navigation operating paradigm*. Recall that a number of factors contribute to these error statistics, not just the error inherent in piUSBL positioning – errors in the recorded GPS position of the beacon and motorboat, as well as errors in passive LBL due to acoustic effects also add to the total error. In fact, passive LBL can produce significant outliers, especially at the limit of its range, as is illustrated in the bottom-right plot of Fig. 6.16.

6.4 Proof-of-Concept Multi-AUV Applications

Now that we've demonstrated both the accuracy and utility of the *piUSBL relative navigation operating paradigm*, here we provide some simple example applications that demonstrate the advantage or uniqueness of multi-AUV deployments.

6.4.1 Environmental Monitoring

An especially useful application that requires the use of multiple AUVs is the ability to diversely and accurately sample oceanographic phenomena that vary quickly in both space and time – for example, biochemical or chemical plumes, one whose spatial pattern can vary rapidly over time due to the movement of ocean currents. By positioning vehicles over the spatial extent of the plume, recordings can be gathered that measure how the plume changes over time at each AUV position. Such measurements can be used to provide a more accurate and diverse dataset as an input to a model of the phenomena. Such models are usually described by a series of partial differential equations, so data that measures how parameters of the model vary in space and time could allow for a relaxation of model constraints, enabling us to improve how these plumes are modeled and provide us with a better understanding of their spatiotemporal evolution.

Unfortunately, our fleet of SandShark AUVs were not equipped with any sensors that would allow them to measure environmental phenomena, such as conductivity-temperature-depth sensors. However, the vehicles are each equipped with an altimeter, which allows them to measure their altitude above the riverbed – summing this altitude data with corresponding depth data from the AUV's pressure sensor allows us to obtain an estimate of the bathymetric depth at the position of the vehicle. Unfortunately, only one vehicle in our fleet, *Platypus*, has an operational altimeter, with the altimeters on *Quokka* and *Wombat* not functional.

Using the piUSBL position estimates of *Platypus* over the course of all six missions presented earlier in this chapter, we estimate the bathymetric depth of the riverbed at each position via:

$$z_{bathy} = z_{alt} + z_{depth} \tag{6.1}$$

These bathymetric depths are plotted at each position along the six mission trajectories in the left plot of Fig. 6.24 – the trajectories are overlaid on an existing bathymetric map of the Charles River. We then select a square slice of area, outlined by the black box in Fig. 6.24, and subdivide the area into a regular grid of 1×1 m cells. Using MATLAB's[2] `griddata` function, we then perform a cubic interpolation to fit the bathymetric depths along the vehicle trajectories onto this regular grid; this interpolated surface is then blurred with a

[2] https://www.mathworks.com/products/matlab.html/

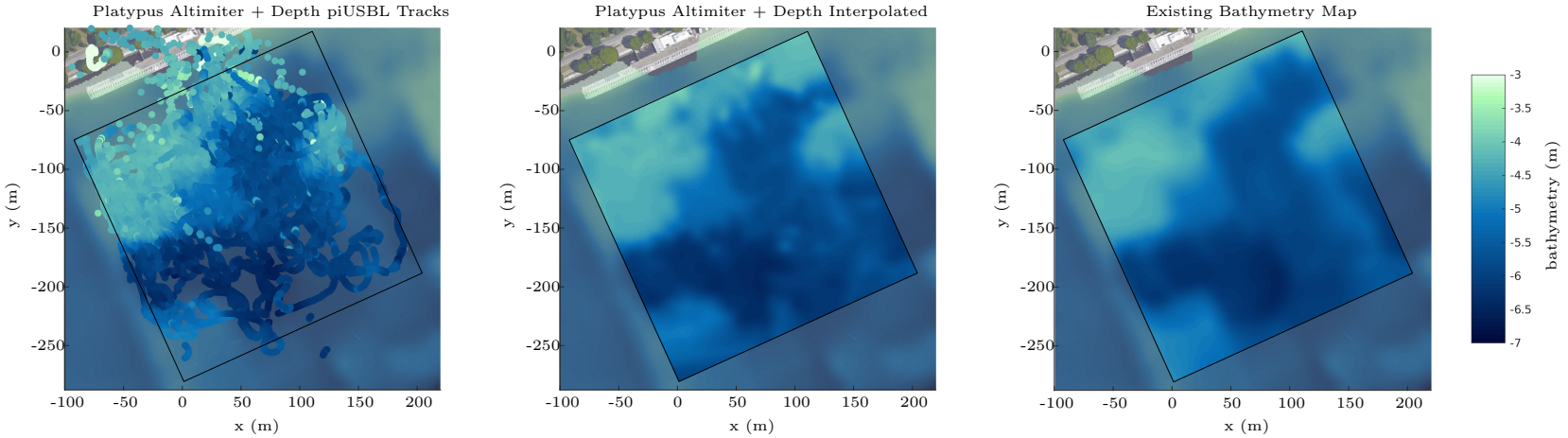

Figure 6.24: Depth-map of bathymetry generated using *Platypus* altimeter measurements – plots are overlaid on an existing bathymetric map of the Charles River. *Left*: altimeter plus depth data along *Platypus* AUV trajectories gathered over the six missions. *Middle*: interpolation of *Platypus* bathymetric depth measurements on a regular grid of 1×1 m cells and smoothed using a Guassian kernel. *Right*: existing MIT Sea Grant bathymetric map over the same grid.

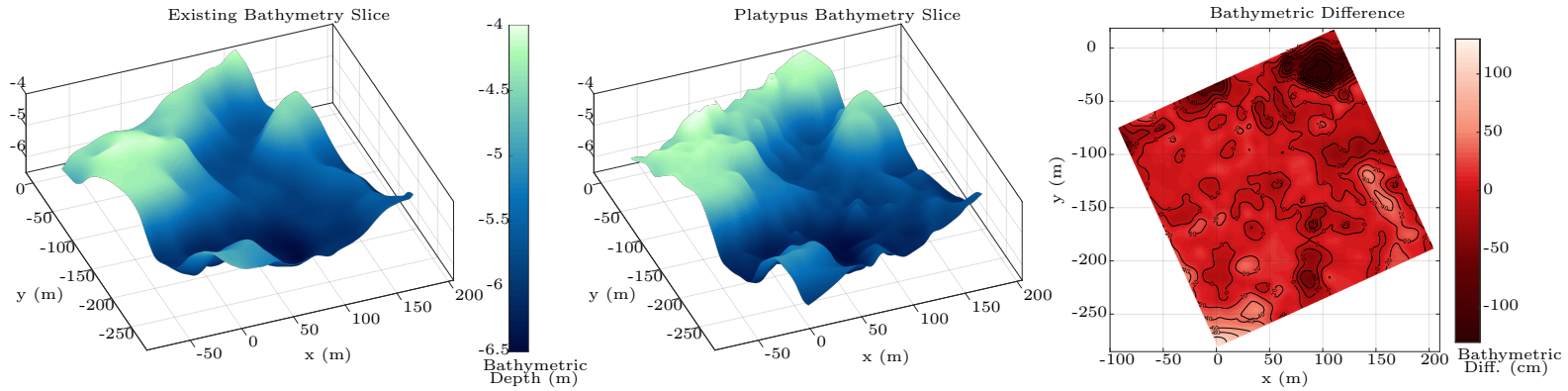

Figure 6.25: Bathymetric surface slice generated using *Platypus* altimeter measurements – *Left*: bathymetric slice from existing MIT Sea Grant bathymetric map. *Middle*: bathymetric slice using *Platypus* bathymetric depth measurements. *Right*: difference between the two slices in centimeters; 78.7% of the difference is within 20 cm, and 94.7% is within 20 cm.

Gaussian smoothing kernel with a 1σ value of 4 m using MATLAB's `imgaussfilt` function – this results in the bathymetric map generated by the interpolation and smoothing of *Platypus* measurements shown in the middle of Fig. 6.24. For comparison, the right plot of Fig. 6.24 shows the existing bathymetric map of the area, generated by the Charles River Alliance of Boaters and MIT Sea Grant[3] [191]. This data was collected on a motorboat using a Lowrance GPS unit for positioning, and a consumer fish-finder sonar; data was collected at along-track increments of 3 cm at a rate of 10 Hz and a boat speed of up to 2.57 $\mathrm{m\,s^{-1}}$; the boat was run in parallel tracks of 10–20 m spacing; the bathymetric map was generated from this fish-finder data using commercial software [191]. It is clearly apparent that the bathymetric map produced by *Platypus* agrees extremely well with the existing map, with both sharing many of the same features.

The existing bathymetric map and the bathymetric map produced by our AUV altimeter-plus-depth measurements are shown as surface slices in Fig. 6.25 – again, the close similarity between the two is apparent. The right plot in Fig. 6.25 shows the difference between the two surface slices in centimeters; we can see that the majority of our interpolated slice is within 20 cm of the existing bathymetry slice – in fact, 78.7% of the bathymetric slice produced by *Platypus* measurements has a difference of less than 20 cm, and 94.7% has a difference of less than 50 cm, with large differences occurring in the top-right and bottom-left corners of the slice, in which there were very few or no altimeter measurements. This close agreement between our bathymetric map and the existing bathymetric map is further evidence of how accurately our piUSBL system is able to navigate.

Although measuring bathymetry is far from an ideal application of multi-AUV deployments, we present it here as a surrogate of how multiple vehicles may be used to obtain environmental measurements. In this case we used the trajectories from one vehicle over many deployments to generate our map – but imagine instead that the altimeters on all three vehicles were functional; in that case, instead of performing multiple deployments over many hours, we would have been able to generate the same bathymetric map using a single deployment with multiple vehicles instead, thus saving a large amount of time. This highlights the advantage of multi-vehicle deployments not only to measure highly varying spatiotemporal phenomena, but also to be able to gather measurements more efficiently in terms of time and cost.

6.4.2 Fleet-Wide Coherent Source Localization

Another possible application that can only be achieved through the use of multiple AUVs, is deploying multiple vehicles for use as single elements in a 'virtual' acoustic array composed of all the AUVs in the fleet. Such a virtual array would possess a singular noteworthy advantage:

[3] https://seagrant.mit.edu/

its geometry could be dynamically modified on-the-fly in order to optimize for its ability to detect and localize non-cooperating acoustic sources in the ocean, be they man-made or natural. For example, a horizontal line of AUVs could be used to detect the bearing to an acoustic source, and once a certain level of confidence in the bearing measurement has been achieved, the vehicles could then alter their positions to form a vertical line in order to determine the elevation to the acoustic source. Additionally, the spacing between AUVs in this virtual line array can be dynamically set in order to tune for the best detection of an acoustic source operating within a certain frequency range.

To demonstrate this concept of fleet-wide coherent acoustic source localization, we use acoustic data measured by our fleet of three SandShark AUVs during the first mission on September 14, and perform time difference of arrival (TDOA) angle estimation to estimate the angle to the West passive LBL beacon secured to the dock. The acoustic data measured by the first element of the USBL array on each vehicle was first matched filtered against a template of the 250–1500 Hz, 20 ms LFM up-chirp broadcast by the passive LBL beacon; the differences in the maximums of these matched filter outputs were then used as the time difference between when the broadcast signal reached one AUV and when it reached the other AUV, $\Delta T_{i,j}$. We can then use TDOA angle estimation to estimate the bearing to the beacon [125]:

$$\cos(\phi) = \frac{c \cdot \Delta T_{i,j}}{||\boldsymbol{x}_{i,j}||} \tag{6.2}$$

$$\cos(\phi) = \frac{\boldsymbol{u} \cdot \boldsymbol{x}_{i,j}}{||\boldsymbol{x}_{i,j}||} \tag{6.3}$$

where ϕ is the bearing to the source, c is the speed of sound in water, $\boldsymbol{x}_{i,j}$ is the vector pointing from element i to element j, and $\boldsymbol{u}$ is the unit vector pointing in the direction of the beacon. Combining these equations leads to:

$$\boldsymbol{u} \cdot \boldsymbol{x}_{i,j} = c\Delta T_{i,j} \tag{6.4}$$

which can be rewritten as:

$$u(x_j - x_i) + v(y_j - y_i) + w(z_j - z_i) = c\Delta T_{i,j} \tag{6.5}$$

where the position of element i is (x_i, y_i, z_i). In our case we have a virtual array, and the element positions are the positions of the AUVs – we use the piUSBL position estimates as the positions of our virtual array 'elements', of which we have three, and formulate the system of equations:

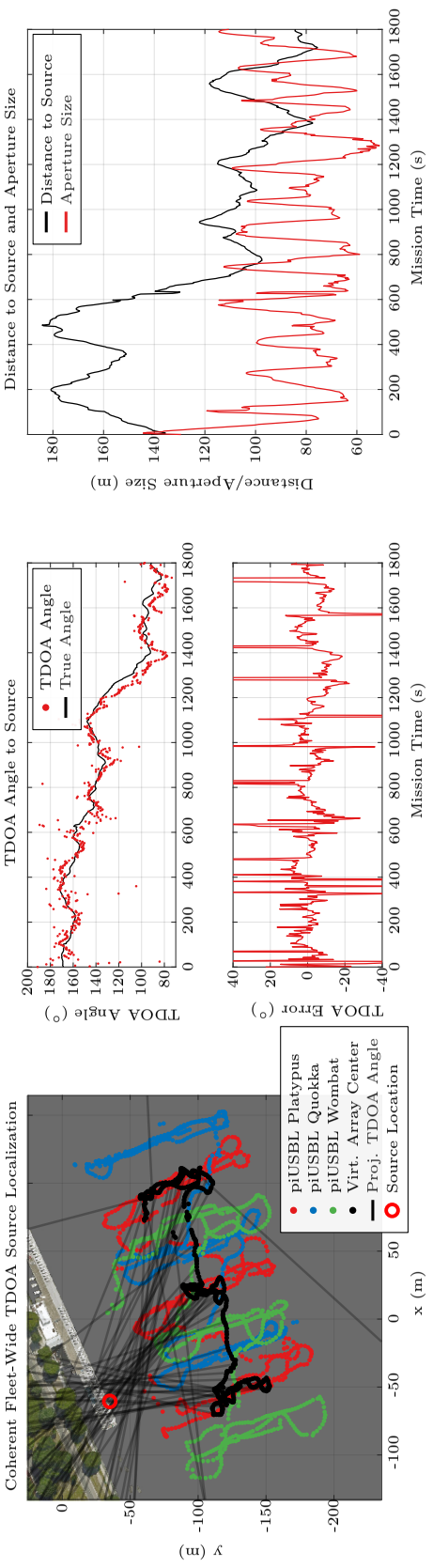

Figure 6.26: Acoustic source localization using the fleet of SandShark AUVs as a virtual acoustic array – *Left*: piUSBL positions of *Platypus*, *Quokka*, and *Wombat* in red, blue, and green respectively, are used as the positions of the elements in the virtual array; the virtual array center is shown in black, and the bearing estimates from time difference of arrival (TDOA) angle estimation are shown as projected semi-transparent black lines. *Middle*: the true bearing to the source calculated from the virtual array center is shown as the black line, with the TDOA bearing estimates shown with red dots; the lower plots shows the error between the TDOA estimate and the true bearing. *Right*: plots of the distance to the source and the array aperture show that they are of the same order of magnitude, indicating that the source is not in the far-field.

$$\begin{bmatrix} (x_w - x_q) & (y_w - y_q) & (z_w - z_q) \\ (x_w - x_p) & (y_w - y_p) & (z_w - z_p) \\ (x_q - x_p) & (y_q - y_p) & (z_q - z_p) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} c\Delta T_{q,w} \\ c\Delta T_{p,w} \\ c\Delta T_{p,q} \end{bmatrix} \tag{6.6}$$

where we have used the subscripts p, q, and w to indicate *Platypus*, *Quokka*, and *Wombat*. We solve this system of equations at every timestep of the first mission on September 14 using least-squares, calculating the TDOA between AUVs using matched filtering and using the position estimates from piUSBL; the values of u and v from these solutions are converted to circular coordinates to obtain an estimate of the bearing, ϕ. The output from this process is shown on the left in Fig. 6.26, where the bearing lines have been projected from the virtual array center at each timestep. This plot shows the positions of the virtual array 'elements' (the piUSBL positions of each vehicle) as colored dots, the center of this virtual array as black dots, the projected lines from the estimated TDOA angles, and the true position of the beacon as a red circle. It is clear from this plot that the estimated bearings more-or-less point in the direction of the beacon.

The middle top plot of Fig. 6.26 shows the true bearing to the beacon from the virtual array center in black, and the estimated bearing from TDOA as red dots; the lower plot shows the error between the true and estimated bearing to the beacon. These plots demonstrate fairly good agreement between the true and estimated bearings.

The main reason why the bearing estimation is not very accurate is shown in the right plot of Fig. 6.26 – this TDOA angle estimation formulation assumes that the acoustic beacon is in the far-field, where the distance to the source is much larger than the aperture (spacing between elements) of the virtual array. In this plot we can clearly see that the array aperture is on the same order of magnitude as the distance to the beacon, meaning that the source is in the near-field of the virtual array. It was shown in [125] that the angular error grows very rapidly as the source moves from the far-field to the near-field – this is the cause of the error in our bearing estimate from TDOA. Nevertheless, these results are a promising proof-of-concept toward using a fleet of AUVs as a virtual array for acoustic source localization; as the number of vehicles in the fleet grows, the ability for such a virtual array to accurately detect and track non-cooperating acoustic sources should improve significantly.

6.5 Summary

In this chapter we outlined the components that make up the *piUSBL relative navigation operating paradigm*, namely: (i) piUSBL positioning and closed-loop navigation; (ii) vehicle behaviors specifically designed to operate in a beacon-centric frame of reference with relative

navigation; (iii) operator fleet command via mode switching through the broadcasting of unique signals by the piUSBL beacon; and (iv) operator fleet control in the absolute frame of reference via the movement of the piUSBL beacon in order to 'lead' the vehicles in the fleet. This operational paradigm represents a powerful method of command-and-control for multi-AUV deployments, enabling a small number of operators to intuitively and easily plan and execute multi-AUV deployments. The power of this operating paradigm was demonstrated via a number of multi-AUV deployments using our fleet of three SandShark AUVs. We presented results from six of our 12 total deployments, demonstrating how the fleet was successfully able to perform closed-loop, relative navigation, showing how the operator was able to command the fleet to perform different behaviors on demand, and illustrating how the operator was able to control the absolute position of the group of vehicles as a whole through the movement of the piUSBL beacon.

We also provided statistics on the accuracy of piUSBL navigation operating under this paradigm, by comparing its position solution to that of a secondary passive LBL solution computed offline. These results demonstrated that piUSBL navigation is able to provide an accurate and bounded position solution for our fleet of low-cost, miniature SandShark AUVs, especially compared to naive dead-reckoning.

Finally, we also discussed some simple proof-of-concept applications for multi-AUV deployments. The cooperative use of multiple AUVs has the potential to revolutionize our understanding of complex oceanographic phenomena that vary rapidly in space and time; it also holds the possibility of opening up completely new applications, such as the use of multiple AUVs as elements in a 'virtual' acoustic array, whose geometry can be dynamically altered, in order to detect, localize and track non-cooperative underwater acoustic sources.

Deployments of multiple AUVs remain exceptionally rare – the results from the experiments detailed in this chapter represent some of the very few instances in which multi-AUV deployments have been practically demonstrated. The repeatability of these deployments, with six deployments performed over three days, illustrate just how robust our approach is for managing multiple AUVs, enabling these deployments to become routine. The *piUSBL relative navigation operating paradigm* holds great promise in becoming a standard method for multi-AUV deployments – however, there are still some issues to address that would allow the system to be improved; we describe these issues in the next and final chapter.

Chapter 7

A Vision of Ubiquity:
Closing Remarks and the Future

The body of work contained in this manuscript makes contributions to the field of underwater robotic navigation, with the aim of of democratizing autonomous underwater vehicle (AUV) technology and enabling multi-AUV deployments to become more commonplace. To achieve this aim, this work has developed an acoustic localization and navigation system termed Passive Inverted Ultra-Short Baseline (piUSBL) navigation, which addresses two major contributors to the inaccessibility of underwater vehicle technology: the high-cost and size of conventional AUVs due to the use of sophisticated navigational sensors including the Doppler velocity log (DVL) and high-grade attitude and heading reference system (AHRS), and the expense and unwieldiness associated with conventional AUV deployments that often require the need for expert personnel and support platforms. The piUSBL system has been demonstrated to provide accurate navigation, is scalable, low-cost, low-power, and easy to deploy, and combined with the novel operating paradigm of relative navigation, enables a single operator to intuitively command and control a fleet of AUVs. By equipping a new generation of inexpensive and miniature AUV with a similarly low-cost navigation suite, this piUSBL system has potentially provided the missing key that could allow multi-AUV ocean deployments to be democratized.

7.1 Contributions and Concluding Remarks

The piUSBL system is characterized by the use of a single, periodically broadcasting acoustic navigation beacon, and by a passive hydrophone array mounted on each underwater vehicle. Time synchronization of the beacon and the receivers enable each vehicle to determine relative range to the beacon, and acoustic processing of the signals captured by the array enable

each vehicle to determine the relative angle to the beacon. Bayesian filtering of these acoustic measurements and fusion with vehicle inertial measurements provides each vehicle with a temporally-consistent and accurate estimate of the relative position of the beacon, resulting in a navigation system that can localize an arbitrary number of AUVs in a manner that is convenient and inexpensive. This work first detailed a prototypical implementation of this system on a prototype low-cost and miniature SandShark AUV. Data collected by the prototypical piUSBL system successfully validated the feasibility of this approach, and was used to identify limitations to be addressed. Chapter 3 detailed the sequential Monte-Carlo beamformer (SMCB), the solution provided for the issue of the high computational cost associated with the two-stage processing pipeline of the conventional beamformer (CBF) followed by particle filtering. Close coupling of beamforming and particle filtering significantly increased computational efficiency, enabling the described closed-loop navigation experiments that were performed using the prototype SandShark AUV, as well as those undertaken with a conventional Bluefin-21 AUV. Chapter 4 then detailed a novel beamforming method developed in this work to address the issue of low-resolution angle measurements that were the result of using the CBF. element pair decomposition (EPD) beamforming is an elegant approach that drastically reduces computational and memory cost for angle estimation, by making use of the insight that the beamforming output at any angle can be estimated through the intersection of coning angles from pairs of elements in an array. This reduction in memory usage enabled the piUSBL system to increase the resolution of the acoustic angle measurement, significantly improving its precision. Chapter 5 detailed the implementation of the improved final piUSBL system on an autonomous surface vehicle (ASV), as well as on a fleet of three commercial SandShark AUVs. Data collected by the ASV piUSBL system to two independent navigation beacons was used to quantify the localization accuracy of the system. Finally, chapter 6 provided results from a set of multi-AUV experiments using the fleet of three commercial SandShark AUVs. These experiments made use of a novel operating paradigm called relative navigation, which use a set of custom vehicle behaviors designed for use in a beacon-centric reference frame. Relative navigation also provides a intuitive and user-friendly method of fleet command-and-control, by allowing the operator to switch the fleet between different modes of behavior by selecting various signal broadcast by the beacon, and by enabling the operator to control the fleet-wide movement of the AUVs by moving the beacon itself, causing all the vehicles to shift their positions with it.

7.2 Limitations of the System

The current form of the piUSBL system has a number of limitations. We detail these limitations here, and provide suggestions on how to address them in the future.

- **Increasing system range:** In its current implementation, the piUSBL system has a maximum theoretical range of around 1500 m, due to the 1 Hz repetition rate of the beacon firing and the receivers synchronously recording (since the acoustic wave travels at approximately 1500 $\mathrm{m\,s^{-1}}$). For applications that require larger, multi-km ranges, such as acoustic homing and docking, the range of the system must be improved by reducing the rate at which the beacon transmits and the arrays receive. However, this reduction necessarily decreases the navigational performance of the system, since the piUSBL receivers collect fewer measurements over a given period of time. In addition, because the receivers must record acoustic data over a larger time window, processing of the data to determine range and angle to the beacon must compensate for platform movement in the interim between the start of the recording and the incorporation of the measurements into the filter. This range/performance trade-off must be carefully considered depending on the use-case of the system.

  Increasing the range of the system would likely be a fairly straightforward process – since the acoustic beacon and each of the piUSBL receivers in the fleet share a synchronized, time-aligned clock (typically Coordinated Universal Time from GPS), the beacon and receivers can be configured to simply transmit and record at the start of every n-th second, beginning at some pre-determined time. This approach maintains synchronization between beacon firing and receiver recording, and increases the travel-time of the acoustic wave to any arbitrary length. As mentioned, doing so increases range without bound, at the cost of a likely performance reduction in terms of positional accuracy.

- **Complex acoustic environments:** The current iteration of the system operates under the assumption that the acoustic signal travels along the shortest, direct-path between the beacon and the receiver. This assumption generally holds at the ranges demonstrated in this work, since the density of the water in a localized area tends to be consistent – as such, the acoustic 'ray' tends to linear, and does not deviate along a refractive path caused by changes in sound speed.

  In environments that have complex sound-speed fields, or which have bathymetry or objects that can result in complex acoustic effects such as multipath and reflections, the path along which the strongest acoustic response travels may differ greatly from the direct-path. This may also occur at longer ranges where changes in seawater density can have a significant effect on the travel path of the acoustic energy. In cases like these, the piUSBL system presented in this work would suffer from limitations in accuracy due to the breakdown of this direct-path assumption.

  One possible approach that could be used to accurately account for how the acoustic energy travels through these complex underwater environments is to integrate a computational acoustic model of the environment within the piUSBL navigation system.

Different computational models using different underlying approximations can be used to solve the wave equation in order to understand how acoustic energy propagates within the environment, and what the distribution and composition of the sound pressure field is in the environment [192]. These models include BELLHOP [193], which uses a ray-tracing approximation, KRAKEN [194], which is a normal-modes-based model, and OASES [195], which uses a wave number integration approach. By simulating how the sound from the beacon propagates within the environment that the fleet of vehicles is operating in (taking into account how the acoustic energy 'bends' along fluid boundaries of different densities, how the sound is reflected off and attenuated by objects in the environment, the water surface and the sea-bottom, and how the multiple paths of the sound interfere in constructive and destructive 'zones'), the piUSBL receiver would be able to compensate for these environmental effects so as to improve its accuracy in determining range and angle to the beacon. High-fidelity simulation of the environment can be performed offline if the computational acoustic model is demanding, with the resulting 'map' provided to the vehicle before deployment; if the model is less computationally demanding, then in-situ modeling of the acoustic environment is also a possibility, which could also be done while integrating sensor measurements collected by the vehicle (e.g. salinity, temperature, pressure, conductivity) on-the-fly. Integrating acoustic models on-board an AUV was successfully demonstrated by Schneider [196] for improving acoustic communication in an anisotropic shallow water environment and in the deep sea. Compensating for how sound propagates underwater via acoustic models is especially important in environments such as the Arctic, where interaction of the sound with ice floes and fields is complicated, and where the sound-speed profile can be highly nonlinear, as demonstrated by experimental results from Schmidt [197] – complex acoustic propagation due to a local maxima in the sound-speed profile was shown to have a drastic effect on the performance of acoustic navigation and communications.

- **Physical environmental conditions:** Physical environmental conditions such as ocean currents, bathymetry, internal waves, eddies and fronts, all have a direct, physical effect on the operation of AUVs, as well as having an effect on acoustic propagation. The current piUSBL system is limited by not considering the physical environment and its effects on both vehicle dynamics and acoustic navigation.

  As mentioned in the previous point, computational models can be used to provide the piUSBL system with a better understanding of the operating environment, both in terms of acoustics as well as the ocean's physical parameters. A coupled physical-acoustic model such as that demonstrated by Lermusiaux [198] can be used to improve the accuracy of acoustic modeling through the integration of physical ocean properties, thereby possibly improving the navigational performance of the piUSBL system. Physi-

cal properties such as ocean currents may have a significant impact on the ability of the AUV fleet to carry out their mission; another possibility of performing ocean modeling (either pre-deployment or in-situ) is that currents and tidal effects can be harnessed by the vehicles in order to perform their mission more efficiently – this concept was illustrated by Wang [199] in an exercise that used AUVs and ASVs for adaptive sampling in order to improve ocean field estimates for acoustic predictions.

- **Acoustic multipath and ambiguous measurements:** The current piUSBL system is somewhat limited in its ability to discern between ambiguous measurements caused by acoustic effects such as multipath spreading; if a secondary mode in the acoustic measurements persists for a significant period of time, then the piUSBL particle filter can incorrectly 'lock-onto' this mode, resulting in incorrect position estimates.

  The piUSBL system can be improved to disambiguate the correct mode in multimodal acoustic measurements through two approaches: the first is to model the acoustic environment in order to detect areas in time or space in which ambiguous measurements occur due to multipath, and to incorporate this model within the filter in order to allow for the system to compensate; the second is to use a *smoothing*, rather than filtering approach, in which the smoother can account for multimodal measurements – by integrating multimodal measurements over the entire vehicle trajectory, short periods in which secondary modes occur can be effectively detected and ignored, since they will not be consistent with the remaining measurements collected over the entire mission. Fourie [200] developed a multimodal smoothing framework known as multimodal incremental smoothing and mapping, which can be used to perform smoothing using non-parametric measurements and models.

- **Signal-to-Noise:** In its current form, the piUSBL system relies on an acoustic beacon that transmits a large amount of energy into the water – the signal-to-noise ratio (SNR) of the system is typically 10–25 dB. Because the system must detect the transmitted signal in order to determine range and angle, navigational accuracy would likely reduce significantly with lower SNR.

  This limitation can be addressed in a number of ways. One possibility is to use a lock-in amplifier to modulate the signal within a narrow bandwidth, and filter out any broadband noise. Another approach is to design an acoustic signal that can maintain a high SNR while reducing the amount of acoustic energy transmitted. This can be done, for example, by using a specific coding scheme, or by transmitting a periodic signal faster than 1 Hz and averaging the measurements over time – this would cause the noise to reduce as the square root of the averaged samples. To get around the range limitation caused by transmitting at a high rate, one possibility is to continuously transmit a LFM

up-chirp during odd seconds, and to continuously transmit a LFM down-chirp during even seconds – this would allow the system to determine range by detecting the onset of the signal change, while improving SNR by averaging the LFM signals transmitted during the full second. One final possibility is to measure and model the environmental noise; if the characteristics of the noise are known and understood to be very different to the signal characteristics, it would be possible to simply filter out the noise and increase the SNR.

One final note to make is that the piUSBL system in its current state is not stealthy, with the beacon being easily detectable – it is completely unsuitable for military stealth applications, since an adversary can easily detect its use. Designing a signal that mimics environmental noise or natural ocean sounds could be a possible method of overcoming this limitation by acoustically camouflaging the system against background noise.

7.3 Improving the System

There are a number of improvements to the piUSBL system that are the subject of ongoing and future work.

- **Array geometry, calibration and local acoustic effects:** The local acoustic effects experienced by the piUSBL receiver is a significant source of localization error, as a consequence of biasing the acoustic angle measurement away from the true direction. Further work in mitigating these effects through the design of improved array geometries, calibration procedures, or via physical baffles for acoustic absorption should be investigated in order to significantly improve the accuracy of the system.

- **Operational considerations:** Relative navigation is a user-friendly approach for fleet-wide command and control, enabling an operator to command all vehicles to change their behavior, control their movement, and to request all vehicles to return to a single location. However, improved methods and the development of a holistic and systematic approach for deploying, retrieving, and downloading and managing vehicle data would significantly improve the operator experience.

- **Fleet tracking:** The current piUSBL system enables multiple vehicles to navigate, but the operator has no insight into the state of the vehicles. AUVs are trusted to operate correctly within their predefined behavior parameters. Methods for tracking the positions of all vehicles and to enable the operator to understand the state of the fleet are a necessary improvement to making the system truly useful. One possibility is to have the beacon transmit a tracking signal concurrently with the navigation signal, and

use acoustic reflections off the vehicles to track their position using an array collocated with the beacon.

- **Synchronization without a CSAC:** The chip-scale atomic clock (CSAC) comprises the majority of the cost of the piUSBL system. However, no other hardware alternative exists that can enable synchronization between the beacon and the receiver to the degree necessary for accurate ranging. Possible algorithmic approaches to estimate time drift using a lower cost receiver clock would be a useful avenue of investigation for reducing the total cost of the piUSBL system – for example, by transmitting two chirps in quick succession, and assuming that the range is constant during this period, it may be possible to track the clock drift sufficiently.

- **Trajectory synchronization:** Currently, AUVs in the fleet use behaviors that command the vehicles to follow a desired spatial trajectory without placing any temporal constraints on the vehicles. It would be very useful to extend these behaviors to include time-parameterization, in order to synchronize the movement and positions of multiple vehicles. By doing so, the fleet could be used to perform synchronized sampling.

- **Improved Bayesian filtering:** The particle filtering approach used in this work has been demonstrated to be effective, but alternative Bayesian filtering methods may prove to have greater accuracy. Approaches that include a mixture of distributions to represent the state are of particular interest. One possible approach is to separate the tracking problem into two filters: one to estimate the angle to the beacon using a mixture of circular distributions, and a second to estimate the range to the beacon using a Gaussian mixture and a bank of parallel extended Kalman filters (EKFs). For example, the work of Markovic [201] provides an approach to filtering directional data, such as the measurements output by beamforming, using a mixture of von-Mises distributions. Alternatively, approaches using EKFs on Lie Groups such as those suggested in [202] and [203], or an approach like [204] in which the complications of directional statistics in spherical coordinates are directly taken into account, may provide methods by which to explicitly construct a *hybrid* state space that combines non-Euclidean (angle) and Euclidean (range) variables within a single filter.

- **Post-processing of fleet trajectories:** The work detailed in this thesis focused on providing an online, closed-loop navigation solution for low-cost AUVs. As such, the on-board trajectories estimated by these vehicles tend to have discontinuities, due to the nature of the Bayesian filtering methods used. However, once the deployment has concluded and vehicle data has been retrieved, there is the possibility of *smoothing* vehicle trajectories offline so that the end-product of collected samples are more accurately georeferenced in a global coordinate system. Optimization of the positions of all vehi-

cles, as well as that of the navigation beacon, can be performed *all together*, in a batch processing approach, for example, using nonlinear least-squares techniques. Another possibility is to incorporate the work of Fourie [205], which provides a method of representing multi-modal measurement distributions in a factor graph framework; using this approach, we can directly represent acoustic angle and range measurements collected by the piUSBL system in order to optimize all vehicle trajectories in a principled manner. Real-time, on-board multi-modal SLAM using Fourie's framework may also be possible using a fixed-lag structure to reduce its computational burden.

- **Further investigation of element pair decomposition (EPD) beamforming:** EPD beamforming has provided a novel approach for generating the angle measurement distribution with a low computational and memory cost. Further work should be performed to investigate its properties, in particular its sensitivity to parameters such as array geometry, anytime stopping, and number of coning angles. This beamforming approach also provides intuitive insights into array design in general; for example, the improved performance of random and spiral arrays can be understood in terms of the variation in aperture sizes of different array element pairs. A further appealing possibility for future work is to use this decomposition of the array into element pairs directly as a method for designing arrays: an optimization method by which pairs are sequentially added to an array in order to achieve a desired beampattern may be conceivable with this approach.

- **Theoretical guarantees and accuracy bounds:** This body of work has primarily provided experimental results and statistical evidence of the accuracy of piUSBL navigation. Theoretical guarantees and bounds on accuracy have been investigated previously by Jakuba [95], in which he numerically generated the error surfaces in position of a OWTT inverted USBL system under some accuracy assumptions on range and angle measurements. This line of work can be extended by placing additional assumptions on the array geometry, which would provide a numerical understanding of the accuracy of beamforming using a given array (via measurement of its theoretical half-power beamwidth), as well as via the selection of a specific beacon-transmitted signal, which would provide a measurement of the achievable range resolution. Further work in assessing the theoretical performance of our system using an 8 cm pyramidal array and a 9–11 kHz LFM chirp can be performed by generating the positional error surfaces numerically using these design choices, and comparing these results to the experimental results provided in this manuscript. The error bounds of piUSBL in more complex underwater environments and the deep ocean can also be determined through the modeling of acoustic propagation in these environments, and investigating the impact of variations in the acoustic path on the assumption of range from the direct line-of-sight path

between the beacon and the receiver. This modeling could also be used to investigate the impact of SNR on the ability of the piUSBL system to accurately detect range.

- **Improved dead-reckoning via hydrodynamic modeling:** The current piUSBL system makes use of a very simple constant velocity model to perform the filter prediction step using vehicle attitude and speed from propeller RPM. Randeni [180] recently developed a hydrodynamic model of the SandShark vehicle using LBL position data of the AUV as well as IMU data, which was demonstrated to significantly outperform the simple constant velocity model in dead-reckoning. Integrating this improved hydrodynamic model within the prediction step of the filter is the subject of ongoing work, and could potentially improve the navigational ability of the SandShark fleet.

- **Expanding the fleet:** In this thesis we demonstrated multi-AUV deployments with a limited fleet of three SandShark AUVs. Expanding this fleet to include tens, or multiple tens of vehicles should be a priority for future work. How well the piUSBL navigation approach and command-and-control paradigm handles this expansion would be of particular interest, and would provide additional insight into the drawbacks and advantages of the relative navigation operating paradigm.

7.4 Using the System

The piUSBL system and command-and-control operating paradigm have opened up the possibility of novel uses for multi-AUV deployments.

- **Integrating sensors:** Ongoing work with the fleet of SandShark AUVs has been to integrate conductivity-temperature-depth (CTD) sensors in order to be able to demonstrate multi-AUV sampling of highly spatiotemporally varying temperature gradients. Further work should investigate other possible sensors to integrate with the fleet, especially sensors that can sample from ocean phenomena that can be better measured using multiple vehicles.

- **Beacon-centric behavior development:** This work has developed a small set of beacon-centric behaviors to demonstrate the relative navigation operating paradigm. The development of additional behaviors that are tailored for this paradigm should be the focus of future work, as well as developing a full planning and vehicle management system that is better suited to the relative navigation paradigm, since all current systems assume planning and mission operations in a global coordinate system.

- **Acoustic homing and docking:** By providing an estimate of the relative beacon position, the piUSBL system is ideal for providing an AUV or ROV with a means for

autonomous homing and docking underwater. Such a capability could enable *resident AUV* operations, whereby an underwater vehicle can dock autonomously for the purposes of recharging, uploading data, and downloading updated operator commands. The use of the system for this purpose is an interesting avenue for further research.

- **Multi-AUV sampling:** Multi-AUV deployments enabled by piUSBL navigation have the potential to revolutionize the sampling of complex ocean phenomena. Synoptic measurement of ocean features such as chemical plumes, temperature fronts, and internal waves are now a possibility, thus breaking the spatiotemporal aliasing associated with the measurement of such phenomena using single vehicles. Experiments demonstrating this ability should be of high priority.

- **Multi-AUV coherent acoustic processing:** This work demonstrated a proof-of-concept application of localization an acoustic source using the fleet of AUVs as a virtual acoustic array. Further work in performing formation control with the fleet should be undertaken, in order to provide additional demonstrations of the use of multiple vehicles as a volumetric array for the application of acoustic and seismic sensing.

- **Complex and real ocean environments:** The experimental results provided in this work were limited to operational environments that were fairly benign in terms of acoustics. Besides the large acoustic reflector present in the form of the stone river wall running the length of its North side, the water in the Charles River is shallow and very uniform in density, meaning that the direct-path assumption for range used in these experiments was more-or-less valid, especially at the sub 100 m ranges that occurred between the beacon and the vehicles. The results using the Bluefin-21 AUV in Broad Sound provided in section 3.7 represent an experiment in a slightly more complex ocean environment and at ranges up to 600 m, but one in which the direct-path assumption also held fairly well. Using the piUSBL system in more complex, real ocean environments and at longer ranges may prove challenging as acoustic energy travels along more complex paths due to density changes in the water column, meaning that the direct-path assumption no longer holds. This non-direct acoustic propagation may lead to biases in the estimates of range and angle to the beacon, increasing the positional error of the piUSBL system and degrading the navigational ability of each vehicle. Understanding how the positional error bounds grow in such complex environments is necessary in order to prevent these vehicles and their operators from becoming overconfident in their position estimates. However, by incorporating simple ocean and acoustic models within the piUSBL system on-board the AUV, and by integrating real-time conductivity-temperature-depth (CTD) measurements within these models, it may be possible to operate the system while accurately taking into account how the acoustic wave transmitted

by the beacon propagates within the environment. Alternatively, the piUSBL system can be operated under the direct-path assumption, with the understanding that the vehicle positions as estimated by the system are inaccurate – post-processing of AUV trajectories taking into account the acoustic environment could then be used to correct their positions post-mission. Doing so would mean that the vehicles may not accurately follow the desired trajectories during the mission (an offset would likely occur due to an inaccurate estimate of range), but data collected during the mission would be accurately geo-referenced in post-processing. Depending on the desires of the operator and the nature of the mission, this may be an acceptable trade-off – for example, missions that perform adaptive sampling do not require a highly accurate navigation estimate in-situ, but do require the sampled data to be accurately localized in post-processing.

7.5 A Vision of the Future: Persistent Networks of Underwater Robots

The use of multiple autonomous underwater vehicles (AUVs) to sample and measure the dynamics of complex ocean processes has been a long-held dream of oceanographic researchers, beginning with the early descriptions of networks of these vehicles by Curtin and Bellingham [37]. Their description of the Autonomous Oceanographic Sensing Network (AOSN) put forth a vision of multiple AUVs being used to sample frontal dynamics of the ocean, alongside other distributed acoustic and point sensors. Traversing the underwater newtork, the vehicles record temperature, salinity, velocity and other data, relaying their observations to network nodes in real-time by autonomously docking at the node. Connection of the underwater network to a centralized shore station would enable near real-time integration of data from the entire network of point sensors and vehicles, and allow shore operators to guide vehicle sampling, dispatching vehicles to locations of the front to obtain detailed cross-front and along-front measurements. Such a network of vehicles, point sensors, and nodes would allow oceanographers to understand the nature and extent of such highly varying ocean phenomena, from the comfort of a centralized, shore-based command center.

There has been a recent revival of interest in such *resident AUV* projects, with one of the first industry-academia resident AUV workshops being held at the University of Washington in 2018. The aim of this workshop was to gather input on the technical challenges and potential benefits of deploying a resident AUV as part of the Ocean Observatories Initiative (OOI) Cabled Array at Axial Seamount off the Washington-Oregon coast. Manalang and Delaney have detailed their vision for how the deployment of such a resident AUV can provide an unprecedented observation capability of eruption events along the Juan de Fuca Ridge [206].

A network of acoustic transponders mounted on the seafloor are envisioned to provide navigational aid and communications to the AUV, as well as be used as a means of detecting seafloor deformation, and to act as a communications backbone for additional low bandwidth sensors and instrumentation. A resident AUV would autonomously navigate within this network of acoustic transponders, operating over timescales of months or even years by autonomously docking to recharge, transmit data, and receive operator commands. This AUV would be used to gather high-resolution bathymetry, sonar and visual imagery, as well as water chemistry and microbial samples. During eruption events, the AUV would autonomously track and map the resulting volcanic plume, providing scientists with an unprecedented insight into the nature of such events, and their impact on the surrounding ocean ecosystem.

The complete and self-contained acoustic navigation system detailed in this body of work, and the associated contributions in low-cost underwater navigation, have the potential to further these visions of the future of underwater robotics. The successful demonstration of Passive Inverted Ultra-Short Baseline (piUSBL) navigation on a fleet of three low-cost and miniature commercial SandShark autonomous underwater vehicles (AUVs) provide validation that this approach can be used to support the vision of sampling and measuring complex oceanographic phenomena using multiple AUVs; And by providing a navigation solution relative to an acoustic beacon, the system can be used to implement the vision of resident AUVs via autonomous homing and docking. By equipping this new generation of inexpensive and small underwater robot with the ability to accurately navigate, this technology and its associated contributions can provide a path towards lowering the cost of entry into underwater robotics research and to furthering the use of these vehicles for ocean science. These contributions in low-cost underwater navigation have the potential to enable the more ubiquitous and coordinated use of robots to explore and understand the underwater domain.

Appendix A

Derivation of the Matched Filter

Suppose that the broadcast signal $s(t)$ (the template in Fig. 2.7) is present in the measurement $x(t)$ (the measured signal in Fig. 2.7), so that:

$$\boldsymbol{x(t)} = s(t) + \boldsymbol{n(t)} \tag{A.1}$$

where $\boldsymbol{n(t)}$ is noise, which is a stationary random process. Consider a LTI filter with impulse response $h(t)$ that takes as an input the measured signal to produce an output $\boldsymbol{y(t)}$, composed of a filtered signal term $y_s(t)$ and a filtered noise term $\boldsymbol{y_n(t)}$:

$$\boldsymbol{y(t)} = \boldsymbol{x(t)} * h(t) = \int_{-\infty}^{\infty} \boldsymbol{x(\tau)} h(t-\tau) d\tau \tag{A.2}$$

$$= (s(t) + \boldsymbol{n(t)}) * h(t) := y_s(t) + \boldsymbol{y_n(t)} \tag{A.3}$$

WHERE

$$y_s(t) = s(t) * h(t) = \int_{-\infty}^{\infty} s(\tau) h(t-\tau) d\tau \tag{A.4}$$

$$\boldsymbol{y_n(t)} = \boldsymbol{n(t)} * h(t) = \int_{-\infty}^{\infty} \boldsymbol{n(\tau)} h(t-\tau) d\tau \tag{A.5}$$

The goal is then to find a filter that will maximize the output signal-to-noise ratio (SNR) at some chosen time t_0. Intuitively, this can be understood as trying to detect the presence of a known signal $s(t)$ in some measurement $\boldsymbol{x(t)}$ that includes random process noise $\boldsymbol{n(t)}$ at time t_0, and doing so by using an LTI filter whose output will be much greater if the signal $s(t)$ is present than when it is absent; to do this, the filter should make the instantaneous power in $y_s(t)$ as large as possible compared to the average power in $\boldsymbol{n(t)}$ at time t_0, which is equivalent to maximizing the standard definition of the SNR. Assuming that the signal $s(t)$ is of finite duration, we can write it as the Fourier transform:

$$s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)e^{j\omega t}d\omega \tag{A.6}$$

WHERE

$$S(\omega) = \int_{-\infty}^{\infty} s(t)e^{-j\omega t}dt \tag{A.7}$$

$$\omega := 2\pi f \tag{A.8}$$

Recall that convolution in the time domain equals multiplication in the frequency domain:

$$y_s(t) = s(t) * h(t) \tag{A.9}$$

$$\Updownarrow$$

$$Y_s(\omega) = S(\omega)H(\omega) \tag{A.10}$$

Thus, we can rewrite $y_s(t)$ from Eq. A.4 using the inverse Fourier transform as:

$$y_s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)S(\omega)e^{j\omega t}d\omega \tag{A.11}$$

where $H(\omega)$ is the transfer function of the LTI filter whose impulse response is $h(t)$. Unfortunately, since the noise $\boldsymbol{n(t)}$ is a random process, it is generally difficult to obtain a similar complete specification for the output $\boldsymbol{y_n(t)}$ from Eq. A.5 using the Fourier transform as in Eq. A.11; however, from the Wiener-Khinchin theorem, we can write the average power of this noise process as:

$$\mathbb{E}\left|\boldsymbol{n(t)}\right|^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_n(\omega)d\omega \tag{A.12}$$

where $S_n(\omega)$ is the power spectral density of the noise at the input of the filter. At the output of the LTI filter, the noise has intensity:

$$\mathbb{E}\left|\boldsymbol{y_n(t)}\right|^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} \left|H(\omega)\right|^2 S_n(\omega)d\omega \tag{A.13}$$

The SNR at time t_0 can now be written in terms of Eqs. A.11 and A.13, as the ratio of the output power of the filtered signal to the output power of the noise:

$$\left(\frac{S}{N}\right)_{t_0} := \frac{|y_s(t_0)|^2}{\mathbb{E}|\boldsymbol{y_n(t)}|^2} \tag{A.14}$$

$$:= \frac{1}{2\pi}\frac{\left|\int_{-\infty}^{\infty} H(\omega)S(\omega)e^{j\omega t_0}d\omega\right|^2}{\int_{-\infty}^{\infty}|H(\omega)|^2 S_n(\omega)d\omega} \tag{A.15}$$

Now we wish to maximize this SNR in order to determine whether or not the signal $s(t)$ is present in the measurement $\boldsymbol{x(t)}$; in other words, we want the absolute value of $y_s(t_0)$ to exceed the quantity $\mathbb{E}|\boldsymbol{y_n(t)}|^2$ by as much as possible in order to improve our ability to detect the signal in our measurement. Consider the following form of the *Cauchy-Schwarz inequality*:

$$\left|\int_{-\infty}^{\infty} A(\omega)B(\omega)d\omega\right|^2 \leq \int_{-\infty}^{\infty}|A(\omega)|^2 d\omega \int_{-\infty}^{\infty}|B(\omega)|^2 d\omega \tag{A.16}$$

We can apply this inequality on the numerator of Eq. A.15 to find the transfer function $H(\omega)$ of the filter we are looking for:

$$\left|\int_{-\infty}^{\infty} H(\omega)S(\omega)e^{j\omega t_0}d\omega\right|^2 \leq \int_{-\infty}^{\infty}|H(\omega)|^2 S_n(\omega)d\omega \int_{-\infty}^{\infty}\frac{|S(\omega)|^2}{S_n(\omega)}d\omega \tag{A.17}$$

WHERE

$$A(\omega) = H(\omega)\sqrt{S_n(\omega)}e^{j\omega t_0}, \qquad B(\omega) = \frac{S(\omega)}{\sqrt{S_n(\omega)}} \tag{A.18}$$

The inequality shows that the SNR of Eq. A.15 can be simplified for the following upper bound:

$$\left(\frac{S}{N}\right)_{t_0} \leq \frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{|S(\omega)|^2}{S_n(\omega)}d\omega \tag{A.19}$$

Now, this upper bound can be *exactly* met if we take:

$$H(\omega) = ce^{-j\omega t_0}\frac{S^*(\omega)}{S_n(\omega)} \tag{A.20}$$

where c is an arbitrary constant, $e^{-j\omega t_0}$ is a simple time delay to the detection time t_0, and $S^*(\omega)$ is the complex conjugate of the Fourier transformed signal $S(\omega)$ that we are trying to detect. The filter defined by Eq. A.20 is thus the best LTI filter for detecting a known signal

in the presence of noise. If the noise $\boldsymbol{n(t)}$ is a Gaussian random process then this filter is an absolute optimum, producing an SNR of:

$$\left(\frac{S}{N}\right)_{t_0} = \frac{2}{N_0}\frac{1}{2\pi}\int_{-\infty}^{\infty} |S(\omega)|^2 \, d\omega \tag{A.21}$$

since the power spectral density of a Gaussian random noise process is $\frac{N_0}{2}$. The filter of Eq. A.20 has a very simple and intuitive meaning: since the transfer function of the filter is proportional to the amplitude spectrum of the signal we are trying to detect, this means that the filter attempts to only 'pass through' the frequencies which are present in the signal [171].

This leads us finally to the *matched filter*. For simplicity, let us consider white noise such that Eq. A.20 simplifies to:

$$H(\omega) = e^{-j\omega t_0} S^*(\omega) \tag{A.22}$$

Since the signal $s(t)$ is real, $S^*(\omega) = S(-\omega)$, and we can write the signal output after filtering (Eq. A.11) as:

$$y_s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-j\omega t_0} S(-\omega) S(\omega) e^{j\omega t} d\omega \tag{A.23}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} S(-\omega) S(\omega) e^{j\omega(t-t_0)} d\omega \tag{A.24}$$

using the Fourier transform of Eq. A.7, we can manipulate the equation as follows:

$$y_s(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(-\omega) e^{j\omega(t-t_0)} \left[\int_{-\infty}^{\infty} s(t') e^{-j\omega t'} dt'\right] d\omega \tag{A.25}$$

$$= \int_{-\infty}^{\infty} S(-\omega) e^{j\omega(t-t_0)} \left[\frac{1}{2\pi}\int_{-\infty}^{\infty} s(t') e^{-j\omega t'} dt'\right] d\omega \tag{A.26}$$

swapping the order of integration:

$$y_s(t) = \int_{-\infty}^{\infty} s(t') \left[\frac{1}{2\pi}\int_{-\infty}^{\infty} S(-\omega) e^{-j\omega t'} e^{j\omega(t-t_0)} d\omega\right] dt' \tag{A.27}$$

$$= \int_{-\infty}^{\infty} s(t') \left[\frac{1}{2\pi}\int_{-\infty}^{\infty} S(-\omega) e^{-j\omega(t'-t+t_0)} d\omega\right] dt' \tag{A.28}$$

and using the inverse Fourier transform of Eq. A.6 and the time-reversal property:

$$y_s(t) = \int_{-\infty}^{\infty} s(t')s(t' - t + t_0)dt' \tag{A.29}$$

Finally, we can use a change of variables to get:

$$y_s(\tau) = \int_{-\infty}^{\infty} s(t')s(t' - \tau)dt' \tag{A.30}$$

$$\text{WHERE} \quad \tau = t - t_0 \tag{A.31}$$

Recall that for a real, continuous-time signal, its autocorrelation is given by:

$$R_{ss}(\tau) = \int_{-\infty}^{\infty} s(t)s(t - \tau)dt \tag{A.32}$$

Comparing Eqs. A.32 and A.30, it is readily apparent that:

$$y_s(t) = R_{ss}(t - t_0) \tag{A.33}$$

Interestingly, the matched filter simply acts as a correlator whose output at time t_0 is $R_{ss}(t - t_0) = R_{ss}(0) = \int_{-\infty}^{\infty} s^2(t)dt = E$ (i.e. the autocorrelation function delayed to have a maximum peak at time t_0). In the presence of noise $\boldsymbol{n(t)}$ in our measurement $\boldsymbol{x(t)}$, the matched filter acts as:

$$\boldsymbol{y(t)} = \int_{-\infty}^{\infty} \boldsymbol{x(\tau)}s(\tau - t)d\tau = \boldsymbol{x(t)} * s(-t) \tag{A.34}$$

So we see that the matched filter simply acts as a *cross-correlation* between the known (broadcast) signal $s(t)$ and the measurement $\boldsymbol{x(t)}$, and it will peak at the time in the measurement at which the known signal occurs. This is equivalent to a convolution of the measurement against a time-reversed (and conjugated if $s(t)$ is complex) version of the known signal $s(t)$. For simplicity, the 'canonical' matched filter is set so that $t_0 = 0$, resulting in a transfer function from Eq. A.22 of $H(\omega) = S^*(\omega)$. The matched filter output in the frequency domain is thus:

$$\boldsymbol{Y(\omega)} = \boldsymbol{X(\omega)}S^*(\omega) \tag{A.35}$$

Finally, for practical purposes, the matched filter for a discrete, digitized signal is given by:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]s[k-n] \tag{A.36}$$

Again, this calculation is more easily computed by taking the DFT of the measurement and known signal and multiplying in the frequency domain:

$$y[n] = x[n] * s[-n] \quad \Leftrightarrow \quad Y[\omega] = X[\omega]S^*[\omega] \tag{A.37}$$

Bibliography

[1] R.A. Brooks, A. M. Flynn, *Fast Cheap and Out of Control: A Robot Invasion of the Solar System*. Journal of the British Interplanetary Society, vol. 42, pp. 478-485, 1989.

[2] R.A. Brooks, *A Robust Layered Control System for a Mobile Robot*. IEEE Journal on Robotics and Automation, vol. 2, no. 1, pp. 14-23, 1986.

[3] C.V. Alt, *Autonomous Underwater Vehicles*. Meeting of the Autonomous Underwater Lagrangian Platforms and Sensors Workshop, 2003.

[4] D.R. Bildberg, *The Development of Autonomous Underwater Vehicles (AUV); a Brief Summary*. IEEE International Conference on Robotics and Automation, 2001.

[5] H.R. Widditsch, *SPURV - The First Decade*. Technical Report, The University of Washington Applied Physics Lab, 1973.

[6] P. Moorhouse, *A Modern History of the Manned Submersible*. Marine Technology Society Journal, no. 49, pp. 65-78, 2015.

[7] R. Christ, R. Wernli, *The ROV Manual*. Waltham, MA: Butterworth-Heinemann, Elsevier, 2013.

[8] R.D. Ballard, *The MEDEA/JASON Remotely Operated Vehicle System*. Deep Sea Research Part I: Oceanographic Research Papers, vol. 40, iss. 8, pp. 1673-1687, 1993.

[9] R. Capocci, et al., *Inspection-Class Remotely Operated Vehicles – A Review*. Journal of Marine Science and Engineering, vol. 5, iss. 1, pp. 1-32, 2017.

[10] C. Mai, et al., *Subsea Infrastructure Inspection: A Review Study*. IEEE International Conference on Underwater System Technology: Theory and Applications, pp. 71-76, 2016.

[11] M. Breivik, T.I. Fossen, *Guidance Laws for Autonomous Underwater Vehicles*. Intelligent Underwater Vehicles, Vienna, Austria: I-Tech Education and Publishing, ch. 4, pp. 51-76, 2009.

[12] R.B. Wynn, et al., *Autonomous Underwater Vehicles (AUVs): Their Past, Present and Future Contributions to the Advancement of Marine Geoscience*. Marine Geology, vol. 352, pp. 451-468, 2014.

[13] D.L. Rudnick, et al., *Underwater Gliders for Ocean Research*. Marine Technology Society Journal, vol. 38, no. 1, pp. 48-59, 2004.

[14] A.D. Bowen, et al., *The Nereus Hybrid Underwater Robotic Vehicle for Global Ocean Science Operations to 11,000m Depth*. IEEE OCEANS, pp. 1-10, 2008.

[15] B. Johansson, et al., *Seaeye Sabertooth A Hybrid AUV/ROV Offshore System*. IEEE OCEANS, pp. 1-3, 2010.

[16] J. Yuh, G. Marani, D.R. Blidberg, *Applications of Marine Robotic Vehicles*. Intelligent Service Robotics, vol. 4, pp. 221-231, 2011.

[17] S. Reed, Y. Petillot, J. Bell, *An Automatic Approach to the Detection and Extraction of Mine Features in Sidescan Sonar*. IEEE Journal of Oceanic Engineering, vol. 28, no. 1, pp. 90-105, 2003.

[18] P. Chapple, *Automated Detection and Classification in High-Resolution Sonar Imagery for Autonomous Underwater Vehicle Operations*. Technical Report, Defence Science and Technology Organisation, Maritime Operations Division, 2002.

[19] P. Blondel, L.M. Parson and V. Robigou, *TexAn: Textural Analysis of Sidescan Sonar Imagery and Generic Seafloor Characterisation*. IEEE OCEANS, pp. 419-423, 2003.

[20] V.L. Lucieer, *Object-Oriented Classification of Sidescan Sonar Data for Mapping Benthic Marine Habitats*. International Journal of Remote Sensing, vol. 29, no. 3, pp. 905-921, 2008.

[21] H. Kumagai, *Hydrothermal Plumes Imaged by High-Resolution Side-Scan Sonar on a Cruising AUV, Urashima*. Geochemistry, Geophysics, Geosystems, vol. 11, iss. 12, pp. 1-8, 2010.

[22] M. Purcell, at al., *Use of REMUS 6000 AUVs in the Search for the Air France Flight 447*. IEEE OCEANS, pp. 1-7, 2011.

[23] P.K. LeHardy, C. Moore, *Deep Ocean Search for Malaysia Airlines Flight 370*. IEEE OCEANS, pp. 1-4, 2014.

[24] M. Grasmueck, et al., *Autonomous Underwater Vehicle (AUV) Mapping Reveals Coral Mound Distribution, Morphology, and Oceanography in Deep Water of the Straits of Florida*. Geophysical Research Letters, vol. 33, iss. 23, pp. 1-6, 2006.

[25] P. Wadhams, M.J. Doble, *Digital Terrain Mapping of the Underside of Sea Ice from a Small AUV*. Geophysical Research Letters, vol. 35, iss. 1, pp. 1-6, 2008.

[26] G. Williams, et al., *Thick and Deformed Antarctic Sea Ice Mapped with Autonomous Underwater Vehicles*. Nature Geoscience, vol. 8, pp. 61-67, 2015.

[27] R. Yeo, *Surveying the Underside of an Arctic Ice Ridge Using a Man-Portable GAVIA AUV Deployed Through the Ice*. IEEE OCEANS, pp. 1-8, 2007.

[28] D.W. Caress, et al., *High-Resolution Multibeam, Sidescan, and Subbottom Surveys using the MBARI AUV D. Allan B*. Technical Report, Marine Habitat Mapping Technology for Alaska, Alaska Sea Grant College Program, University of Alaska Fairbanks, pp. 47-69, 2008.

[29] B. Bingham, et al., *Robotic Tools for Deep Water Archaeology: Surveying an Ancient Shipwreck with an Autonomous Underwater Vehicle*. Journal of Field Robotics, vol. 27, iss. 6, pp. 702-717, 2010.

[30] B. Foley, D. Mindell, *Precision Survey and Archaeological Methodology in Deep Water*. The Journal of the Hellenic Institute of Marine Archaeology, vol. VI, pp. 49-56, 2002.

[31] H. Singh, et al., *Imaging Underwater for Archaeology*. Journal of Field Archaeology, vol. 27, no. 3, pp. 319-328, 2000.

[32] H. Singh, et al., *Imaging Coral I: Imaging Coral Habitats with the SeaBED AUV*. Subsurface Sensing Technologies and Applications, vol. 5, iss. 1, pp. 25-42, 2004.

[33] C. Roman, G. Inglis, J. Rutter, *Application of Structured Light Imaging for High Resolution Mapping of Underwater Archaeological Sites*. IEEE OCEANS, pp. 1-9, 2010.

[34] D.S. Kelley, et al., *A Serpentinite-Hosted Ecosystem: The Lost City Hydrothermal Field*. Science, vol. 307, iss. 5714, pp. 1428-1434, 2005.

[35] A. Bodenmann, B. Thornton, T. Ura, *Generation of High-Resolution Three-Dimensional Reconstructions of the Seafloor in Color using a Single Camera and Structured Light*. Journal of Field Robotics, vol. 34, iss. 5, pp. 833-851, 2016.

[36] J.S. Willcox, et al., *Performance Metrics for Oceanographic Surveys with Autonomous Underwater Vehicles*. IEEE Journal of Oceanic Engineering, vol. 26, no. 4, pp. 711-725, 2001.

[37] T.B. Curtin, et al., *Autonomous Oceanographic Sampling Networks*. Oceanography, vol. 6, no. 3, pp. 86-94, 1993.

[38] E. Fiorelli, et al., *Multi-AUV Control and Adaptive Sampling in Monterey Bay*. IEEE Journal of Oceanic Engineering, vol. 31, no. 4, pp. 935-948, 2006.

[39] N.E. Leonard, et al., *Collective Motion, Sensor Networks, and Ocean Sampling*. IEEE Proceedings, vol. 95, no. 1, pp. 48-74, 2007.

[40] R. Camilli, et al., *Integrating In-Situ Chemical Sampling with AUV Control Systems*. IEEE OCEANS, vol. 1, pp. 101-109, 2004.

[41] W. Li, et al., *Moth-Inspired Chemical Plume Tracing on an Autonomous Underwater Vehicle*. IEEE Transactions on Robotics, vol. 22, no. 2, pp. 292-307, 2006.

[42] D.R. Yoerger, et al., *Autonomous and Remotely Operated Vehicle Technology for Hydrothermal Vent Discovery, Exploration, and Sampling*. Oceanography, vol. 20, no. 1, pp. 152-161, 2007.

[43] J. Das, et al., *Coordinated Sampling of Dynamic Oceanographic Features with Underwater Vehicles and Drifters*. International Journal of Robotics Research, vol. 31, no. 5, pp. 626-646, 2012.

[44] R. Camilli, et al., *Tracking Hydrocarbon Plume Transport and Biodegradation at Deepwater Horizon*. Science, vol. 330, iss. 6001, pp. 201-204, 2010.

[45] Y. Zhang, et al., *Thermocline Tracking Based on Peak-Gradient Detection by an Autonomous Underwater Vehicle*. IEEE OCEANS, pp. 1-4, 2010.

[46] S. Petillo, A. Balasuriya, H. Schmidt, *Autonomous Adaptive Environmental Assessment and Feature Tracking via Autonomous Underwater Vehicles*. IEEE OCEANS, pp. 1-9, 2010.

[47] H. Schmidt, et al., *Real-Time Frontal Mapping with AUVs in a Coastal Environment*. IEEE OCEANS, vol. 3, pp. 1094-1098, 1998.

[48] S. Petillo, et al., *Autonomous & Adaptive Oceanographic Front Tracking On Board Autonomous Underwater Vehicles*. IEEE OCEANS, pp. 1-10, 2015.

[49] S. Petillo, H. Schmidt, A. Balasuriya, *Constructing a Distributed AUV Network for Underwater Plume-Tracking Operations*. International Journal of Distributed Sensor Networks, pp. 1-12, 2012.

[50] A.J. Poulsen, D.P. Eickstedt, J.P. Ianniello, *Bearing Stabilization and Tracking for an AUV with an Acoustic Line Array*. IEEE OCEANS, pp. 1-6, 2006.

[51] C.M. Clark, et al., *Tracking and Following a Tagged Leopard Shark with an Autonomous Underwater Vehicle*. Journal of Field Robotics, vol. 30, iss. 3, pp. 309-322, 2013.

[52] M.J. Hamilton, S.K. Kenma, D. Hughes, *Antisubmarine Warfare Applications for Autonomous Underwater Vehicles: The GLINT09 Sea Trial Results*. Journal of Field Robotics, vol. 27, no. 6, pp. 890-902, 2006.

[53] J.R. Edwards, H. Schmidt, K.D. LePage, *Bistatic Synthetic Aperture Target Detection and Imaging With an AUV*. IEEE Journal of Oceanic Engineering, vol. 26, no. 4, pp. 690-699, 2001.

[54] T.C. Liu, H. Schmidt, *AUV-Based Seabed Target Detection and Tracking*. IEEE OCEANS, pp. 474-478, 2002.

[55] D.P. Eickstedt, H. Schmidt, *A Low-Frequency Sonar for Sensor-Adaptive, Multistatic, Detection and Classification of Underwater Targets with AUVs*. IEEE OCEANS, pp. 1440-1447, 2003.

[56] R. Chen, A. Poulsen, H. Schmidt, *Spectral, Spatial, and Temporal Characteristics of Underwater Ambient Noise in the Beaufort Sea in 1994 and 2016*. The Journal of the Acoustical Society of America, vol. 145, no. 2, pp. 605-614, 2019.

[57] S. Petillo, *Autonomous & Adaptive Oceanographic Feature Tracking On Board Autonomous Underwater Vehicles*. Thesis: Ph.D., Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, Joint Program in Oceanography/Applied Ocean Science and Engineering, 2015.

[58] H. Schmidt, J.G. Bellingham, J.W. Bales, *Mobile Underwater Arrays*. U.S. Patent, US5894450A, 1997.

[59] N.R. Rypkema, H. Schmidt, *Formation Control of a Drifting Group of Marine Robotic Vehicles*. Distributed Autonomous Robotic Systems, pp. 633-647, 2018.

[60] N.R. Rypkema, *Distributed Autonomy and Formation Control of a Drifting Swarm of Autonomous Underwater Vehicles*. Thesis: S.M., Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, Joint Program in Applied Ocean Science and Engineering, 2015.

[61] I. Schjolberg, et al., *Next Generation Subsea Inspection, Maintenance and Repair Operations*. IFAC Conference on Control Applications in Marine Systems, vol. 49, iss. 23, pp. 434-439, 2016.

[62] S. Negahdaripour, P. Firoozfam, *An ROV Stereovision System for Ship-Hull Inspection*. IEEE Journal of Oceanic Engineering, vol. 31, no. 3, pp. 551-564, 2006.

[63] P.V. Teixeira, et al., *Underwater Inspection Using Sonar-Based Volumetric Submaps*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4288-4295, 2016.

[64] G. Hollinger, et al., *Uncertainty-Driven View Planning for Underwater Inspection*. IEEE International Conference on Robotics and Automation, pp. 4884-4891, 2012.

[65] P. Ridao, et al., *Visual Inspection of Hydroelectric Dams Using an Autonomous Underwater Vehicle*. Journal of Field Robotics, vol. 27, iss. 6, pp. 759-778, 2010.

[66] Y.R. Petillot, S.R. Reed, J.M. Bell, *Real Time AUV Pipeline Detection and Tracking Using Side Scan sonar and Multi-Beam Echo-Sounder*. IEEE OCEANS, pp. 217-222, 2002.

[67] A. Bagnitsky, at al., *Side Scan Sonar Using for Underwater Cables & Pipelines Tracking by means of AUV*. IEEE Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, pp. 1-10, 2011.

[68] E.M. Fischell, *Characterization of Underwater Target Geometry from Autonomous Underwater Vehicle Sampling of Bistatic Acoustic Scattered Fields*. Thesis: Ph.D., Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, Joint Program in Applied Ocean Science and Engineering, 2015.

[69] G. M. Trimble, *The Doppler Inertial Acoustic System For Littoral Navigation (DIAS)*. IEEE Autonomous Underwater Vehicles, pp. 27-33, 1998.

[70] M. B. Larsen, *High Performance Doppler-Inertial Navigation - Experimental Results*. IEEE OCEANS, pp. 1449-1456, 2000.

[71] S. Byrne, V.E. Schmidt, *Uncertainty Modeling for AUV Acquired Bathymetry*. U.S. Hydrographic Conference, pp. 1-24, 2015.

[72] N.A. Brokloff, *Matrix Algorithm for Doppler Sonar Navigation*. IEEE OCEANS, vol. 3, pp. 378-383, 1994.

[73] J. Snyder, *Doppler Velocity Log (DVL) Navigation for Observation-Class ROVs*. IEEE OCEANS, pp. 1-9, 2010.

[74] O. Hegrenas, E. Berglund, O. Hallingstad *Model-Aided Inertial Navigation for Underwater Vehicles*. IEEE International Conference on Robotics and Automation, pp. 1069-1076, 2008.

[75] J.A. Farrell, *Aided Navigation Systems: GPS and High Rate Sensors*. New York: McGraw-Hill, ch. 11-12, 2008.

[76] B. Jalving, et al., *DVL Velocity Aiding in the HUGIN 1000 Integrated Inertial Navigation System*. Modeling, Identification and Control, vol. 25, pp. 223-235, 2004.

[77] L. Paull, et al., *AUV Navigation and Localization - A Review*. IEEE Journal of Oceanic Engineering, vol. 39, no. 1, pp. 131-149, 2014.

[78] L. Stutters, et al., *Navigation Technologies for Autonomous Underwater Vehicles*. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews, vol. 38, no. 4, 2008.

[79] J.C. Kinsey, R.M. Eustice, L.L. Whitcomb, *A Survey of Underwater Vehicle Navigation: Recent Advances and New Challenges*. IFAC Conference of Manoeuvering and Control of Marine Craft, vol. 88, pp. 1-12, 2006.

[80] S. Carreno, et al., *A Survey on Terrain Based Navigation for AUVs*. IEEE OCEANS 2010, pp. 1-7, 2010.

[81] R.M. Eustice, O. Pizarro, H. Singh, *Visually Augmented Navigation for Autonomous Underwater Vehicles*. IEEE Journal of Oceanic Engineering, vol. 33, no. 2, pp. 103-122, 2008.

[82] I.T. Ruiz, S. de Raucourt, Y. Petillot, *Concurrent Mapping and Localization Using Sides-can Sonar*. IEEE Journal of Oceanic Engineering, vol. 29, no. 2, pp. 442-456, 2004.

[83] D. Ribas, et al., *SLAM using an Imaging Sonar for Partially Structured Underwater Environments*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5040-5045, 2006.

[84] M. Fallon, et al., *Efficient AUV Navigation Fusing Acoustic Ranging and Side-Scan Sonar*. IEEE International Conference on Robotics and Automation, pp. 2398-2405, 2011.

[85] P. Newman, J. Leonard, *Pure Range-Only Sub-Sea SLAM*. IEEE International Conference on Robotics and Automation, pp. 1921-1926, 2003.

[86] J. Vaganay, et al., *Experimental Validation of the Moving Long Base-Line Navigation Concept*. IEEE Autonomous Underwater Vehicles, pp. 59-65, 2004.

[87] J. Curcio, et al., *Experiments in Moving Baseline Navigation Using Autonomous Surface Craft*. IEEE OCEANS, 2005.

[88] J. Melo, A. Matos, *Towards LBL Positioning Systems for Multiple Vehicles*. IEEE OCEANS, 2016.

[89] B. Hodgkinson, D. Shyu, K. Mohseni, *Acoustic Source Localization System Using a Linear Arrangement of Receivers for Small Unmanned Underwater Vehicles*. IEEE OCEANS, 2012.

[90] M. Morgado, P. Oliveira, C. Silvestre, *Tightly Coupled Ultrashort Baseline and Inertial Navigation System for Underwater Vehicles: An Experimental Validation*. Journal of Field Robotics, vol. 30, no. 1, pp. 142-170, 2013.

[91] R.M. Eustice, et al., *Recent Advances in Synchronous-Clock One-Way-Travel-Time Acoustic Navigation*. IEEE OCEANS, 2006.

[92] S.E. Webster, et al., *Advances in Single-Beacon One-Way-Travel-Time Acoustic Navigation for Underwater Vehicles*. The International Journal of Robotics Research, vol. 31, pp. 935-950, 2012.

[93] B. Claus, et al., *Closed-Loop One-Way-Travel-Time Navigation Using Low-Grade Odometry for Autonomous Underwater Vehicles*. Journal of Field Robotics, vol. 35, iss. 4, pp. 421-434, 2017.

[94] M.V. Jakuba, et al., *Long-Baseline Acoustic Navigation for Under-Ice Autonomous Underwater Vehicle Operations*. Journal of Field Robotics, vol. 25, iss. 11-12, pp. 861-879, 2008.

[95] M.V. Jakuba, et al., *Feasibility of Low-Power One-Way Travel-Time Inverted Ultra-Short Baseline Navigation*. IEEE OCEANS, pp. 1-10, 2015.

[96] W.L. Garfield, *TACAN: A Navigation System for Aircraft*. IEE Proceedings Part B: Radio and Electronic Engineering, vol. 105, iss. 9S, pp. 298-306, 1958.

[97] O.M. Cliff, et al., *Online Localization of Radio-Tagged Wildlife with an Autonomous Aerial Robot System*. Robotics: Science and Systems, pp. 1-9, 2015.

[98] A. Posch, S. Sukkarieh, *UAV Based Search for a Radio Tagged Animal Using Particle Filters*. ARAA Australian Conference on Robotics and Automation, pp. 107-112, 2009.

[99] J.T. Isaacs, et al., *Quadrotor Control for RF Source Localization and Tracking*. International Conference on Unmanned Aircraft Systems, pp. 244-252, 2014.

[100] L. Dressel, M.J. Kochenderfer, *Efficient and Low-Cost Localization of Radio Signals with a Multirotor UAV*. AIAA Guidance, Navigation, and Control Conference, pp. 1-17, 2018.

[101] J.M. Valin, F. Michaud, J. Rouat, *Robust Localization and Tracking of Simultaneous Moving Sound Sources Using Beamforming and Particle Filtering*. Robotics and Autonomous Systems, vol. 55, iss. 3, pp. 216-228, 2007.

[102] C.T. Kim, et al., *Robust Estimation of Sound Direction for Robot Interface*. IEEE International Conference on Robotics and Automation, pp. 3475-3480, 2008.

[103] F. Grondin, F. Michaud, *Time Difference of Arrival Estimation Based on Binary Frequency Mask for Sound Source Localization on Mobile Robots*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 6149-6154, 2015.

[104] A. Badali, et al., *Evaluating Real-Time Audio Localization Algorithms for Artificial Audition in Robotics*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2033-2038, 2009.

[105] H. Liu, M. Shen, *Continuous Sound Source Localization based on Microphone Array for Mobile Robots*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4332-4339, 2010.

[106] Y. Cho, et al., *Sound Source Localization for Robot Auditory Systems*. IEEE Transactions on Consumer Electronics, vol. 55, iss. 3 pp. 1663-1668, 2009.

[107] I. Markovic, I. Petrovic, *Speaker Localization and Tracking with a Microphone Array on a Mobile Robot using von Mises Distribution and Particle Filtering*. Robotics and Autonomous Systems, vol. 58, iss. 11, pp. 1185-1196, 2010.

[108] K. Sekiguchi, et al., *Online Simultaneous Localization and Mapping of Multiple Sound Sources and Asynchronous Microphone Arrays*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1973-1979, 2016.

[109] J.S. Hu, et al., *Simultaneous Localization of Mobile Robot and Multiple Sound Sources Using Microphone Array*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 29-34, 2009.

[110] C. Evers, P.A. Naylor, *Acoustic SLAM*. IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 9, pp. 1484-1498, 2018.

[111] S. Argentieri, P. Danes, P. Soueres, *A Survey on Sound Source Localization in Robotics: From Binaural to Array Processing Methods*. Computer Speech & Language, vol. 34, iss. 1, pp. 87-112, 2015.

[112] C. Rascon, I. Meza, *Localization of Sound Sources in Robotics: A Review*. Robotics and Autonomous Systems, vol. 96, pp. 184-210, 2017.

[113] Q.H. Wang, T. Ivanov, P. Aarabi, *Acoustic Robot Navigation Using Distributed Microphone Arrays*. Information Fusion, vol. 5, iss. 2, pp. 131-140, 2004.

[114] S. Ogiso, et al., *Self-Localization Method for Mobile Robot Using Acoustic Beacons*. ROBOMECH, vol. 2, no. 1, pp. 1-12, 2015.

[115] Y. Lin, et al., *Cooperative Relative Robot Localization with Audible Acoustic Sensing*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 662-667, 2005.

[116] M. Basiri, et al., *On-Board Relative Bearing Estimation for Teams of Drones Using Sound*. IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 820-827, 2016.

[117] M. Basiri, et al., *Robust Acoustic Source Localization of Emergency Signals from Micro Air Vehicles*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4737-4742, 2012.

[118] M. Basiri, et al., *Audio-Based Relative Positioning System for Multiple Micro Air Vehicle Systems*. Robotics: Science and Systems, pp. 1-8, 2013.

[119] M. Basiri, et al., *Audio-Based Localization for Swarms of Micro Air Vehicles*. IEEE Conference on Robotics and Automation, pp. 4729-4734, 2014.

[120] S.V. Schell, W.A. Gardner, *High-Resolution Direction Finding*. Handbook of Statistics, vol. 10, pp. 755-817, 2014.

[121] B.D.V. Veen, K.M. Buckley, *Beamforming: A Versatile Approach to Spatial Filtering*. IEEE ASSP magazine, vol. 5, iss. 2, pp. 4-24, 1988.

[122] J. Capon, *High-Resolution Frequency-Wavenumber Spectrum Analysis*. IEEE Proceedings, vol. 57, iss. 8, pp. 1408-1418, 1969.

[123] R.O. Schmidt, *Multiple Emitter Location and Signal Parameter Estimation*. IEEE Transactions on Antennas and Propagation, vol. 34, no. 3, pp. 276-280, 1986.

[124] A. Paulraj, R. Roy, T. Kailath, *Estimation of Signal Parameters via Rotational Invariance Techniques- ESPRIT*. 19th Asilomar Conference on Circuits, Systems and Computers, pp. 83-89, 1985.

[125] J.M. Valin, et al., *Robust Sound Source Localization Using a Microphone Array on a Mobile Robot*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1228-1233, 2003.

[126] P. Svaizer, M. Matassoni and M. Omologo, *Acoustic Source Location in a Three-Dimensional Space Using Crosspower Spectrum Phase*. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 231-234, 1997.

[127] J.H. DiBiase, *A High-Accuracy, Low-Latency Technique for Talker Localization in Reverberant Environments Using Microphone Arrays*. Thesis: Ph.D., Brown University, Division of Engineering, 2000.

[128] J.H. DiBiase, H.F. Silverman, M.S. Brandstein, *Robust Localization in Reverberant Rooms*. Microphone Arrays, chap. 8, pp. 157-180, Berlin, Heidelberg: Springer, 2001.

[129] J. Chen, J. Benesty, Y. Huang, *Time Delay Estimation in Room Acoustic Environments: An Overview*. EURASIP Journal on Applied Signal Processing, vol. 1, pp. 1-19, 2006.

[130] U.R.O. Nickel, *Subarray Configurations for Digital Beamforming with Low Sidelobes and Adaptive Interference Suppression*. Proceedings of International Radar Conference, pp. 714-719, 1995.

[131] C.D. Bailey, D.D. Aalfs, *Design of Digital Beamforming Subarrays for a Multifunction Radar*. International Radar Conference, pp. 1-6, 2009.

[132] D. Wang, H. Hu, Z. Yang, *Improved Genetic Algorithm for the Configuration Optimization of the Sub Arrays in Phased Array Radar*. 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, pp. 930-934, 2016.

[133] K.V. Ramachandra, *Kalman Filtering Techniques for Radar Tracking*. CRC Press, 2000.

[134] F. Gustafsson, et al., *Particle Filters for Positioning, Navigation, and Tracking*. IEEE Transactions on Signal Processing, vol. 50, iss. 2, pp. 425-437, 2002.

[135] B. Ristic, S. Arulampalam, N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Radar Library, 2004.

[136] L.D. Stone, et al., *Bayesian Multiple Target Tracking*. Artech House Inc., 2013.

[137] P.B. Choppala, *Bayesian Multiple Target Tracking*. Thesis: Ph.D., Victoria University of Wellington, 2014.

[138] R. Chen, J.S. Liu, *Mixture Kalman Filters*. Journal of the Royal Statistical Society Series B, vol. 62, iss. 3, pp. 493-508, 2000.

[139] W.I. Tam, *Tracking Filters for Radar Systems*. Thesis: S.M., University of Toronto, Department of Electrical and Computer Engineering, 1997.

[140] R.P.S. Mahler, *Multitarget Bayes Filtering via First-Order Multitarget Moments*. IEEE Transactions on Aerospace and Electronic Systems, vol. 39, iss. 4, pp. 1152-1178, 2003.

[141] B.N. Vo, W.K.M. Ma, *The Gaussian Mixture Probability Hypothesis Density Filter*. IEEE Transactions on Signal Processing, vol. 54, iss. 11, pp. 4091-4104, 2006.

[142] J.F. Moura, *Narrow-Band Passive Systems Theory with Applications to Positioning and Navigation*. Thesis: Ph.D., Massachusetts Institute of Technology, Research Laboratory of Electronics, 1976.

[143] D.A. Paley, F. Zhang, N.E. Leonard *Cooperative Control for Ocean Sampling: The Glider Coordinated Control System*. IEEE Transactions on Control Systems Technology, vol. 16, no. 4, pp. 735-744, 2008.

[144] N.E. Leonard, et al., *Coordinated Control of an Underwater Glider Fleet in an Adaptive Ocean Sampling Field Experiment in Monterey Bay*. Journal of Field Robotics, vol. 27, iss. 6, pp. 718-740, 2010.

[145] J. Das, et al., *Towards Mixed-Initiative, Multi-Robot Field Experiments: Design, Deployment, and Lessons Learned*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1-8, 2011.

[146] A. Branch, et al. *Front Delineation and Tracking with Multiple Underwater Vehicles*. Journal of Field Robotics, vol. 36, iss. 3, pp. 568-586, 2018.

[147] Y. Lin, et al. *A Multi-Autonomous Underwater Vehicle System for Autonomous Tracking of Marine Life*. Journal of Field Robotics, vol. 34, iss. 4, pp. 757-774, 2017.

[148] J.M. Soares, et al., *Joint ASV/AUV Range-Based Formation Control: Theory and Experimental Results*. IEEE International Conference on Robotics and Automation, pp. 5579-5585, 2013.

[149] S. Petillo, H. Schmidt, *Exploiting Adaptive and Collaborative AUV Autonomy for Detection and Characterization of Internal Waves*. IEEE Journal of Oceanic Engineering Special Issue on Marine Vehicle Autonomy, vol. 39, no. 1, pp. 150-164, 2014.

[150] J.M. Walls, et al., *Cooperative Localization by Factor Composition over a Faulty Low-Bandwidth Communication Channel*. IEEE International Conference on Robotics and Automation, pp. 401-408, 2015.

[151] P. Rigby, et al., *Towards Geo-Referenced AUV Navigation Through Fusion of USBL and DVL Measurements*. IEEE OCEANS, 2006.

[152] N.R. Rypkema, E.M. Fischell, H. Schmidt, *One-Way Travel-Time Inverted Ultra-Short Baseline Localization for Low-Cost Autonomous Underwater Vehicles*. IEEE International Conference on Robotics and Automation, pp. 4920-4926, 2017.

[153] N.R. Rypkema, E.M. Fischell, H. Schmidt, *Closed-Loop Single-Beacon Passive Acoustic Navigation for Low-Cost Autonomous Underwater Vehicles*. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 641-648, 2018.

[154] E.M. Fischell, N.R. Rypkema, H. Schmidt, *Relative Autonomy and Navigation for Command and Control of Low-Cost Autonomous Underwater Vehicles*. IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1800-1806, 2019.

[155] A. Anderson, et al., *An Overview of MIT-Olin's Approach in the AUVSI RobotX Competition*. Field and Service Robotics, pp. 61-80, 2016.

[156] M.R. Benjamin, et al., *Nested Autonomy for Unmanned Marine Vehicles with MOOS-IvP*. Journal of Field Robotics, vol. 27, no. 6, pp. 834-875, 2010.

[157] C.H. Knapp, G. C. Carter, *The Generalized Correlation Method for Estimation of Time Delay*. IEEE Transactions on Acoustics, Speech and Signal Processing, pp. 320-327, 1976.

[158] H.L. Van Trees, *Optimum Array Processing*. John Wiley & Sons, 2002.

[159] L.R. Rabiner, B. Gold, *Theory and Application of Digital Signal Processing*. Prentice-Hall, pp. 393-399, 1975.

[160] B. Allen, et al., *FINDCHIRP: An Agorithm for Detection of Gravitational Waves from Inspiraling Compact Binaries*. Physical Review D, vol. 85, no. 12, art. 122006, 2012.

[161] W. Premerlani, P. Bizard, *Direction Cosine Matrix IMU: Theory* [Online]. Available: https://owenson.me/build-your-own-quadcopter-autopilot/DCMDraft2.pdf

[162] M. Quigley, et al., *ROS: an Open-Source Robot Operating System*. IEEE International Conference on Robotics and Automation Workshop on Open Source Software, 2009.

[163] P.M. Woodward, *Probability and Information Theory with Applications to Radar*. New York: McGraw-Hill, 1953.

[164] E. Fernandez, D. Calero, M.E. Pares, *CSAC Characterization and its Impact on GNSS Clock Augmentation Performance*. Sensors, vol. 17, no. 370, pp. 1-19, 2017.

[165] A.T. Gardner, J.A. Collins, *A Second Look at Chip Scale Atomic Clocks for Long Term Precision Timing*. IEEE OCEANS, 2016.

[166] R. Lutwak, et al., *The Chip-Scale Atomic Clock - Prototype evaluation*. Proceedings of the 39th Annual Precise Time and Time Interval Meeting, pp. 269-290, 2007.

[167] S. Knappe, *MEMS Atomic Clocks*. Comprehensive Microsystems, Maryland Heights: Elsevier, vol. 3, pp. 571-612, 2007.

[168] J. Kitching, *Chip-Scale Atomic Devices*. Applied Physics Reviews, vol. 5, 2018.

[169] *U.S. Army Manufacturing Technology Program: Chip Scale Atomic Clock (CSAC)* [Online]. Available: https://www.dodmantech.com/mantechprograms/Files/Army/ArmySuccess_ChipScaleAtomicClock_02Feb15.pdf

[170] Y. Kim, C.P. Walter, *Retrace and Disciplining Time Constant Effects on Holdover Clock Drifts in Chip-Scale Atomic Clock*. Joint Conference of the European Frequency and Time Forum and IEEE International Frequency Control Symposium, pp. 310-314, 2017.

[171] L. Wainstein, V. Zubakov, *Extraction Of Signals From Noise*. Englewood Cliffs, NJ: Prentice-Hall, 1962.

[172] H. Krim, M. Viberg, *Two Decades of Array Signal Processing Research: the Parametric Approach*. IEEE Signal Processing Magazine, vol. 13, no. 4, pp. 67-94, 1996.

[173] H. Elkachouchi and M. Abd elsalam Mofeed, *Direction-of-Arrival Methods (DOA) and Time Difference of Arrival (TDOA) Position Location Technique*. 27nd National Radio Science Conference, pp. 173-182, 2005.

[174] T.E. Tuncer, B. Friedlander, *Classical and Modern Direction-of-Arrival Estimation*. Burlington, MA: Elsevier, 2009.

[175] S. Thrun, W. Burgard, D. Fox *Probabilistic Robotics*. Cambridge, MA: The MIT Press, 2006.

[176] T.I. Fossen, *Guidance and Control of Ocean Vehicles*. New York, NY: John Wiley & Sons, 2004.

[177] T.I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. New York, NY: John Wiley & Sons, 2011.

[178] J. Kim, et al., *Estimation of Hydrodynamic Coefficients for an AUV Using Nonlinear Observers*. IEEE Journal of Oceanic Engineering, vol. 27, no. 4, pp. 830-840, 2002.

[179] O. Hegrenas, O. Hallingstad, *Model-Aided INS With Sea Current Estimation for Robust Underwater Navigation*. IEEE Journal of Oceanic Engineering, vol. 36, no. 2, pp. 316-337, 2011.

[180] S.A.T. Randeni P., et al., *Implementation of a Hydrodynamic Model-Based Navigation System for a Low-Cost AUV Fleet*. IEEE OES Autonomous Underwater Vehicle Symposium, 2019.

[181] P. Stano, et al., *Parametric Bayesian Filters for Nonlinear Stochastic Dynamical Systems: A Survey*. IEEE Transactions on Cybernetics, vol. 43, no. 6, pp. 1607-1624, 2013.

[182] Z.S. Chen, *Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond*. Statistics: A Journal of Theoretical and Applied Statistics, 2003.

[183] P.M. Saunders, N.P. Fofonoff, *Conversion of Pressure to Depth in the Ocean*. Deep Sea Research and Oceanographic Abstracts, vol. 23, no. 1, pp. 109-111, 1976.

[184] R. Douc, O. Cappe, E. Moulines, *Comparison of Resampling Schemes for Particle Filtering*. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, pp. 64-69, 2005.

[185] G. Grisetti, et al., *A Tutorial on Graph-Based SLAM*. IEEE Intelligent Transportation Systems Magazine, vol. 2, no. 4, pp. 31-43, 2010.

[186] F. Dellaert, *Factor Graphs and GTSAM: a Hands-On Introduction*. Technical Report GT-RIM-CP&R-2012-002, 2012.

[187] M. Kaess, et al., *iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree*. International Journal of Robotics Research, vol. 31, no. 2, pp. 216-235, 2012.

[188] M.J.D. Powell, *A Hybrid Method for Nonlinear Equations*. Numerical Methods for Nonlinear Algebraic Equations, vol. 7, pp. 87-114, 1970.

[189] A. Underwood, C. Murphy, *Design of a Micro-AUV for Autonomy Development and Multi-Vehicle Systems*. IEEE OCEANS, pp. 1-6, 2017.

[190] A.B. Phillips, et al., *Agile Design of Low-Cost Autonomous Underwater Vehicles*. IEEE OCEANS, pp. 1-7, 2017.

[191] M.K. Yoder, *Lower Charles River Bathymetry : 108 Years of Fresh Water*. Thesis: S.B., Massachusetts Institute of Technology, Dept. of Earth, Atmospheric, and Planetary Sciences, 2017.

[192] F.B. Jensen, W.A. Kuperman, M.B. Porter, H. Schmidt, *Computational Ocean Acoustics*. London, U.K.: Springer-Verlag, 2011.

[193] M.B. Porter, Y. Liu, *Finite-Element Ray Tracing*. International Conference on Theoretical Computational Acoustics, vol. 2, pp. 947-956, 1995.

[194] M.B. Porter, *The KRAKEN Normal Mode Program* SACLANT Undersea Research Centre, La Spezia, Italy, Technical Report, 1991.

[195] H. Schmidt, F.B. Jensen, *A Full Wave Solution for Propagation in Multilayered Viscoelastic Media with Application to Gaussian Beam Reflection at Fluid-Solid Interfaces.* Journal of the Acoustical Society of America, vol. 77, pp. 813-825, 1985.

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

[197] H. Schmidt, T. Schneider, *Acoustic Communication and Navigation in the New Arctic — A Model Case for Environmental Adaptation.* IEEE Third Underwater Communications and Networking Conference, pp. 1-4, 2016.

[198] P.F.J. Lermusiaux, C.S. Chiu, *Four-Dimensional Data Assimilation for Coupled Physical-Acoustical Fields.* In Acoustic Variability, SACLANTCEN: Kluwer Academic Press, 2002.

[199] D. Wang, et al., *Acoustically Focused Adaptive Sampling and On-Board Routing for Marine Rapid Environmental Assessment.* Journal of Marine Systems, vol. 78, pp. S393-S407, 2008.

[200] D. Fourie, J. Leonard, M. Kaess, *A Nonparametric Belief Solution to the Bayes Tree.* International Conference on Intelligent Robots and Systems, pp. 2189-2196, 2016.

[201] I. Markovic, I. Petrovic, *Bearing-Only Tracking with a Mixture of von Mises Distributions.* IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 707-712, 2012.

[202] G. Bourmaud, et al. *Continuous-Discrete Extended Kalman Filter on Matrix Lie Groups Using Concentrated Gaussian Distributions.* Journal of Mathematical Imaging and Vision, vol. 51, no. 1, pp. 209-228, 2015.

[203] I. Markovic, et al. *On Wrapping the Kalman Filter and Estimating with the SO(2) Group.* IEEE International Conference on Information Fusion, pp. 2245-2250, 2016.

[204] D. F. Crouse, *Cubature/Unscented/Sigma Point Kalman Filtering with Angular Measurement Models.* IEEE International Conference on Information Fusion, pp. 1550-1557, 2015.

[205] D. Fourie, *Multi-Modal and Inertial Sensor Solutions to Navigation-type Factor Graphs.* Thesis: Ph.D., Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, Joint Program in Applied Ocean Science and Engineering, 2017.

[206] D. Manalang, J. R. Delaney, *Axial Seamount – Restless, Wired and Occupied: A Conceptual Overview of Resident AUV Operations and Technologies*. IEEE OCEANS, pp. 1-7, 2016.