Actuated Mobile Sensing in Distributed, Unstructured Environments

Andrew Tinka

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-190
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-190.html

December 1, 2014

Actuated Mobile Sensing in Distributed, Unstructured Environments

by

Andrew Brendan Tinka

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Associate Professor Alexandre M. Bayen, Chair
Professor Claire Tomlin
Professor Kristofer S. J. Pister
Professor J. Karl Hedrick

Fall 2012

Actuated Mobile Sensing in Distributed, Unstructured Environments

Abstract

Actuated Mobile Sensing in Distributed, Unstructured Environments

by

Andrew Brendan Tinka

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Science

University of California, Berkeley

Associate Professor Alexandre M. Bayen, Chair

Mobile sensor networks present opportunities for improved *in situ* sensing in complex hydrodynamic environments such as estuarial deltas. This dissertation considers the design and implementation of the mobile sensor network system that was built as part of the Floating Sensor Network project for use in the Sacramento-San Joaquin Delta in California over the 2007–2012 time period. Individual Lagrangian sensor units collect hydrodynamic state information, which is then transmitted to a centralized server and assimilated to produce a state estimate for the entire hydrodynamic system. Physical obstacles, including the shoreline and natural vegetation, present a major challenge to operating mobile sensors in estuarial environments. Actuated mobile sensors are shown to be a viable solution; appropriate control techniques allow these sensors to avoid obstacles, meet navigational goals, and still collect the Lagrangian data necessary for the sensing objective. Issues addressed include physical design, communication techniques for mobile sensor networks, control schemes for fleets of underactuated vehicles in unstructured flow environments, assimilation techniques for mobile Lagrangian data, and field experiments to validate and demonstrate the actuated mobile Lagrangian sensor concept.

To Kathryn Cannon Miller

for making it all make sense

To these from birth is Belief forbidden; from these till death is Relief afar.
They are concerned with matters hidden — under the earthline their altars are —
The secret fountains to follow up, waters withdrawn to restore to the mouth,
And gather the floods as in a cup, and pour them again at a city's drouth.

Kipling, *The Sons of Martha*

Jon Beard retrieves a drifter near Mandeville Island.

Acknowledgments

Leading the Floating Sensor Network team over the last six years has been one of the singular honors of my career. This dissertation would not have been possible without the talent, ingenuity, and dedication of the undergraduate and graduate student researchers who joined me on this strange and wonderful journey.

Nico van der Kolk, Jason Wexler, Andrew Spencer, Jonathan Ellithorpe, Nahi Ojeil, Tarek Ibrahim, Anwar Ghoche, Jean-Séverin Deckers, David Wood, Martin Deterre, Fabien Chraim, Dennis Chan, Matt Holland, Carlos Oroza, Colin Foe-Parker, Sébastien Diemers, Emmanuel Sevrin, Nadine Moacdieh, Pierre-Henri Reilhac, Jonathan Beard, Jean-Baptiste Gariel, Jean-Benoît Saint-Pierre, Jad Kabbarra, Leah Anderson, Jack Reilly, Kevin Weekly, Chiheng Huor, Nolan Wagener, and Brandon Wong all spent many hours in Room 3 in McLaughlin Hall to make this project a reality. To them I give my deepest gratitude.

Many of my colleagues within the Bayen research group devoted their time and energy to help make the drifter project succeed. From research collaborations and whiteboard sessions to field trips and hardware production marathons, I have always been able to count on

their support and enthusiastic help. Issam Strub, Qingfang Wu, Julie Percelay, Olli-Pekka Tossavainen, Dan Work, Christian Claudel, Sébastien Blandin, Mohammad Rafiee, Jérôme Thai, Saurabh Amin, Samitha Samaranayake, and Timothy Hunter have been an essential part of my success.

Trying to summarize Alexandre Bayen's role in my development as a researcher would be a futile task. Instead, here are a few examples of how he has been an essential part of my career. Alex created an environment of true collaboration and teamwork within his group. The constraints and objectives of academic research do not naturally lead to harmonious group efforts; Alex developed trust and confidence between the members of his team, allowing us to work together for everyone's benefit. As a research mentor, Alex led by example through the sometimes murky waters of academic engineering. He showed me how to maintain the balance between theory and practice: by focusing on the missing pieces in the research community's understanding of a topic, and finding innovative techniques to meet those needs. Finally, and most importantly, Alex resolved any conflicts between his role as a research group leader and his role as his students' advisor by always, without fail, placing his students' interests above his own. For this, and very much more, I owe him a great deal.

Other professors and their students have added their contributions to my research journey. Mark Stacey, and his students Maureen Downing-Kunz and Wayne Wagner, have supported the Floating Sensor Network development effort and been essential parts of field experiments. Patrick Saint-Pierre and Jean-Pierre Aubin showed me great hospitality in Paris and gave me a unique personal introduction to viability theory. Peter Schwartz and Eli Ateljevich of the Berkeley Lab and the Department of Water Resources have been long-time friends of the FSN project. Ian Mitchell hosted me at UBC in order to collaborate further on viability for reachability computations. João Sousa and his entire LSTS team came to California with their amazing autonomous submarine to take part in a joint experiment that led to one of my first publications. Sonia Martínez and her student Andrew Kwok collaborated with me on the Zermelo-Voronoi approach to multivehicle control.

Many staff members at UC Berkeley and other institutions made important contributions. Dee Korbel suffered many strange equipment requisition requests and odd purchasing card transactions. Dan Plumlee maintained the Geography truck and boat, and happily trained many students on the fine art of backing a boat trailer down a ramp without crashing. James Fitzgerald and Henry Fastenau at UC Davis ran motorboat operator training sessions with inimitable style and humor, without which our safety record would not be nearly as good. David Bell and his colleagues at the Romberg Tiburon Research Center of San Francisco State University provided essential logistical support during our more ambitious field experiments.

Direct financial support for my work came from CALFED, the National Science Foundation, and the National Sciences and Engineering Research Council of Canada.

Some of the work presented in this dissertation has been adapted from published journal articles. I gratefully acknowledge Taylor and Francis for their liberal policies on authors' rights, and the Institute for Computer Sciences, Social Informatics and Telecommunications

Engineering for their policy of Creative Commons licencing and author copyright retention, for enabling this integration of my previous work.

Throughout my time at Berkeley, I have been strengthened by the love, understanding, and support of Kathryn. She has been with me in good times and bad. There have been days when this journey was not easy or fun, and darker days when the pressures and stresses of this work have gotten the upper hand. We made it through, together.

In the end, everything I have done was made possible by my parents. They gave me a good home, and helped me develop a love of learning. From this, all else follows.

Contents

List of Figures

List of Tables

Chapter 1

Mobile Sensors for Estuarial Systems

We outline the need to study freshwater hydrodynamic systems, particularly rivers and estuaries, by briefly exploring the close relationships between freshwater resources and human activity. The Sacramento-San Joaquin Delta is introduced as a region of significant importance to California. Existing methods for studying large-scale open water bodies are summarized. An innovative approach, namely mobile Lagrangian sensing, is presented, and historical context for why these systems are used in oceanography but not in estuarial studies is provided. Some example applications for Lagrangian sensing in inland environments are presented. The chapter concludes with an outline of the work that will be presented in the rest of the dissertation and the overall framework of the study of mobile, actuated sensor systems.

1.1 Freshwater Resources

It is difficult to overstate the importance of freshwater to human civilization. Human uses of freshwater include drinking, irrigation, fish production, transportation, hydroelectric power, and waste disposal. Per-capita municipal consumption of water varies from 50 L/d to 100 L/d (litres per day) in developing nations to 300 L/d to 1000 L/d in industrially developed nations [102]. (Actual human *intake* of water ranges from 2 L/d to 4 L/d [89]; the rest is used for domestic purposes like waste disposal, washing, cooking, gardening, etc.) Agriculture, power generation, and other industries also require water inputs. Water *abstraction* (physical extraction or diversion of water) is 3.8×10^{15} L/yr [80] (approximately 1500 L/d per person). Abstraction can be accounted into the water cycle, but other impacts of human water use, like the use of waterways to dilute or carry away waste, are more difficult to track. Societal shifts towards urbanization and water-intensive agriculture, and the growing world population, will increase freshwater demand significantly over the next fifty years [94].

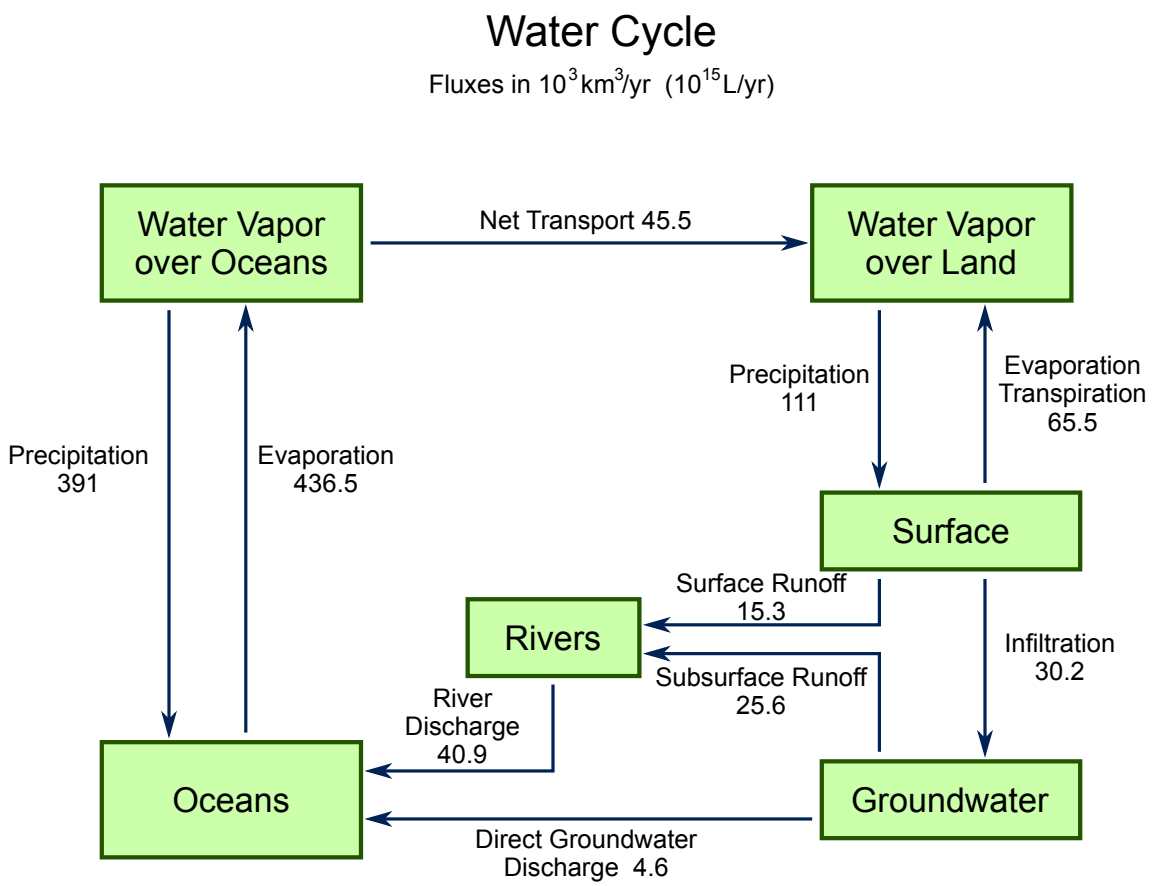

Figure 1.1: Simplified global water cycle, adapted from [80]. Numerical values are estimates, synthesized from many studies and models; values differ between authors.

1.1.1 The Significance of Rivers and Estuaries

A simplified view of the global water cycle is illustrated in Figure 1.1. Once water falls on land through precipitation, it either returns to the atmosphere through evapotranspiration, or to the sea via *runoff* processes. Almost all runoff starts as *surface* or *subsurface* runoff before combining in rivers and streams and travelling to the ocean (known as *streamflow*, *channel runoff*, or *river discharge*). Approximately one third of the water in the terrestrial water cycle moves through rivers.

Broadly speaking, an *estuary* is a body of water at the interface between the ocean (sea water) and inland (fresh water). Typical characteristics of estuaries include time-varying flow, temperature, and salinity, as the fresh water flowing down from river systems interacts with the tidally-driven sea water. Many formal definitions of estuaries have been proposed, mainly to deal with corner cases such as the absence of tidal action or intermittent connec-

tivity. The definition used in this work is:

> A semi-enclosed coastal body of water, which has a free connection with the open sea, and within which sea water is measurably diluted with fresh water derived from land drainage. [95]

Estuaries have unique features that have made them particularly important over the course of human development. Incoming water from the land carries large quantities of nutrients through the estuary; however, the range of salinity and temperature within the estuary make it a challenging environment. As a result, the species that live in the estuary are less diverse, and usually specialized to tolerate the range of conditions there, but populations are large due to the high nutrient content [37]. This specialized but high biological productivity can be a plentiful food source [132]. Estuaries are useful as transportation routes as well. As the connection between rivers and the sea, estuaries are the gateway between river travel and coastal travel. They have been the focal point of human development for millennia; approximately 60% of the world's population now lives along estuaries and the coast [68].

The spatial extent of estuarial systems, and the interrelationships of the water movement over large distances and periods of time, mean that hydrodynamic systems of this type are best studied as *distributed parameter systems*. Modeling distributed parameter systems with finite-dimensional state space models is non-trivial; there is a significant risk of losing important phenomena if the model is not properly constructed. *Partial differential equations* (PDEs) are one class of model that is commonly used to study the phenomena at work in distributed parameter systems. The Shallow Water Equations are a PDE system commonly used for estuarial hydrodynamics; these equations will be reviewed and discussed in Chapter 4.

1.1.2 A Local View: the Sacramento-San Joaquin Delta

The Sacramento-San Joaquin Delta is an estuarial region in northern California where the Sacramento River and San Joaquin River join and flow into the San Francisco Bay. The region has a tumultuous history of development and conflicting needs for its freshwater resources. Major modification of the landscape began in 1850, when the Swamps and Overflowed Lands Act encouraged small farmers to develop the region by granting ownership to homesteaders who developed it for agriculture [133]. Over the next 80 years, the region was transformed from marshy wetlands to a collection of drained and levee-protected islands. Soil loss caused the islands to sink, and confining the natural flow of water to a network of narrow channels caused the water level to rise; the water level is now 3 m – 5 m above the soil level in most of the Delta. As a result, the levees are highly stressed, and reversion to the pre-development marshland status is impossible; the infrastructure of the Sacramento-San Joaquin Delta is in an unsustainable position [70].

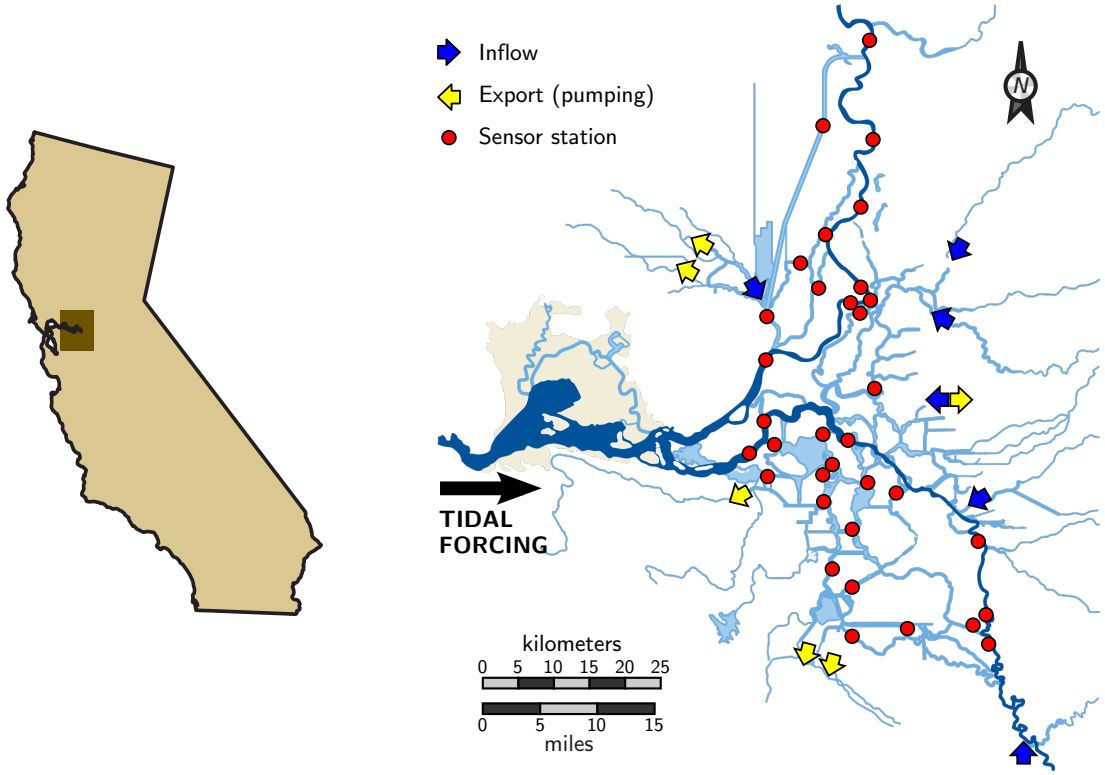

Figure 1.2: The Sacramento-San Joaquin Delta, showing sensor stations and incoming and outgoing flows. Adapted from [70].

In the 1930s, the Central Valley Project, a federally funded public works project, was founded with the goal of transferring water from the Sacramento Rivershed to the San Joaquin Valley (the southern half of California's Central Valley). Reservoirs in the north, and aqueducts, canals, and pumping stations in the south, allowed the more abundant freshwater resources of the Sacramento watershed to be brought to the relatively dry San Joaquin region. The state-funded State Water Project started work in the 1960s to expand the export of water to the communities of southern California. The California Aqueduct, whose construction started in 1963, brings water from the south of the Delta near Tracy to Los Angeles County, Santa Barbara County, and San Bernadino County. More than two thirds of Californians receive some part of their drinking water from the State Water Project [20].

Both of these major public water projects use the Sacramento-San Joaquin Delta as the transfer point for moving the water from the Sacramento watershed to the various southern destinations. The Delta is now a critical part of California's water infrastructure. Figure 1.3 shows the counties of California containing major population centers dependent on the supply

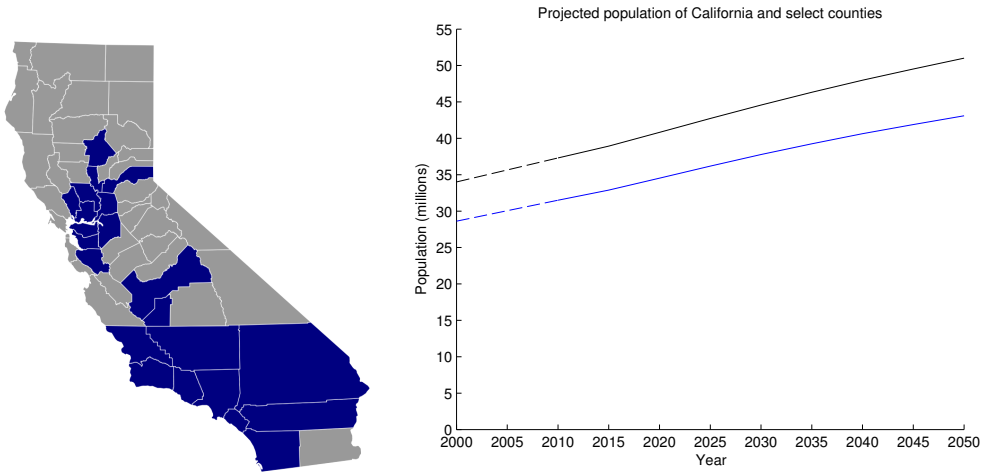

Figure 1.3: Left: counties with significant population centers dependent on the Central Valley Project or State Water Project. Right: Population projections for California and the Delta-dependent counties. Sources: [20, 106]

.

of water moving through the Delta, and the projected growth in the population of those counties over the next 40 years.

The Sacramento-San Joaquin Delta poses enormous challenges to managers, policy makers, and engineers. One goal of this work is to present a new tool for measuring and understanding the hydrodynamic processes at work in the Delta, and similar environments, in the hope that better information can help the decision makers who will determine the Delta's future.

1.2 Existing Methods for Sensing in Water Systems

To put this work in context, we present an overview and taxonomy of the types of measurement techniques used to gather information in open bodies of water. Figure 1.4 shows the hierarchy we will use to classify various methods.

In situ sensing refers to sensing techniques where a device is in direct contact with the environmental phenomena it measures. In contrast, *remote* sensing refers to techniques like satellite imagery, in which measurements are taken from afar. Remote sensing techniques for inland freshwater include aerial surveys and satellite imagery. They can be further classified by the information-carrying medium that links the remote observer to the phenomena of interest. Electromagnetic (E/M) radiation (visible light, infrared, radar) is the most common medium. Examples include the Aqua satellite [91], which observes infrared and visible E/M emission as part of the Earth Observing System, and TOPEX/Poseidon [47], which maps the ocean surface with active radar. Other techniques are also possible; one example is the

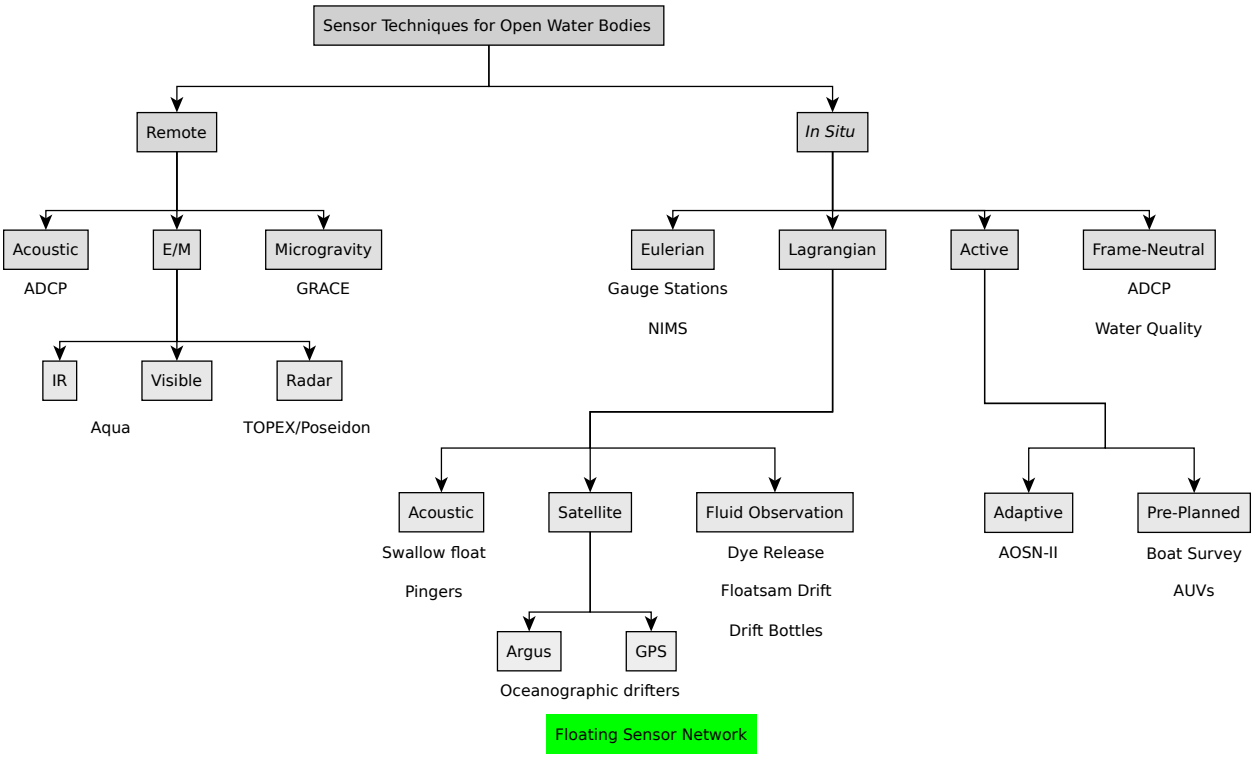

Figure 1.4: Taxonomy of sensor techniques for natural, open water bodies.

GRACE [116] microgravity observing satellite system, which measures Earth's gravity by observing orbital perturbations in a pair of satellites separated by 220 km in a polar orbit.

In situ sensing in fluid environments is classified into *Eulerian* and *Lagrangian* techniques, using the terminology for the different reference frames in hydrodynamics. Eulerian sensors are fixed to the external reference frame (e.g. the river bank) and take measurements from the water as it moves by. Lagrangian sensors float freely in the fluid itself, and gather measurements about the water as it moves through the environment. Some sensing techniques fall into neither category; for instance, a boat crew surveying a river by navigating upstream with a current profiler is in neither the Eulerian nor the Lagrangian reference frame. These techniques will be referred to as *active* techniques, because they usually involve propulsion through the water. Finally, many *in situ* sensors simply measure characteristics of the water they contact, with no specific requirement on the reference frame; these are *frame-neutral* techniques.

The classic Eulerian sensor technology for inland hydrodynamics is a *gauge station*. The water surface elevation, or *stage*, is measured at regular intervals. One fairly common method of taking such measurements is to use a *stilling well*, as shown in Figure 1.5. The velocity of the water or the discharge of the river may also be measured at the gauge station, either by direct observation, or by making inferences from the stage itself. More complex Eulerian

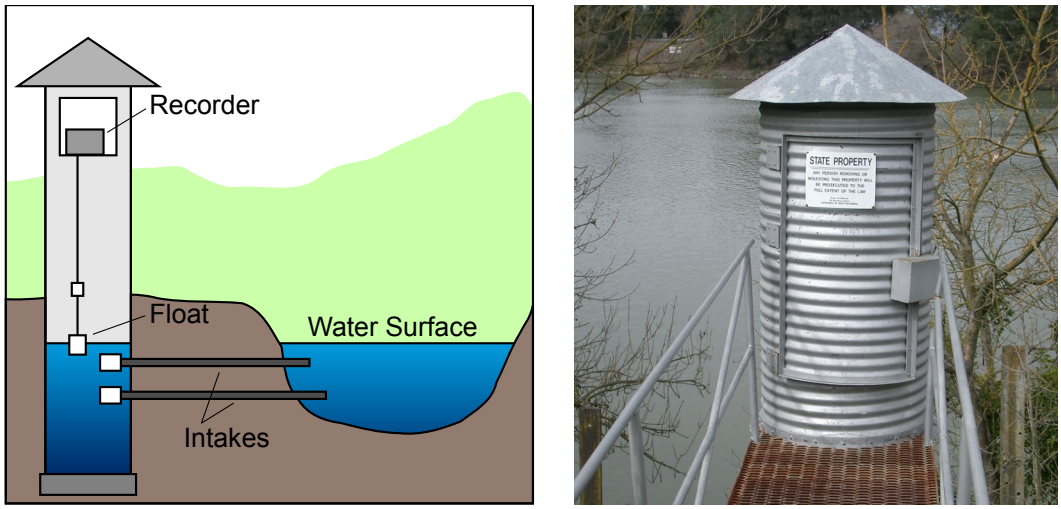

Figure 1.5: Stilling well for gauge station. Diagram adapted from [81]; photo of DWR gauge station in Walnut Grove, CA.

sensor systems exist; one example is the NIMS cable profiling robot [52], which uses a cable stretched across a river channel to move a sensor device in a scanning pattern over the complete two-dimensional river cross-section in order to build a profile of velocity, temperature, and other physical variables of interest.

In oceanography, a sensor device that operates in the Lagrangian mode by floating untethered with the current is called a *drifter*. Although studies of *flotsam drift* (drawing inferences about currents from the observed movement of accidentally dropped material) can be found in antiquity, the first deliberate drifter study seems to be the work of Georges Aimé *circa* 1845 [22]. His first drifters were *drift bottles*: sealed bottles containing a message asking the eventual recipient to report the date and location found. Drift bottle studies became a widely used technique in European oceanography around the beginning of the 20$^{\text{th}}$ century [21].

One of the most important kinds of data gathered by Lagrangian sensing is the time series of the sensor's position. Drifter design has always been constrained by the positioning and communications technologies available; drifters can be classified by their positioning technology. Underwater acoustic techniques and satellite positioning are the two most common technologies.

The first drifter that could actively communicate its position back to researchers was the "Swallow float", invented by John Swallow in 1955 [114]. It was a neutrally buoyant float that would drift approximately 1000 m underwater while transmitting acoustic pulses that would be received by researchers' hydrophones. Development of drifters with acoustic communication capabilities continued in the 1960s and 1970s [50]. In 1978, the introduction of the Argos satellite service [26] gave oceanographic researchers a global location and data

uplink system, which lead to the development of oceanographic drifters that could communicate their position and sensor data during the mission. Examples of oceanographic drifters that leverage the Argos system include the Davis, aka Coastal Dynamics Experiment drifter [31], the Ministar, aka World Ocean Climate Experiment drifter [77], and the Low Cost Tropical Drifter [14], each developed in the mid-1980s. Power, cost, and size constraints meant that Argos-based drifters were better suited to oceanography than inland environments like rivers and estuaries. In the 1990s, GPS technology began to replace Argos for positioning [130, 131]. In many cases, GPS drifters used the same form factors as the Argos drifters, maintaining the focus on oceanographic as opposed to inland applications.

Underwater acoustic positioning continued to develop, even after the introduction of satellite positioning. Acoustic pingers have been miniaturized significantly, to the point where they can be implanted in juvenile fish to track migration [122]. Pingers on fish, of course, are not a Lagrangian sensor, but these miniaturized pingers can also be used to locate Lagrangian drifters, in oceanographic and inland contexts. One disadvantage of such techniques is the need to install and calibrate hydroacoustic receivers to listen for the pingers' signal.

Another kind of Lagrangian sensing does not rely on floating devices, but rather on induced or known properties of the fluid. In dye tracer studies, a fluorescent dye is released at a specific point in the water body, and fluoroscopic measurements are taken at different points downstream to measure the arrival and concentration of the tracer. Similar studies involve natural properties of the water; for example, when two streams come together, and the water in each stream has different characteristics (temperature, salinity, turbidity, etc.), water coming from either source can be disambiguated downstream by measuring that characteristic. Mixing and diffusion are confounding processes in all studies of this nature. Although the direct sensing technique is Eulerian (the sensor is at a fixed location), the information being gathered is really Lagrangian in nature; there is an inferred particle trajectory between the upstream source and the sensor location. These techniques can be classified as Lagrangian, with "fluid characteristic observation" as the positioning technique.

An important sensor for inland hydrodynamic studies is the *acoustic Doppler current profiler* (ADCP). This device emits acoustic pulses in a confined beam, and captures the returning acoustic energy as the pulse bounces off air bubbles and particles in the fluid. The Doppler shift of the returning signal can be used to infer the velocity of the water along the beam path. This is a remote sensing technique, using acoustics as the information-carrying medium, but the relatively short range and need to immerse the sensor head in the water column mean that the ADCP is best classified as a frame-neutral *in situ* technique.

The most common active sensing technique is a boat-driven survey, as mentioned above. Significant research attention has been devoted recently to autonomous unmanned sensing vehicles, particularly in the oceanographic context. Examples include the AMOUR project at MIT [32], the NEPTUS *Autonomous Underwater Vehicle* (AUV) at LSTS in Portugal [112], and the Slocum underwater gliders [127, 13]. The most important distiguishing characteristic between different active techniques is whether the data being gathered by the sensor

influences the path taken by the sensor through the water. This division will be described as *adaptive* versus *pre-planned*. Most AUV missions are pre-planned; one exception is the Second Autonomous Ocean Sampling (AOSN-II) experiment in Monterey Bay [97], in which teams of Slocum gliders estimated the temperature gradient and adjusted their trajectory autonomously in order to map an upswelling front.

1.3 Innovative Approach: Actuated Mobile Sensing

The *Floating Sensor Network* (FSN) project at UC Berkeley has developed a new approach to Lagrangian drifter-based sensing in rivers and estuaries. We have developed small, inexpensive, highly capable drifting sensors, including a new class of sensor: an actuated Lagrangian sensor. A Lagrangian drifter with a propulsion capability that is used only intermittently is a novel approach. This work focuses on the design of these new sensors, including the design of the algorithms needed to process their data and control their movement.

1.3.1 Challenges for Lagrangian sensing in river environments

Classic Lagrangian drifters are extensively used in oceanographic contexts, but are fairly rare in river and estuarial environments. One reason is the history of satellite-based positioning and communication capabilities. Positioning using the Argos satellite system is perfectly adequate in the oceanographic context, but river studies require accuracy that was not available until the advent of the *Global Positioning System* (GPS). Even with the availability of GPS, however, river Lagrangian studies are not widely used. River environments present extensive obstacles, including underwater vegetation, channel banks, and man-made structures, that are essentially absent from the ocean environment. These obstacles are characteristic of an *unstructured* environment: they present complicated challenges to movement, and there is usually a high degree of uncertainty about their location or extent. Applying drifter technology to inland environments therefore requires addressing the obstacle challenge directly. Typically, passive drifters are supervised by personnel in boats, who retrieve the drifters when they get snagged on obstacles. Lagrangian studies in rivers and estuaries therefore have a comparitively high operating cost, due to the need for supervision. Our approach, which is to add limited propulsion and autonomous obstacle avoidance to floating drifters, is an approach that sets the FSN project apart from other drifter research.

As mentioned above, there are many research projects investigating the capabilities of AUVs and other active water vehicles. However, these vehicles may not meet the requirements of a Lagrangian sensor when unactuated. In Chapter 3, we explore these requirements. The active drifters of the Floating Sensor Network fleet have a very different hull shape, which makes this work distinct from most AUV research.

1.3.2 Actuation conflicts with Lagrangian Behavior

A floating Lagrangian sensor gathers useful information by moving at the same speed as the water in which it is immersed. Applying actuation to move in a different direction, by definition, breaks this Lagrangian assumption. Although it could be possible to extract useful Lagrangian information from an actively moving device, to do so would require a very precise system identification, to be able to infer the exact deviation of the vehicle trajectory from the "natural" water flow. We have not developed this possibility; instead, we simply assume that the information gathered by the active device is invalid during those times when the propulsion is applied. Naturally, this leads to a control objective of using the propulsion as little as possible.

1.3.3 Mobile Phones and Inland Lagrangian Sensing

As mentioned in Section 1.2, Lagrangian drifters can be classified by their positioning and communication techniques. Each new positioning or communication technology has driven a new generation of drifters. Inland environmental studies are being transformed by the availability of data communication via the mobile phone network. Although civilian cellphone technology is not new, it is only fairly recently that mobile phone coverage was comprehensive enough to ensure high availability. In addition to providing communication for monitoring and sensor telemetry, the growth in the smartphone industry has enabled a new class of environmental sensor: sensors built with an actual smartphone as a key component for computation and positioning, not just communications. In Chapter 3 we will discuss the design of an inland drifter using an Android smartphone as the principal positioning, computation, and communication unit.

1.4 Scenarios for Mobile Sensors

This section explores the possible benefits of Lagrangian sensing in river environments with three illustrative scenarios. Two of these examples feature actual test deployments of FSN equipment.

1.4.1 Contaminant Propagation in Large-Scale Estuarial Systems

Jones Tract is one of the Delta "islands", that is, a drained and agriculturally developed body of land surrounded by levees. On June 3, 2004, one of the levees surrounding Jones Tract failed. Water from the Middle River poured into Jones Tract, creating a lake with an approximate area of 48 km^2 and average depth of approximately 4 m [115]. Figure 1.6 shows satellite imagery of the new body of water.

In order to restore Jones Tract to agricultural use, the water in Jones Tract was pumped back into the Middle River after the levee repairs were completed. This pump-out process

Figure 1.6: Satellite imagery of Jones Tract before and after breach.

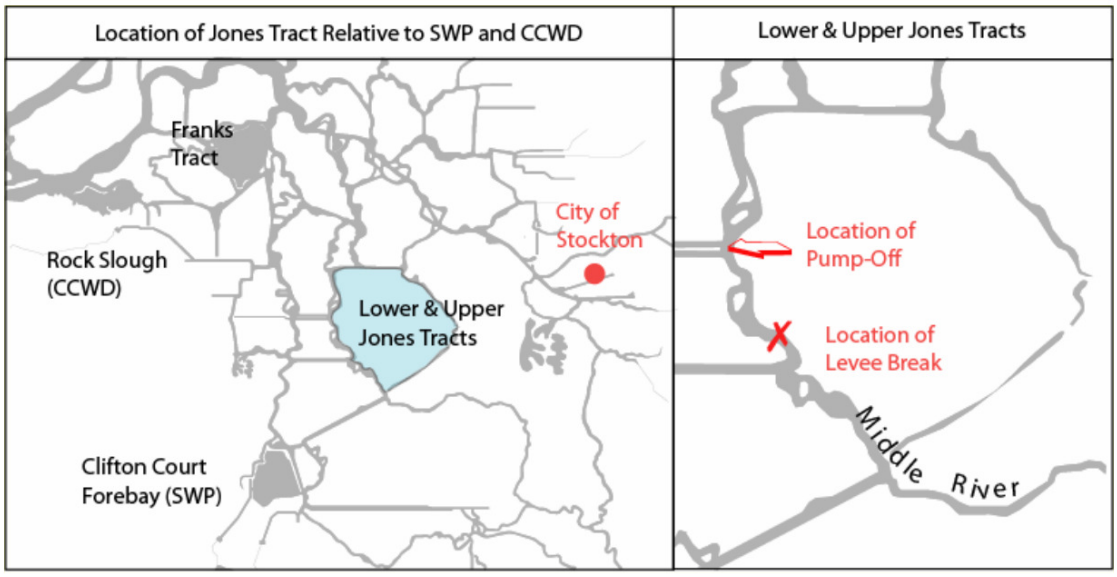

Figure 1.7: Map of the Jones Tract region and Clifton Court export point. Reproduced from [72].

took approximately six months. The water pumped from Jones Tract contained many constituents that would reduce its suitability as drinking water [115]. As shown in Figure 1.7, the Jones Tract and the Middle River are close to the Clifton Court Forebay, which is the export point for the State Water Project and the Central Valley Project. The pump-out operation potentially threatened the quality of the exported water.

In order to assess the potential harm to water quality at Clifton Court, the managers at the California Department of Water Resources would need to understand the outcome of contaminated water entering the Middle River at the pump-out point. Lagrangian sensors provide a natural way to investigate these processes. Sensors released at the proposed pump-out site could trace the path of water as it flows out to the San Francisco Bay, and

Figure 1.8: HERU facility: Hydraulic test area, empty, with test levee in background. This facility was used to test the Floating Sensor Network drifters' survivability in levee breach conditions.

evaluate whether or not this water reaches Clifton Court. Mixing processes would be a confounding factor. Nevertheless, Lagrangian tracing would be a useful tool for investigating water contamination questions like this example.

Disaster response is an important application for water contamination investigations, but it is not the only one. Under the Clean Water Act of 1972, most dredging activities in waters under U.S. jurisdiction require a permit from the U.S. Army Corps of Engineers [27]. Assessing the potential environmental impact of disturbed sediment or other released material is part of the permitting process. Lagrangian sensing could be a useful tool for determining the downstream impact of these types of activities.

1.4.2 Network Remapping in Natural Disasters

Natural disasters in estuarial regions could lead to levee failures at many, possibly unknown, locations. In the case of the Sacramento-San Joaquin Delta, there are over 1500 km of levee-bounded channel; a major seismic event could have severe repercussions, and the extent of the damage could be difficult to ascertain. Other natural disaster scenarios, like a hurricane striking coastal infrastructure, include ongoing hazardous conditions that would make it dangerous or impossible to send personnel out to assess levee structures.

In these kinds of scenarios, a fleet of Lagrangian sensors could be used to map out the levee system, including any failures, without risking human life. By tracing the path of water, and communicating position information back to responders, breaks in levees could be detected.

Figure 1.9: FSN drifter about to pass through the levee breach at the HERU facility.

This capability was tested in 2009 when the FSN team was invited to participate in the Rapid Repair of Levee Breaches Demonstration at the Department of Agriculture's Hydrologic Engineering Research Unit (HERU) in Stillwater, Oklahoma. The test was operated by the Department of Homeland Security and the US Army Corps of Engineers. A test levee was built at the HERU facility, as seen in Figure 1.8. The region behind the levee was filled with water and the levee was breached, causing a progressive levee failure and a hydraulic jump. The sensors were deployed upstream and allowed to pass through the breach, as shown in Figure 1.9. The sensor survived the transition and transmitted GPS information from the other side, illustrating the potential application for natural disaster response.

1.4.3 Hydrodynamics Studies

Lagrangian sensors provide valuable information when investigating the hydrodynamics of tidally forced estuarial regions. Many estuarial regions feature *tidal reversal*: at the interface between a freshwater stream and a larger body of water, the water flows out of the stream during low tide, but at high tide the flow reverses and goes "upstream". One important question for studying such interfaces is whether the water that is pushed back up the stream at high tide is the *same* water that flowed out of the stream in the preceding low tide, or whether mixing processes in the tidal body have diluted the "original" water. This

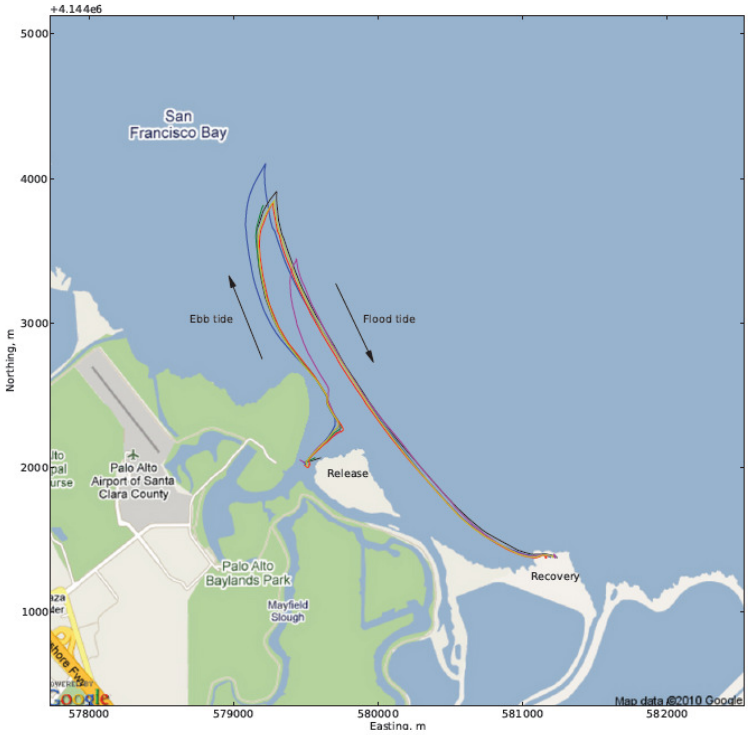

Figure 1.10: GPS traces from floating sensors during 24 hr San Francisco Bay experiment.

kind of question is difficult to answer with Eulerian sensors, but Lagrangian sensors are a natural way to "tag" the outgoing water to trace its later disposition.

In May 2010, the FSN team assisted with an experiment with the California Bay-Delta Authority to trace the outgoing water from a small slough into the southern San Francisco Bay. The drifters were deployed near the mouth of the slough during an outgoing tide, and recovered 24 hr later. Their GPS traces support the conclusion that significant mixing processes occur, meaning that the water pushed up the slough on high tide is not the same as the water that left the slough on the last cycle.

1.5 Project Overview

1.5.1 Problem Statement

This dissertation focuses on floating sensors, particularly *actuated* floating sensors, for Lagrangian sensing in estuarial environments. In particular, it addresses the following questions:

- Are sensing systems of this kind feasible?

- How should these systems be designed?
- What aspects or subsystems are well understood, and which subsystems require innovation or pioneering research? Where are the limitations of our current understanding for these issues?

1.5.2 Development of the Floating Sensor Network

The Floating Sensor Network project developed several successive generations of Lagrangian sensor device. Although the sensor unit itself is not the entirety of the Lagrangian sensor system, as will be discussed in Chapter 2, it is such a major component of the overall system that it serves as a useful milestone for chronicling the overall development of the project. Table 1.1 lists the four major sensor types developed by the FSN team. Generation 1 and Generation 2 were purely passive models, with no actuation or propulsion (although prototypes based on Generation 2 had actuation, they were never built in volume). Generation 3 incorporated a twin propeller system for moving on the surface of the water. As this dissertation investigates *actuated* Lagrangian sensing, much of this work is directly relevant only to this generation of the device. Finally, the Android drifter was developed with a radically different approach to the internal instrumentation (using an Android smartphone instead of custom integrated electronics for all functions). The Android drifter, like Generations 1 and 2, is a passive drifter. However, its low cost and ease of assembly allowed us to reproduce it in significant numbers, opening up the possibility of Lagrangian sensing on a previously unexplored scale.

1.5.3 Outline of Work

Chapter 2 introduces a system decomposition of actuated Lagrangian sensing in estuarial environments, and defines the subsystems that are considered in subsequent chapters. Chapter 2 also discusses the question of how formal design methodologies can be adapted to facilitate design documentation for research projects like the Floating Sensor Network.

Chapter 3 considers the design of the four generations of floating Lagrangian sensor shown in Table 1.1. The functionality objectives and design constraints for this design problem are analyzed and discussed in order to illustrate some of the issues in the design and development of this class of sensor system.

Chapter 4 reviews the basic models for the phenomena of interest in freshwater systems: the Shallow Water Partial Differential Equations. This chapter discusses Quadratic Programming for drifter data on two-dimensional shallow water systems, a tractable variational approach to data assimilation that was developed for the Floating Sensor Network.

Chapters 5 and 6 cover two approaches to the control problem for fleets of actuated sensor vehicles in environmental flow fields, like the ones found in river environments. Chapter 5 introduces a single-vehicle control scheme to remain safe in the presence of obstacles, using

Name	Generation 1 [120]	Generation 2 [119]	Generation 3 [82]	Android [11]
Image				
Fleet size	10	10	40	70
Dimensions	11.5 cm diameter, 39 cm tall	11.5 cm diameter, 39 cm tall	13 cm diameter, 47 cm tall, 24 cm span across motor pods	13 cm diameter, 29 cm tall
Cost (parts)	$1 000	$1 200	$2 500	$300
Assembly time	60 h	60 h	10 h	2 h
Mission time	24 h	24 h	24 h	48 h
Computation	Gumstix Connex: 400 MHz, 64 MB RAM, Linux 2.6	Gumstix Verdex: 400 MHz, 64 MB RAM, Linux 2.6	Gumstix Overo: 720 MHz, 512 MB RAM, Linux 2.6	Motorola Defy: 800 MHz, 512 MB, Android 2.3
Communication	GSM	GSM and 802.15.4	GSM and 802.15.4	GSM
Propulsion	None	None	Twin parallel propellers	None

Table 1.1: Four generations of Floating Sensor Network devices, and corresponding technical specifications.

the Hamilton-Jacobi-Bellman-Issacs Partial Differential Equation to solve a differential game. Chapter 6 describes an algorithm for fleet control based on an adaptation of the Voronoi partition to flow environments, namely the Zermelo-Voronoi partition; it also describes a novel approximation technique to allow the computation of these partitions efficiently in an embedded systems context.

Chapter 7 considers a design for a decentralized communication scheduling scheme for dynamic connectivity on mobile robots. It is designed to fill the role of scheduling algorithm at the Media Access Control layer of the IEEE 802.15.4 low-power mesh networking standard in Time Synchronized Channel Hopping mode.

Chapter 8 documents a pioneering field experiment conducted by the Floating Sensor Network team in May 2012, in which 100 floating Lagrangian sensors were deployed in the Sacramento River in order to demonstrate and validate the actuated Lagrangian sensor concept for inland river and estuarial studies.

Chapter 9 concludes this dissertation with a summary of the progress made on the research agenda described in this chapter, as well as an overview of future directions for actuated Lagrangian sensing and its applications.

1.5.4 Contributions

While Lagrangian sensing is a well-accepted tool for oceanography, the challenges posed by inland environments described in Section 1.3.1 have prevented widespread use in river and estuarial environments. Chapters 2 and 3 describe new designs for actuated and passive Lagrangian sensors that can surmount these challenges.

Chapters 4, 5, 6, and 7 focus on specific advances in data processing, vehicle control, and communications scheduling necessary for the implementation of these new sensors. Chapter 4 documents a new, tractable formulation of the data assimilation problem for mobile sensing in rivers. Chapter 5 describes the development of a PDE approach to the optimal control of an actuated sensor in a river; the approach it describes is a new contribution in the way the framework handles the flow field of the river while respecting the computational limits of the sensor platform. Chapter 6 presents a new variation of the Voronoi partition, adapted to affine flow fields, and a new technique for approximating these partitions suitable for platforms with limited computational power. Chapter 7 describes a new algorithm for scheduling communications over short-range mesh networking radios in a decentralized setting, which is an important feature for mobile sensors distributed in an unstructured environment.

Chapter 8 presents a field experiment where 100 mobile sensors were deployed in a river environment to gather flow data. The scope and scale of this experiment are a significant advance in actuated mobile sensing in inland hydrodynamic environments. The results of this experiment demonstrate the feasibility of the mobile sensing concept, and provide insight into the future work necessary for the development of this technology.

Chapter 2

Designing Mobile Floating Sensor Systems

A system for making estimates of the state of a water system encompasses more than just the Lagrangian sensors floating in the water. From 2007 to 2012, the Floating Sensor Network project developed water estimation systems for estuarial studies. In this chapter, we take a broad view of the entire water sensing system and its architecture, and discuss the design methodologies that will be used to present the implementations developed by the Floating Sensor Network project. A design presentation framework is defined; it is important to note that this framework is a *post hoc* method for documentation and analysis. In Chapter 3, we will examine the design of successive generations of the drifter units themselves, involving issues of mechanical, electrical, and software design.

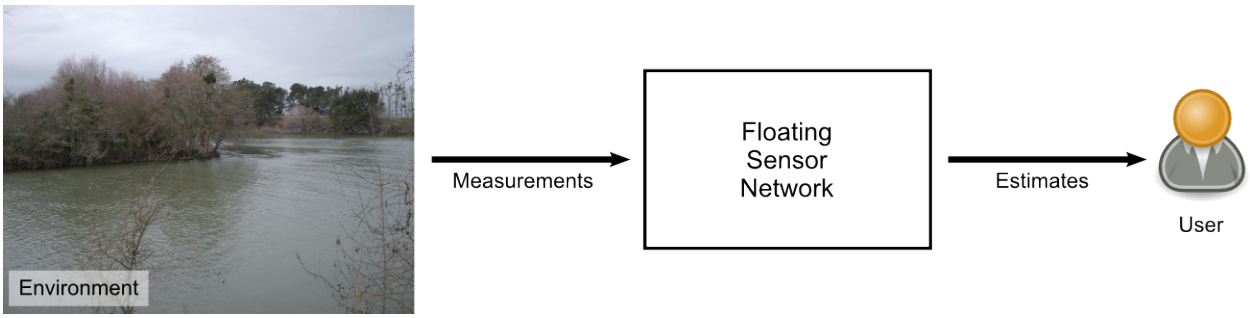

Figure 2.1: Black box view of a Lagrangian water estimation system.

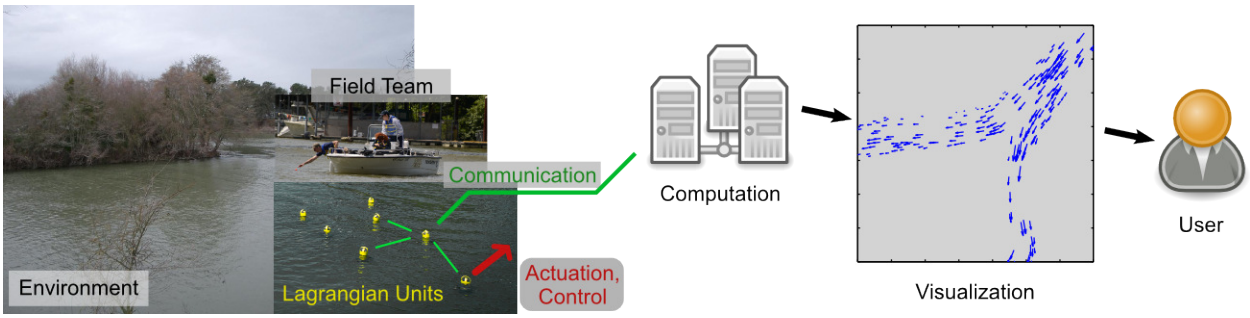

Figure 2.2: Overview of the elements of a Lagrangian water estimation system.

2.1 A Transformation System View of a Lagrangian Water State Estimation System

Hubka and Eder [56] define a *transformation system* as a collection of subsystems that work to change the state of operands, which may be materials, energy, or information. Using this perspective, we will consider the Floating Sensor Network system as an information transformation system, gathering *measurements* (information in the form of sensor readings from the environment) and producing *estimates* (usable quantitative values for the state of the environment). Figure 2.1 gives a black-box view of the Floating Sensor Network system, showing the input and output relationship as information flows from the environment, through the Floating Sensor Network system, to the user.

When we consider a state estimation system that uses discrete Lagrangian devices as the sensor element, many of the subsystems necessary for the estimation function can be defined. Figure 2.2 gives an overview of the elements of a Lagrangian water estimation system. The input and output side of the information transformation, namely the environment and the user, are included as "endpoint elements".

2.1.1 Endpoint Elements

Environment

The environment in which the Lagrangian sensors are immersed, namely the river or estuarial system in which the sensors drift, is not typically under design control; in some senses, it is one of the givens of the problem. Nevertheless, some parameters of the system are within the designer's scope: the specifications of the kinds of water environments where the system should be applied. This could involve classifying the domain by type — oceans, bays, lakes, rivers, streams, etc. — or by *sea state* parameters, such as wave height distributions and wave periods, that represent the loads placed on structures in the water. The spatial and temporal variability in the velocity of the water affects the design of an "ideal" Lagrangian particle, and so this should be specified as well. The limits of the experimental domain in

space and time can also be specified.

User

Similarly, the nature of the user of the system may not be a design choice so much as a given of the problem. Including the user in the system description is a way of acknowledging that the design of the system should proceed from the user's needs and requirements for the estimates that the system produces. Section 1.4 has a number of use cases and example applications in which a system of this type could be applied.

2.1.2 Intermediate Elements

Lagrangian Units

This refers to the physical units themselves that are immersed in the water. The design problem for the Lagrangian units must cover their properties in the fluid, the way they collect measurements (in particular, the positioning method used to determine their location and speed), and many other implementation issues. If the Lagrangian devices have propulsion, then the capabilities of individual devices (turn rate, maximum speed relative to the water, power consumption, etc) are part of this element. Chapter 3 is dedicated to exploring these issues, presenting our solutions using the framework described in Section 2.4.

Actuation and Control

If the Lagrangian units have actuation, then the policy and control schemes used to effect changes in position are part of the design problem. As noted in Section 1.3.2, it is quite likely that active movement of the unit means that the Lagrangian assumption is broken and that the position of the vehicle can no longer be used to infer the velocity of the water. (Various system identification schemes to extract the water movement from the active movement could be developed, or other sensor techniques that allow direct measurement of the "ground speed" of the water, or a modification to the assimilation procedure that allows limited inference or low-confidence estimates based on the corrupted measurements; but these possibilities are not explored in the current work).

Field Team

The deployment and recovery of the Lagrangian units will most likely involve human effort. The actions required by human operators, the skills they need to employ, and the equipment they need to have are an important part of the system design. Deployment and recovery by a boat team is a common requirement. In the case of a river environment, human intervention may be required if drifters get caught on the shore or on other obstacles in the waterway. The need for human intervention will drive up the operating costs of the

system. One possible design variation is to develop disposable Lagrangian units that are not recovered; operating costs can be further reduced by deploying from the shore, a dock, or a bridge instead of from a boat.

Communication

The system user and the Lagrangian sensor units are not physically co-located, so some method of communication must be used to move information. This design problem is bound up with the question of how the information transformation takes place (see the next section, "Computation", for more details). The most common technique is to use some kind of RF wireless communication to deliver the information in real time. Another possibility is to simply store measurements locally on the Lagrangian unit, then extract the measurements in the lab and perform computations in a post-processing context. Communications, however, are also part of the field team's interaction with the devices. If the Lagrangian units are to be recovered and reused, it may be important for the field team to have real-time location and health information about the position and state of the devices.

Computation

On its own, the raw velocity information from a sparse set of locations in a water system is not very useful. *Data assimilation* is the process of taking this sparse information and making the best possible estimate of the state of the system over the complete space/time domain. This is the fundamental transformation process at the core of the system. Chapter 4 cover some of aspects of this element: the model of the water system used for the assimilation, and the algorithm used to perform the data assimilation. Another aspect is the location and nature of the computational units. This relates to the communication element. The measurements gathered by the Lagrangian units could be collected into a central server and processed together, or the CPUs on board the Lagrangian units themselves could perform some or all of the processing (given a suitably decentralized algorithm).

Visualization

The interaction point with the user is where the estimate generated by the data assimilation algorithm is presented for consumption. It is of particular importance in the case of a real-time system, where estimates are made based on new measurements, and delivering comprehensible information to the user quickly is necessary. This design problem is definitely in the discipline of Human-Computer Interaction, and is not significantly developed in this work. Figure 2.3 shows a screenshot of *Dynamic Information Visualization Application* (DIVA), a software application developed for the Mobile Millennium traffic modelling project [8], and adapted to display water velocity estimates and drifter locations in real time.

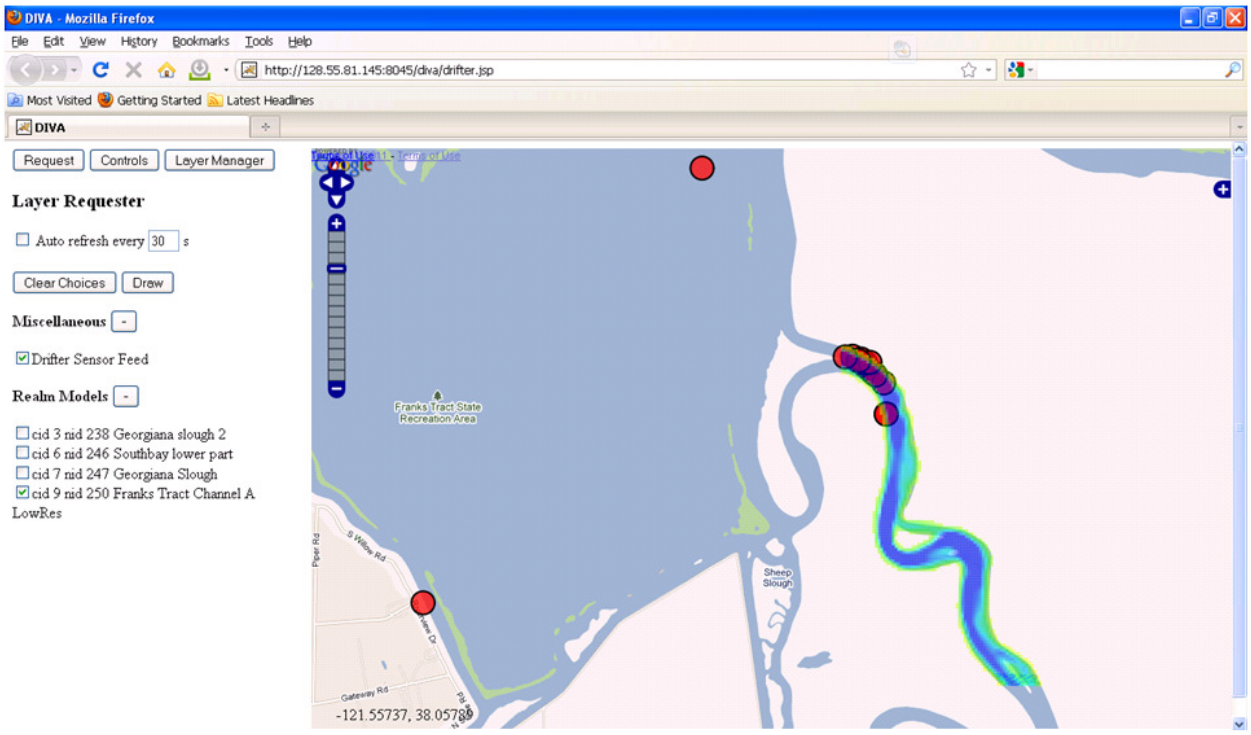

Figure 2.3: DIVA software visualizing drifter positions and water velocity estimate.

2.2 Design methodologies for documentation

Academics have long studied the *process* of design, as practiced within engineering as well as other professions such as architecture and the applied arts. Some of these studies are *descriptive*, that is, they seek to capture and document the cognitive processes used by working engineers to develop designs. Other studies are *prescriptive*, in that they set out protocols and methodologies for doing design the "right" way. In 1989, Finger and Dixon [44] wrote

> An implicit (and occasionally explicit) assumption of [prescriptive methodology] research is that if designers follow the prescribed process, better designs will result. The authors of this article are unaware of any research in which this assumption is tested scientifically.

The literature contains a staggering array of design methodologies, spanning almost all disciplines of engineering. A great many of these studies were developed within geographically distinct lineages of mechanical engineering in the 1970s and 1980s. Thus, there are independent bodies of design methodology literature from Germany, Japan, the United Kingdom, and the United States. More recently, researchers in software engineering have contributed a new collection of design methodologies specific to their discipline.

The tools of principled design methodologies can be adapted to more general purposes. In his introduction to a book on *Unified Modeling Language* (UML), a widely used diagramming tool in software engineering, Martin Fowler identifies two fundamental modes of working with UML diagrams [45]:

> **Forward engineering** draws a UML diagram before you write code, while **reverse engineering** builds a UML diagram from existing code in order to help understand it.

No formal design methodologies were used in the development of the Floating Sensor Network systems. The design processes used followed the more common approach: informal, *ad hoc* methods of describing the requirements, exploring possible solutions, selecting a set of solutions, and refining the specifics. In order to document and explain the design decisions made by the FSN team over the years, we looked to the literature on design methodologies to help organize the ideas we had and express them in a useful way. We are therefore in Fowler's reverse engineering mode: using the structures of principled design methodology not to come up with a new design, but to explain an existing design clearly and effectively.

2.3 The application of design methodologies

The taxonomy of sensor modalities for open water in Section 1.2 is inspired by Fritz Zwicky's *Morphological Analysis* [138, 85], in which a design problem is broken down into subproblems, every possible approach to the individual subproblems is recorded, and feasible combinations of subproblem solutions are sought in a more-or-less combinatorial fashion.

The primary approach we used was based on Nam Suh's *Axiomatic Design* [111], which is based on concepts of *functional requirement* (FR) and *design parameters* (DP). Functional requirements are the specific objectives that a design must satisfy in order to be considered successful; design parameters are the key variables that specify the physical entity that is designed to satisfy the FRs. The methodologies of Axiomatic Design treat design as a function that maps from FRs in "functional space" to DPs in "physical space"; in this framework, designs can be evaluated by the properties of this mapping, such as whether the choice of DP values maintains independence between the FRs. We do not take this normative approach, but the FR and DP concepts, and the idea of the "design matrix" that records the relationships between them, to be very useful.

Class	Questions about Class	Groups or Examples
Function or Effect	What does the TS do? What capability does the TS have?	Working function Auxiliary function Regulating Function
Functionally Determined Properties	What conditions are characteristic of the function?	Power, Speed, Load capacity Functional dimensions
Operational Properties	How suitable is the TS for the working process (operation)?	Reliability Life Energy consumption Maintainability
Ergonomic Properties	How is it to be operated, and what influence does the TS have on human beings?	Operator safety Way of operating Types of secondary outputs Required human attention
Aesthetic Properties	What influence does the TS have on human sensory feelings?	Form Colour Surface distribution
Distribution Properties	How suitable is the TS for transport, storage, packing?	Transportability Storage/packaging suitability Suitability for commissioning
Delivery and Planning Properties	When can the TS be delivered? Manufacturing quantity	Delivery capability Quantity production One-off production
Law Conformance Properties	Does the TS conform to laws, codes of practice, standards?	Patent clearance Legal
Manufacturing Properties	How suitable is the TS for manufacture?	Manufacturability Manufacturing quality
Design Properties	With what are the external properties?	Structure, Form, Shape Dimensions, Tolerances Materials
Economic Properties	How economic is the working and manufacturing process?	Operating costs Manufacturing costs Effectiveness Price Manufacturer
Liquidation Properties	How easy is the TS to liquidate?	Re-cycling Danger of wastes

Table 2.1: Properties of Technical Systems. Adapted from [56]. “TS” is an abbreviation for “Technical System”.

The separation between functional requirements and design parameters is not clean. Faced with an open-ended problem, such as "design a system for estimating the state of a distributed parameter open water system", the initial choice of approach creates new constraints and functional requirements for the design. For example, in Section 3.2 we identify "human portability" as an ergonomics-related functional requirement of a Lagrangian sensor system. This functional requirement does not apply to all approaches to the design problem. If we had chosen instead to build a remote imaging system, like a satellite or an airplane-mounted device, we would not have these ergonomic constraints. Managing this two-way relationship between design parameters and functional requirements is a deep and difficult problem. In this work, we cut this knot by simply taking our first set of design choices as a given, and building the functional requirements that descend from that choice. For that reason, the problem statement in Section 1.5.1 is written to include the choice of using Lagrangian, actuated, floating sensors as the sensing technique.

In the book "Theory of Technical Systems" [56], Vladimir Hubka and W. Ernst Eder categorize and classify different *properties of technical systems* for the purpose of evaluating different designs and solutions to technical problems. Table 2.1 is reproduced from their work. The classes of properties are not orthogonal; for example, "Delivery and Planning Properties", "Manufacturing Properties", and "Economic Properties" share common ground. The classes are also not hierarchically equal; eleven out of the twelve classes are "external properties", while "Design Properties" is an "internal property" that encapsulates all the design decisions made to fulfill the requirements set by the external properties. We can therefore see a rough match between Hubka and Eder's "external property" and Suh's "functional requirement". We use Hubka and Eder's external properties as the basis for a coverage search of the important functional requirements for the design.

2.4 Presentation of design evolution

In Chapter 3 we will examine the design of the Lagrangian Units, Actuation and Control, and Communication elements of the system decomposition presented in Figure 2.2. We will first use the categories in Table 2.1 to enumerate the important Functional Requirements for the design. The Design Parameters will be described, and the coupling matrix between the two sets presented. The coupling relationships between FR and DP will be described, first in terms of the commonalities between all the successive generations of drifter, and then focusing on the design parameters that changed from generation to generation.

Chapter 3

Four Generations of Drifter Design

Using the methodological ideas proposed in Chapter 2, we consider the functional requirements and design parameters pertaining to floating sensor units. The choices made in the four different generations of drifter are considered; first by looking at the commonalities between the design generations, then looking at the differences and improvements between each generation.

This chapter expands on ideas presented in [82], which presented the mechanical design of one generation of FSN drifter through the lens of formal design methodologies.

3.1 Terms of Reference and Limitations

In Chapter 2 we considered the Floating Sensor Network as an information transformation system, and decomposed it into subsystems. In this chapter we take up the question of how to design the Lagrangian Unit subsystem. Other subsystems will be partially examined as well, namely the communications subsystem and the actuation/control subsystem. Some design issues for the communications subsystem (namely, the decentralization of a communication scheduling scheme) will be investigated in Chapter 7. Two different control schemes for the drifter actuation will be covered in Chapters 5 and 6.

By accepting the system decomposition of Chapter 2, we are starting the design problem with a commitment to certain design choices: we are designing floating Lagrangian sensors, with a propulsion system, communicating using wireless technology to a backend server, and recording position with GPS modules.

3.2 Functional Requirements for Drifters

To develop the list of functional requirements, we used the "Properties of Technical Systems" hierarchy reproduced in Table 2.1. The functional requirements are the rows in the design matrix shown in Figure 3.1.

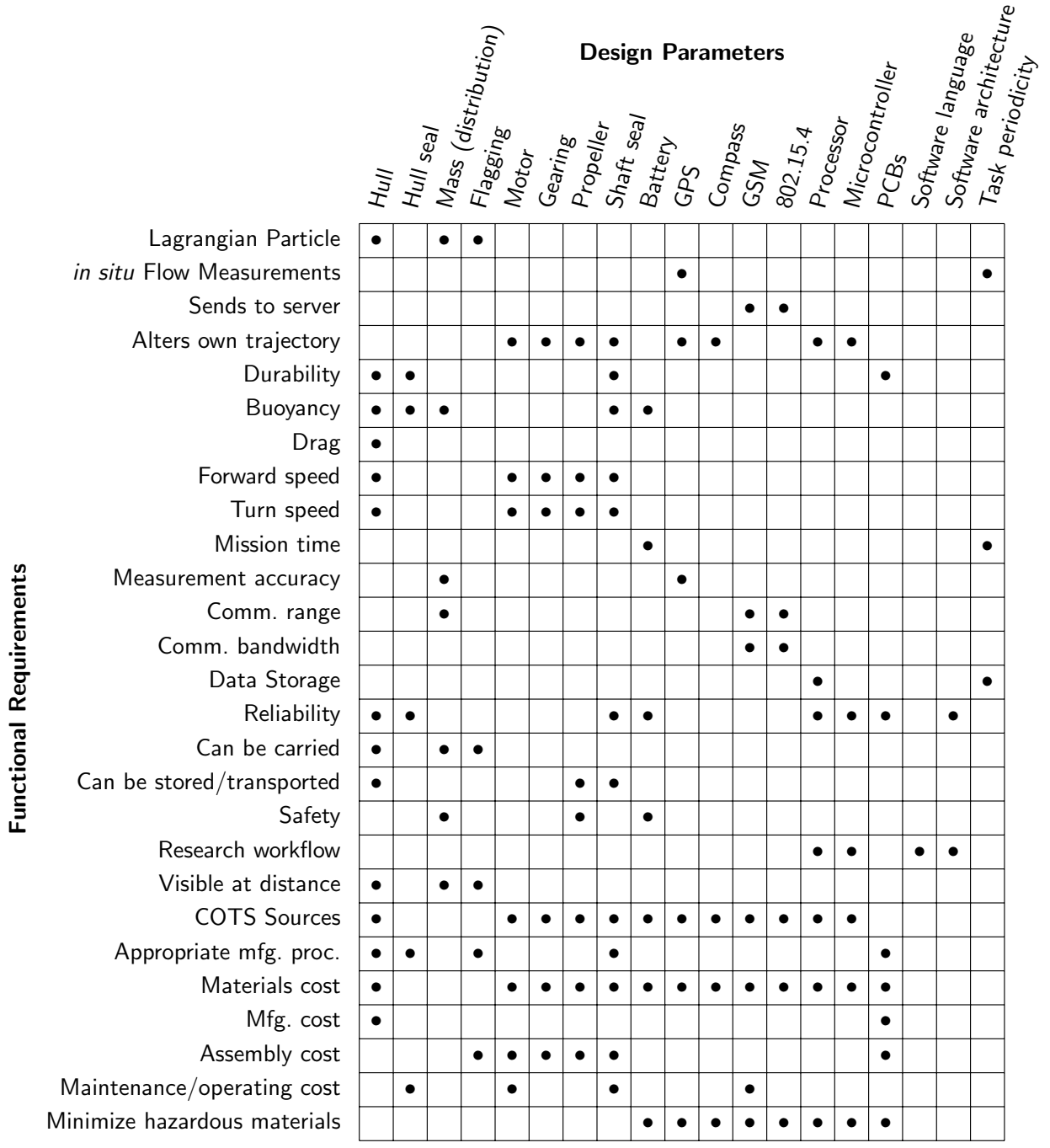

Figure 3.1: Design matrix showing connections between Functional Requirements and Design Parameters for floating sensor design.

3.2.1 Function or Effect Properties

Lagrangian particle movement in rivers

The first function of the floating sensor device is to accurately reproduce the movement of water through a river system. Essentially, this means that when actuation is not being applied, the device must come to rest relative to the stream of water in which it floats. Floating devices, which are exposed to the movement of the air above the water, must not be unduly influenced by any wind effects.

Provides timely *in situ* measurements of water flow

The other elements of the transformation system described in Section 2.1 need accurate measurements of the water motion that is being captured by the floating sensor, at a sufficient spatial and temporal resolution to enable an accurate overall estimate of the state of the hydrodynamic system. It is therefore necessary to take observations of the position (and velocity) of the device. As described in Section 1.2, there are many possible methods for localizing Lagrangian sensors in an outdoor environment; hydroacoustic pingers or the Argus satellite network are other possibilities. For this design, we assume that the decision to use a GPS module has already been made.

Measurements are sent to a central server

The position and velocity measurements must be communicated to the server where further computation takes place. Two wireless communications technologies were used over the course of this research: *Global System for Mobile Communications* (GSM) modules, which use the civilian mobile phone infrastructure to send data to the Internet; and 802.15.4 radios, which enable short-range (approximately 10 m – 100 m) communications in point-to-point or mesh networking configurations. In the case of GSM, once a *General Packet Radio Service* (GPRS) session has been opened, sending data to any server on the Internet is straightforward. In the case of 802.15.4, further links in the communication chain need to be designed and built in order to get data back to a central server; see Section 8.1 for an example.

Another possibility for sending data to the server is to simply store measurements locally on the device, then recover the device and export the data after the experiment is complete. This is an important backup plan, but is not appropriate for the kind of real-time estimation we are trying to build.

Alters trajectory in river system

We built four generations of drifting water sensor, of which only one had propulsion/actuation technologies. Purely passive Lagrangian sensing is practiced in oceans and rivers around the

Figure 3.2: A high-velocity deployment scenario. Photo credit: Jean-Benoît Saint-Pierre.

world. Actuation is clearly an optional function; however, our experience in river deployments suggests that it is essential for reliable, long-term operation in unstructured environments. This question was introduced in Section 1.3. Chapter 5 discusses obstacle avoidance problems in rivers, and our control approach to the problem; Chapter 8 documents field experiments where the motion of actuated and non-actuated fleets in rivers is compared.

3.2.2 Functionally Determined Properties

Durability

Drifting sensors are subjected to rough handling as they are brought to and from the field, shocks caused by being thrown into the water, possibly from height or from a fast-moving vehicle, and collisions and other traumatic events as they move through the river environment. Our field experience shows that insufficiently sturdy equipment will sustain damage in the following key systems:

- Hull cracking or failures at seams and other critical points,
- Water seal failure,
- Electronics modules becoming detached from supporting PCBs due to shock or vibration.

During our field experiment in Stillwater, Oklahoma (first discussed in Section 1.4), we found that our fleet of ten Generation 2 passive drifters were incurring a module-PCB detach

failure at a rate of 1 per experiment day, while shipping the fleet from Berkeley to Stillwater or back via FedEx caused 3 such failures per trip. While we have far too little data to make a formal analysis, as an anecdote it suggests that the rough handling hazards of pre-operation transport are on the same order of magnitude as the collision/trauma hazards of the operation itself.

Buoyancy

Two fundamental conditions for a floating device are that it displace more water than its mass, and (assuming that this is done by enclosing a volume of air) that it remain sealed against the intrusion of water. The distribution of mass within the volume of the device also affects the orientation in which the device will float and the stability of this position when disturbed.

Although every module in the drifter affects the mass distribution and therefore the buoyancy, most of these masses are insignificant compared to the battery, and so this row in the design matrix is simplified by neglecting these lighter objects.

Drag, forward speed, and turn speed

The drag forces induced on a drifter by the moving water around it is key to its performance as a Lagrangian particle and as a propelled vehicle. Drag forces are what enforce Lagrangian behavior in a floating device; they bring the device's relative velocity with respect to the surrounding water to zero. Symmetry of form is key for true Lagrangian behavior. Drag forces work against the performance of a propelled vehicle. A vehicle's maximum velocity will be set when the thrust of propulsion is equal to the drag of moving against the water. Symmetry of form also affects the vehicle's handling characteristics: how fast it can change its orientation, and how stable this orientation is against disturbances.

Drag is determined primarily by the shape and texture of the hull, while forward and turn speeds are determined by the hull as well as by the propulsion modules (motor, gearing, propeller, and shaft seal).

3.2.3 Operational Properties

The distinction between "functional property" and "operational property" is occasionally murky. In this work, we interpret "functional" properties as relating to *what* the system does, and "operational" properties to the *extent* or *limits* to performing those functions. Thus, "gathers position measurements" is a functional property, while "accuracy of measurements" is an operational property.

Mission time

Mission time usually refers to the maximum time the device can be deployed in the water before some consumable quantity or resistance against damage is exhausted. In practical terms, this is set by the power consumption of the electronics and the energy capacity of the battery.

No water seal is perfect, particularly the seals on a rotating propeller shaft, so the mission time can also be constrained by the expected leak rate and the amount of water necessary to cause failure or damage; however, this leak rate is so slow that it is not really a binding constraint.

Data storage may limit the mission time, if the measurements gathered by the device exhaust the available free space. Practically speaking, this is not a concern.

Measurement accuracy

The accuracy of the position and velocity measurements directly affects the quality of the assimilation or other computations. This FR depends on two design parameters: the choice of GPS module and the mass distribution of the drifter. GPS modules have varying degrees of accuracy, depending on their receiver architectures, analog front end quality, and solution engine performance. The position of the GPS antenna relative to the waterline affects the view of the sky it will have and the quality of incoming GPS signals, which will also limit GPS performance. In general, the higher the antenna is off the water, the better the GPS reception will be.

Communication range and bandwidth

These properties determine the success of exporting data from the devices to the server where assimilation takes place. Choice of radio module affects these values. The position of the antennas relative to the waterline affects the range of the wireless communications; the waterline is determined by the mass distribution in the device.

Data storage

As mentioned above, storing all the gathered measurements in order to manually export them after the operation is an important backup to the wireless communication systems. The amount of data storage required will depend on the periodicity of the measurement gathering.

Data storage is also essential for storing environment maps or control feedback maps; for example, the safety control scheme described in Chapter 5 relies on large pre-computed feedback maps.

In the design matrix, data storage is shown as being determined by the processor module; expanding the data storage is usually just a matter of adding a larger MicroSD card to the

Figure 3.3: Retrieving Android drifters with small nets. Photo credit: Berkeley Lab — Roy Kaltschmidt.

embedded computer.

Reliability

The reliability of technical systems is a field of study all of its own. Every single design parameter can plausibly be connected to reliability. However, our field experience has shown that the following aspects are most critical for reliable operation:

- Integrity against water leaks,
- Integrity against module-PCB and connector-PCB detach failures,
- Graceful degradation of functionality as battery capacity is exhausted,
- Meeting real-time deadlines for embedded control,
- Graceful degradation of functionality in the face of software bugs.

3.2.4 Ergonomic, Aesthetic, and Distribution Properties

Carrying properties

Most experimental operations involve team members physically handling the drifters. Examples include loading, unloading, deploying into the river by dropping or throwing,

and retrieving from the river by picking up out of a boat. The overall mass of the device is obviously very important, but a handle is also a key feature for improving carrying properties.

Storage and transportation properties

Over the years, we were surprised by how important it was to be able to pack the drifters in a box. The preparation time for any operation depended strongly on the amount of time needed to load and unload the drifters, and the number of vehicles needed (both ground vehicles to get to the site and water vehicles to move around the experimental region) depend on how compactly the drifters can be stored. Passive drifters, with very regular, cylindrical shapes, were fairly easy to pack and transport using any convenient plastic storage container. The active drifters, however, with their protruding motor pods and relatively sensitive propeller shafts, were not easy to fit into boxes. There is exactly one easily available model of modular plastic box that can fit four Generation 3 drifters. Our operations before we found this box involved many difficult hours of packing and loading drifters, and many failures due to side loads on motor shafts. In retrospect, our mechanical design should have placed "fits into an easily available box" as a primary consideration.

Safety

The hazards posed by our devices to our research team members and to members of the public should be understood and, as much as possible, mitigated or eliminated. The actuated drifters operate autonomously, which means that we must assess these dangers even in case of unexpected or erroneous behavior from the system. Fortunately, the propulsion capabilities of the device are usually incapable of causing any real damage to people; see Section 3.3.5 for more details. The safety consequences of battery selection are also discussed in Section 3.3.6.

Both active and passive drifters pose a hazard to the public in case of collisions between boaters and drifting sensors. Drifters that are small enough to carry by hand are usually too small to pose a risk to most water traffic. Our field experience includes numerous collisions between our research boats and drifters (mostly unintentional, but sometimes on purpose to test the potential for damage). It seems extremely unlikely that a boat or motor propeller could ever be damaged by our device. There are some edge cases, such as waterskiiers or jetskis, where this may not hold.

Visible at a distance

The drifters should be easy to see from a distance, to make it easy for field personnel to retrieve them, and to make it easy for others to avoid them. This is a function of the waterline on the drifter, as well as the design of the hull and any "flagging" attached to the hull to improve visibility.

Research Workflow

Equipment that is to be used for research has implicit requirements of flexibility and ease of development. In order to efficiently test new algorithms and control schemes, the choice of processing hardware and software architecture must facilitate experimentation and frequent updates. Ideally, a simulation environment should be available, so that prototype algorithms can be tested and verified without expensive field work.

3.2.5 Delivery, Planning, Manufacturing, and Economic Properties

Commercial Off-the-Shelf Sources

"*Commercial off-the-shelf*" (COTS) is an informal term to describe the availability and lack of restrictions on a particular technology or piece of equipment. As a research organization building our own equipment in relatively small quantities, we must restrict the parts and materials we use to those that are available quickly, in small quantities, without regulatory restrictions. This is the critical determining factor for most module selection problems; there are many high-performance options for most of the modules we need, but they often require large order sizes or long lead times to source. We learned this lesson the hard way when building the first generation of drifters; the GPS modules we selected were only available from one distributor, and they failed to meet their delivery schedule due to supply chain problems. Our first experiment was a success only due to the generosity of fellow researchers at a much-maligned institution of higher learning in the South Bay, who graciously loaned us an assortment of GPS modules at the last minute. Patching in three different kinds of GPS module to our systems taught us a valuable lesson about making sure that the essential equipment needed for success is actually available within the necessary timeframe.

Appropriate Manufacturing Techniques

Components of the system that are not available commercially must be manufactured from basic materials and components. In earlier generations, these components were often manufactured by graduate and undergraduate student researchers at UC Berkeley. In later generations, we outsourced many of these processes to custom manufacturing companies. Design choices that allow easy home-brew manufacturing, or easy outsourcing of the labor, are very important to the delivery time and overall cost of the device.

Materials Cost

Almost every physical aspect of the design affects the material cost, although some items have been excluded from the design matrix due to trivially low costs.

Manufacturing Cost

Assuming that the design is manufacturable at all, the overall cost of the device can be reduced by controlling the amount of time or the number of operations required to produce the components.

Assembly Cost

Assembly can be defined as "combining two or more components into a physically connected functional group, using simple tools but highly individualized processes". Under this definition, soldering a surface mount resistor to a PCB is manufacturing (not a simple tool, fairly standardized process) while driving a machine screw with a screwdriver to attach a motor pod to a drifter hull is assembly (simple tool, but the unique shapes of the pod and hull make it an individualized process). Assembly is different from manufacturing because it resists automation, making outsourcing for small quantities prohibitively expensive. Therefore, the assembly processes for building research-scale quantities of drifters will almost certainly be in-house; the cost is therefore not in money, but in research personnel time.

Maintenance and operating costs

Some costs are not incurred during the manufacture of the device, but over its operating lifetime. Often, these costs are tied to the amount of actual operating time. Replacement, reconditioning, and repair are some examples of the processes that incur maintenance costs. GSM modules incur operating costs through monthly fees and data transfer fees.

3.2.6 Law Conformance Properties

These properties are not included in the design matrix.

FCC Compliance

Were we to manufacture our floating sensors for sale, we would need to go through a certification process to ensure that they do not produce RF radiation that interferes with other equipment. Small-scale research projects that manufacture their own equipment can, luckily, skip this process, although we are still liable for harmful interference produced by our devices. The modules that we buy have gone through an FCC certification process, and the circuitry we build does not include RF amplifiers or oscillators, so we make the (probably safe) assumption that we do not violate any emission standards.

Maritime Law Compliance

The regulations that pertain to dropping research equipment into waterways under low supervision are unclear, and none of our team members have any legal qualifications. We

have informally contacted U.S. Coast Guard personnel for guidance; their response was that our equipment is small enough to not pose any real hazard, and that in any case, boat operators are always responsible for safely navigating their craft in the presence of obstacles.

3.2.7 Liquidation Properties

Minimize Hazardous Substances

Once a drifter is unneeded, damaged beyond repair, or obsolete, it needs to be disposed of. Modules can be recycled into new drifters, but some components will enter the waste stream. Reducing the use of hazardous substances makes responsible disposal easier. The major components where this is a concern are the battery and the electronics modules, including the PCBs.

The design should also account for the possibility of losing drifters in the field — in essence, an unplanned total liquidation into an unauthorized waste stream. Here, too, minimizing the use of hazardous substances reduces the impact and environmental cost.

3.3 Design Parameters

Generation 1, 2, and 3 of the drifters were designed iteratively, with many of the design decisions from previous generations carrying on to the next. In this section, we will look at how the design parameters of the drifters were chosen and how this satisfied (or did not satisfy) the relevant functional requirements. The Android drifter represented a major change in design approach; it shares some mechanical heritage with the earlier generations, but most of the electronics and software choices were completely different.

In this section, we will consider Generation 1, 2, and 3, and Android where there are commonalities. More emphasis will be placed on Generation 3, as it was manufactured in the largest quantities and represents the end point of this design process. The Android drifter will be considered separately in Section 3.4.

3.3.1 Hull

As shown in the design matrix, the design of the hull must satisfy several functional requirements:

1. Reliably enclose the inner components, protecting them from water and physical shocks,
2. Support Lagrangian behavior in a fluid stream; that is, present a large, symmetric drag,
3. Support actuated behavior; that is, present an asymmetric drag, with very little drag in the forward direction but high drag in other directions,

4. Facilitate operations by being easy to carry, store, and transport, and easy to see at a distance in water,

5. Have a low materials and assembly cost, using easy-to-source materials and appropriate manufacturing processes.

Items (2) and (3) alone are in direct conflict. The functionality of the device as a Lagrangian sensor and as a propelled vehicle have contradictory implications for the shape of the hull.

Three of the four generations of drifter are purely passive, and so this conflict does not arise. In this case, symmetric forms are best, to ensure that the device does not have a bias when immersed in flowing water. Spherical shapes are the most symmetric, but are difficult to manufacture; given that the device will be floating in a river environment, where the horizontal currents dominate the vertical currents, a form with planar symmetry will suffice. A vertical cylinder is therefore a good solution.

During the design of Generation 3, the actuated drifter, we were confronted with the contradictory goals of sensor versus vehicle, as described above. We chose to prioritize the sensor behavior over the vehicle behavior, and so we continued with the vertical cylinder design.

The hull design for Generation 1 and 2 incorporated a segment of fibreglass pipe for the upper body. This seemed like a good design, because it was light, strong, easy to acquire, and easy to manufacture (we only needed to cut it to length). However, the rest of the hull required a number of custom components that needed to be manufactured in the lab, which greatly added to the complexity and cost of the design. The top cap was made from vacuum-formed polycarbonate, and the bottom hull was made from cast fibreglass. The bulkhead between the two hulls, and the flanges that sealed them shut, were machined from aluminum in the UC Berkeley student machine shop. The bonds between the components (pipe to flange, pipe to polycarbonate cap, fibreglass to flange) were epoxy adhesive. This sealing technique proved to be difficult to execute reliably (requiring hand-roughening of the surfaces to be joined) and prone to failure over time (rough handling caused cracks to develop in the epoxy joint, leading to slow leaks). In order to maintain reliability, we needed to conduct a 24 hr submersion test of all drifters before every field experiment; this increased the logistical complexity of experiments, and caused frequent last-minute reductions in the number of drifters deployed.

During the design of Generation 3, we discovered an off-the-shelf component that would simplify our hull design and improve reliability. Domestic water filtration equipment uses filter canisters to house replaceable filter cartridges, as shown in Figure 3.4. One such filter canister manufactured by Pentek features transparent, UV-resistant polyvinyl chloride (PVC) body, an internal thread for attachment, and a captured silicone O-ring for waterproofing. The transparent material allowed easy inspection of the O-ring, which improved our waterproofing reliability to the point that pre-experiment submersion tests were not

Figure 3.4: Two inline water filters in domestic use. Photo credit: `thedriversseat.com`.

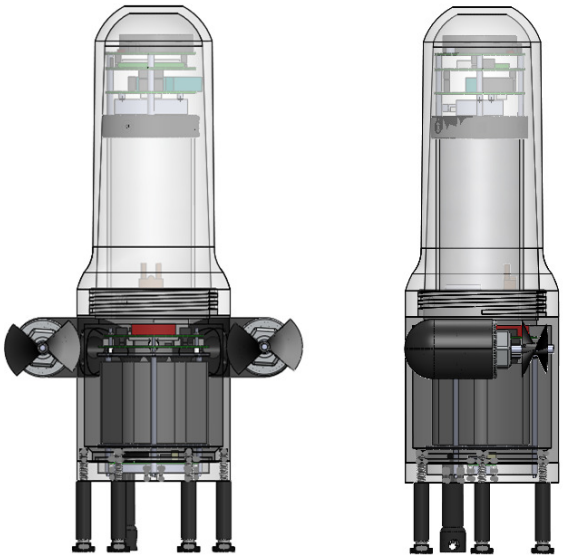

Figure 3.5: CAD drawing of Gen 3 drifter.

necessary. Manufacture of the mating part was simplified by the internal thread, because external threads are easy to machine than internal threads.

The Gen 1 and 2 hulls were made highly visible by painting them a bright yellow. The transparent Gen 3 and Android hulls are not easy to see on their own, making bright flagging necessary.

The other components of the hull were custom manufactured, but we chose a different technology to facilitate outsourcing of the manufacturing job. Instead of hand-casting fibreglass hulls, we designed parts to be machined out of Delrin. Delrin is a relatively expensive polymer material, but it is easy to machine and has excellent resistance to water absorp-

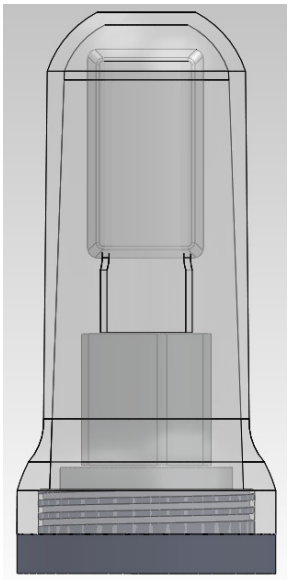

Figure 3.6: CAD drawing of Android drifter.

tion. The remaining components of the hull were designed for *computer numerical controlled* (CNC) machining, which made them easy to outsource to short-run machine shops.

After the successful design of the Generation 3 drifter for improved reliability and ease of manufacture, the Android drifter was designed using the same technologies. The Pentek water filter canister was used for the upper hull, and a Delrin part was CNC machined for the lower cap.

3.3.2 Hull seal

Many methods can be used to create a low-pressure waterproof seal between two static parts. Examples include:

- compression fitting,
- permanent joining via welding, soldering, brazing, etc.,
- crush gaskets,
- hydraulic seals like O-rings, T-seals, V-rings, etc.

The sealing system must be reusable, because the hull will be opened and closed repeatedly to access the components inside. Of these methods, the simplest and most reliable reusable method is the O-ring. The O-ring is a torus of compressible material that is seated in a groove and compressed between two non-compliant surfaces. O-ring manufacturers publish handbooks with specifications for proven, high-reliability designs [90].

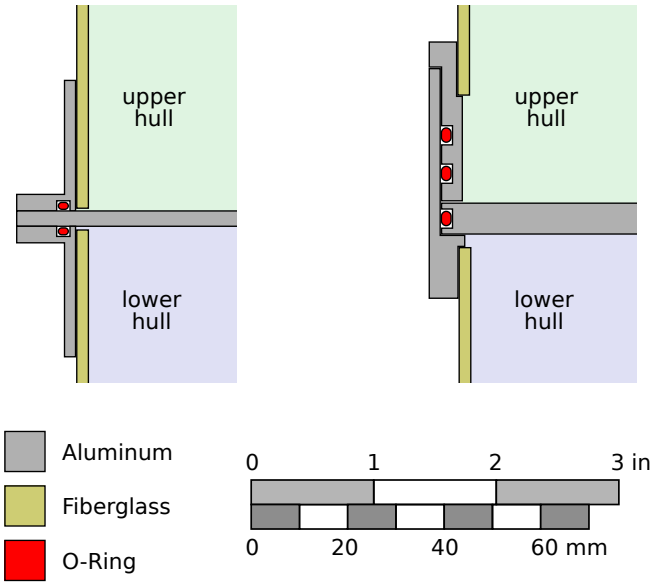

Figure 3.7: Radial and axial seal configurations for early generation drifters.

Figure 3.7 shows two ways of configuring the O-ring seal for a cylindrical hull. Both schemes were considered during Generation 1 and Generation 2 design. The radial style was initially selected for ease of manufacture and used in Generation 1. One key problem with this style is that it is easy to damage the O-ring while sealing the drifter; if the ring has come unseated from the groove and is lying between the sealing faces, it will be severed when the two faces come together.

Axial styles, like the one on the right side of Figure 3.7, are often preferred for submersible vehicle design. In this design, it is harder to damage the O-ring during assembly. Adding backup seals is easy, by adding identical O-rings in a vertical stack. For high-pressure applications (like deep-submerged vehicles), the cylindrical pieces can be tapered, which causes the external overpressure to jam the parts together, increasing seal reliability. These advantages are not as important for a floating vessel, which is not exposed to such high pressures.

We considered the axial seal style for Generation 2, but finally rejected it for reliability and robustness reasons. The radial sealing scheme is more robust to damage and does not require tight tolerances in the aluminum parts. The prototype cylindrical sealing systems were prone to failure when the outer cylinder was knocked out of round through rough handling. The radial sealing system was the final selection. An additional benefit to this choice was that we could simply recycle all the Generation 1 hulls for use in Generation 2.

As discussed above, for Generation 3 and the Android drifter we used an off-the-shelf component that included an O-ring seal in the radial configuration. The O-ring damage problem was mitigated by the transparent PVC body, which allowed easy in-place inspection.

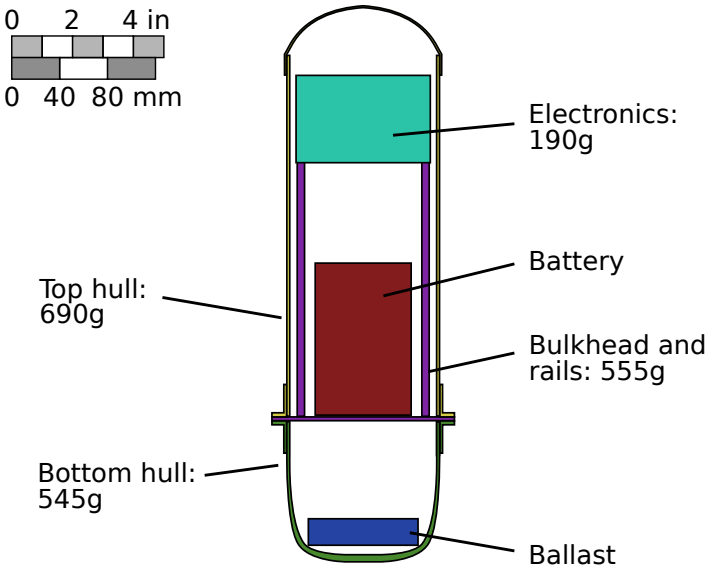

Figure 3.8: Mass distribution in Generation 1 and 2 drifters.

The Generation 3 hull includes several additional components that must be joined together:

- the motor pods to the main lower hull,
- the sensor plate to the main lower hull,
- the motor shaft seal assembly to the motor pod.

Because these seals do not require frequent disassembly, we initially tried to use adhesives such as epoxies or marine caulks to form semi-permanent seals; but these sealing methods are simply not reliable enough. Early Generation 3 prototypes were plagued by leaks until we replaced every one of the joints listed above with properly designed O-ring seals.

3.3.3 Mass and Mass distribution

Hydrostatic analysis places the following restrictions on a vertical cylinder that must float in the river:

- The total mass of the vessel must be less than that of the water displaced by an identical volume, otherwise it will sink instead of floating.
- In the desired orientation, the center of buoyancy must be above the center of mass, otherwise it will rotate to some other configuration (horizontal instead of vertical).

The dynamics of the floating body are more complicated, but as a first order approximation, we can consider the floating object as an ideal pendulum in which the length is the distance between the center of buoyancy and the center of mass; in other words, the floating object has a natural period of oscillation, which can be made longer by increasing the distance between the two centers.

The total mass of the drifter also sets its waterline, which has consequences for the functional requirements:

1. A low waterline (more of the drifter above the waterline) exposes more area to the wind, which reduces its effectiveness as a Lagrangian sensor.

2. A low waterline means the antennas for the communication modules can be placed further away from the water surface, which improves communication range and reliability.

3. A low waterline makes the drifter easier to see at a distance and less of a navigation hazard for other users of the waterways.

We can mitigate the visibility issues through flagging or hull coloring, but (1) and (2) represent another fundamental functional conflict in the drifter design. Again, the Lagrangian sensor function took priority in our design; all four generations of drifter have very high waterlines, with the antennas located within 1 cm of the waterline. Communication reliability definitely suffered from this choice.

Figure 3.8 shows the major masses in the Generation 1 and 2 drifters. Placing the antennas near the waterline also means locating the electronic modules high as well, which works against the goal of placing the center of mass below the center of buoyancy. The battery can be placed at the bottom of the drifter, as well as ballast. The Gen 1 and 2 drifters use a 200 g battery and a 600 g encapsulated lead weight as ballast, while the Gen 3 drifter uses a 900 g battery and no ballast.

3.3.4 Flagging

Adding a highly visible flag to the top of the drifter seems like an easy way to satisfy the requirement that the drifter be visible at a distance. We discovered, however, that a tall flag will often present enough drag to the wind that it acts as a sail, compromising the Lagrangian sensor function. For Generation 3 and the Android drifters, we settled on a low-profile fluorescent orange handle made of duct tape. The handle also makes carrying and retrieval easier.

3.3.5 Propulsion Chain: Motor, Gearing, Propeller, Shaft Seal

The four design parameters directly involved in propulsion have many common dependencies with the functional relationships.

Figure 3.9: Left: three flagged drifters under wind influence. Right: Android drifters in tub, showing duct tape handle. Photo credits: Jean-Benoît Saint-Pierre, Jonathan Beard.

Component cost and availability were the most significant drivers of the component selection and design. A common design choice for small wheeled ground robots is to use a DC motor with an inline gearbox to reduce the rotation speed of the output shaft, allowing greater output torque while allowing the DC motor to operate in a more efficient high-speed regime. Typical gearbox ratios for these applications are 50:1 or 100:1. Propellers normally spin faster than wheels, though, and so our ideal gearbox ratio would be 5:1. We were unable to find an easily available gearbox in this ratio within our space and cost budget, and so we instead used a DC motor with direct coupling to the output shaft (i.e. no gearbox). The DC motor we selected was a specialty hobby RC motor intended for model trucks — its windings were designed for low rotation speed, high torque applications. Ironically, this made it a very good match for our target propeller shaft speed with direct coupling.

We investigated several possible actuator configurations, including a single propeller with a rudder, a single propeller mounted on a rotating external pod, and dual parallel differential-drive propellers. In the end, we selected the differential-drive configuration. Each one of these solutions has two degrees of freedom in the actuator: in the case of the rudder and the rotating pod, the two degrees of freedom are the propeller speed and the steering angle. In the case of differential drive, the two degrees of freedom are two propeller speeds. Having one kind of actuator, repeated twice, is a simpler design than two different actuators. We preferred the simpler design.

The shaft seal was a challenging component selection problem. Unlike the other water-proof seals on the vehicle, which were essentially *static* seals, the shaft seal must keep out water while allowing the shaft to rotate. While O-rings can still be used in these applications, the design of the gland (the assembly of the static and dynamic part, including the O-ring seating) is much more complicated [90]; these sealing schemes fail the "appropriate manufacturing technique" requirements. We searched for a fairly simple, inexpensive, component solution (i.e. something we could install easily into the assembly, as opposed to a specialized

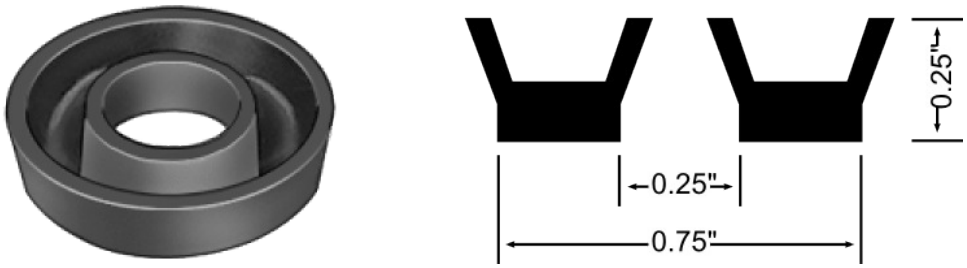

Figure 3.10: Isometric view and cross sectional drawing of U-cup selected for shaft seal.

design that we would have to manufacture ourselves) that would provide a reliable rotary seal while not adding too much friction to the shaft rotation. In the end, we selected a U-cup ring. Similar to an O-ring, but with a different cross-section (see Figure 3.10), it will compress so that more of the rubber material presses against the rotating part. This solution proved to be simple, reliable, easy to assemble and service, and inexpensive; however, it introduced a lot of friction to the assembly.

The only propeller options with reasonable availability terms were hobbyist equipment for RC boats. Precise specifications or mechanical drawings are not normally available in this market, and so instead of a principled component selection, we opted to experiment with different combinations of motor, shaft seal, and propeller until we found an acceptable solution.

The steady state speed of a vehicle in water can be estimated by balancing the propulsion force, F_{prop}, against the drag force, F_{drag}. Using a simple model for the drag,

$$F_{\text{drag}} = \tfrac{1}{2}\rho C_d A v^2 = F_{\text{prop}} \tag{3.1}$$

where A is the cross-sectional area (0.032 m^2), ρ is the density of water, and C_d is the dimensionless drag coefficient. An estimate for C_d is 0.8, based on calculations for an ideal finite cylinder [135].

We set a target maximum forward velocity of 0.3 m/s for the vehicle. This requires a propulsion force of 1.15 N or 0.58 N per motor. We developed a testbench with a force load cell attached to a motor pod in a bucket of water; see Figure 3.11. The input power to the motor was delivered by a constant voltage supply switched by a PWM controller. We chose the component combination that was the most efficient near the 0.58 N target.

Safety concerns were another important constraint on the propeller selection. One very efficient, easily available propeller had to be rejected, because its stainless steel construction (with sharp edges) made it akin to an unprotected blender blade. Our selected propeller is made of thin plastic material, and is much safer if touched accidentally while running.

The actual speed of the vehicle was estimated during tests in an outdoor tank at the UC Davis Bodega Bay Marine Laboratory. By driving back and forth in the still water of the tank, while receiving GPS signals, the speed of the vehicle was estimated. Two techniques

Figure 3.11: Motor force test platform: submerged motor unit is attached to an extended arm which pivots against a force transducer.

were used: first, the GPS velocity signal itself was averaged over a run across the pool, providing an estimate of 0.264 m/s with a standard deviation of 0.036 m/s. An alternate method is to take a finite difference of the GPS positions, spaced 6 s apart: this method results in an estimate of 0.242 m/s with a standard deviation of 0.033 m/s. It is expected that the finite difference estimate would be lower, because the drifter does not travel in a perfectly straight line. Figure 3.13 shows the time series of the speed estimates by the two methods during a run across the pool, and Figure 3.14 shows the GPS positions gathered during the run.

3.3.6 Battery

Rechargeable electrochemical batteries are the cheapest and most convenient way to store the electrical energy needed for the on-board electronics and motors. Energy storage systems are often evaluated in terms of their *energy density* (maximum stored energy per unit mass) and *power density* (maximum output power per unit mass). For the mission durations we are interested in (tens of hours), the battery's energy capacity will be the binding constraint. A summary of battery chemistries and their energy densities is shown in Table 3.2.

A power budget for the electrical systems in Generation 3 is shown in Table 3.1. Earlier generations are similar, except without the propulsion systems. As discussed earlier, a 200 g lithium ion battery was selected for Generation 1 and 2, mainly for cost reasons. The battery could have been much larger by reducing the mass of the ballast.

In Generation 3, the electronic components and the motors require different input voltages. Although it would be easy to incorporate a voltage regulating circuit to lower the voltage for the electronics, allowing all systems to share a common battery, there are reasons

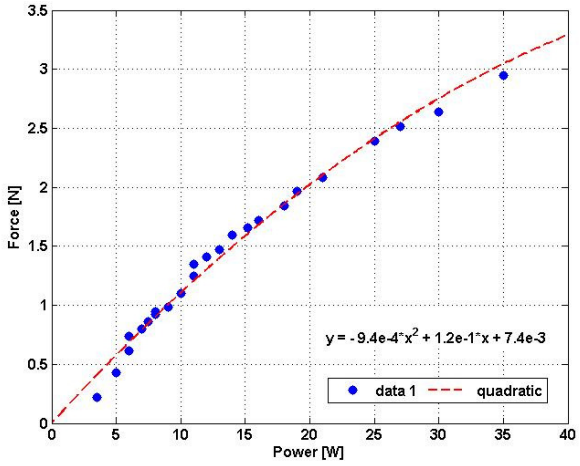

Figure 3.12: Output force as a function of input power for chosen motor/shaft seal/propeller combination.

why it makes sense to keep the motor power separate from the electronics power. First, the inrush current when motors start can cause a voltage slump which could cause drop-outs in the electronics power; second, brushed DC motors can generate noise on the power lines that could adversely affect the electronics; third, if the motors are run long enough to exhaust the battery, and the electronics are on the same battery, the vehicle will no longer be able to gather data or transmit its location. The first and second problems could be mitigated by careful design of the voltage regulator circuitry, but it is easier to simply keep the two power sources separate.

For Generation 3, our selected design was a hexagonal pack of 19 cylindrical lithium ion cells, with five cells dedicated to the electronics (3.7 V, 170 kJ, allowing 80 hours of electronics operation) and 14 cells dedicated to the motors (7.4 V, 480 kJ, allowing 74 hours of operation at the 10% duty cycle). See Table 3.1 for component power requirements.

An iterated revision of Generation 3 was designed and implemented to improve some aspects of reliability and performance. This version is called "Generation 3.1". We kept the hexagonal pack of lithium ion cells, but changed to 11.1 V for both motor power and electronics power instead of 7.4 V and 3.7 V. The motors we used in both Generation 3 and 3.1 are rated for 7.4 V, but we decided that the substantial cooling effect of water immersion meant that we could safely exceed this rating. Switching to 11.1 V allowed us to reduce losses in other parts of the power chain, leading to greater efficiency. The electronics have their own 11.1 V battery; a switching regulator lowers the voltage to 3.7 V, the nominal voltage for the original design. We chose to use a high voltage and a switching regulator to mitigate possible voltage slumps which caused some reliability problems with the original design. Apart from this change, there were very few design differences between Generation

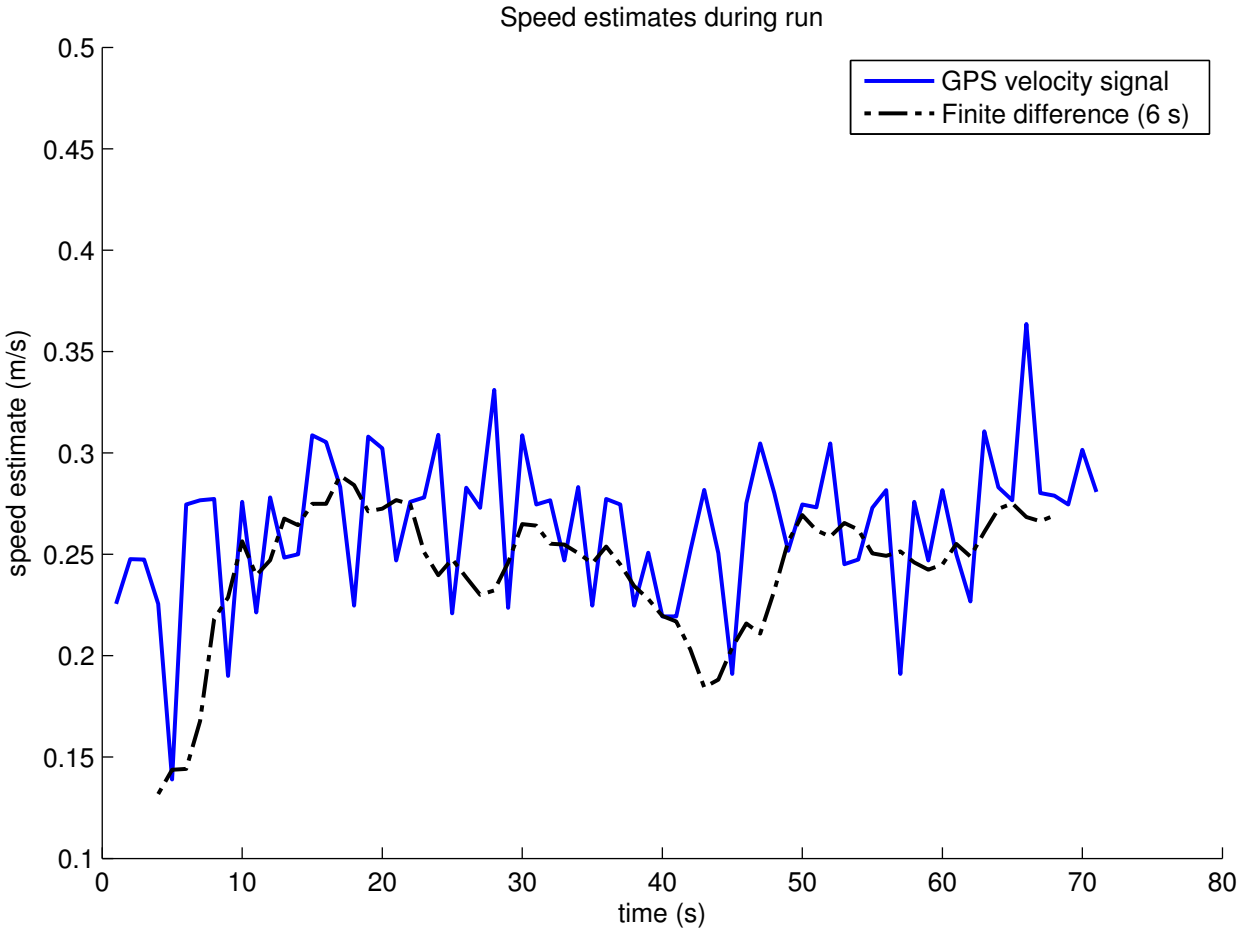

Figure 3.13: Estimated speed of Gen 3 vehicle in test tank.

Component	Voltage	Current	Duty cycle	Power
Overo	3.7 V	0.3 A	50%	0.51 W
G24	3.7 V	0.2 A	5%	0.04 W
XBee-PRO ZB	3.3 V	0.3 A	5%	0.04 W
Motors	7.4 V	2.4 A	10%	1.8 W

Table 3.1: Component power requirements.

3 and 3.1, and in the rest of this work we will refer to the two designs as "Generation 3".

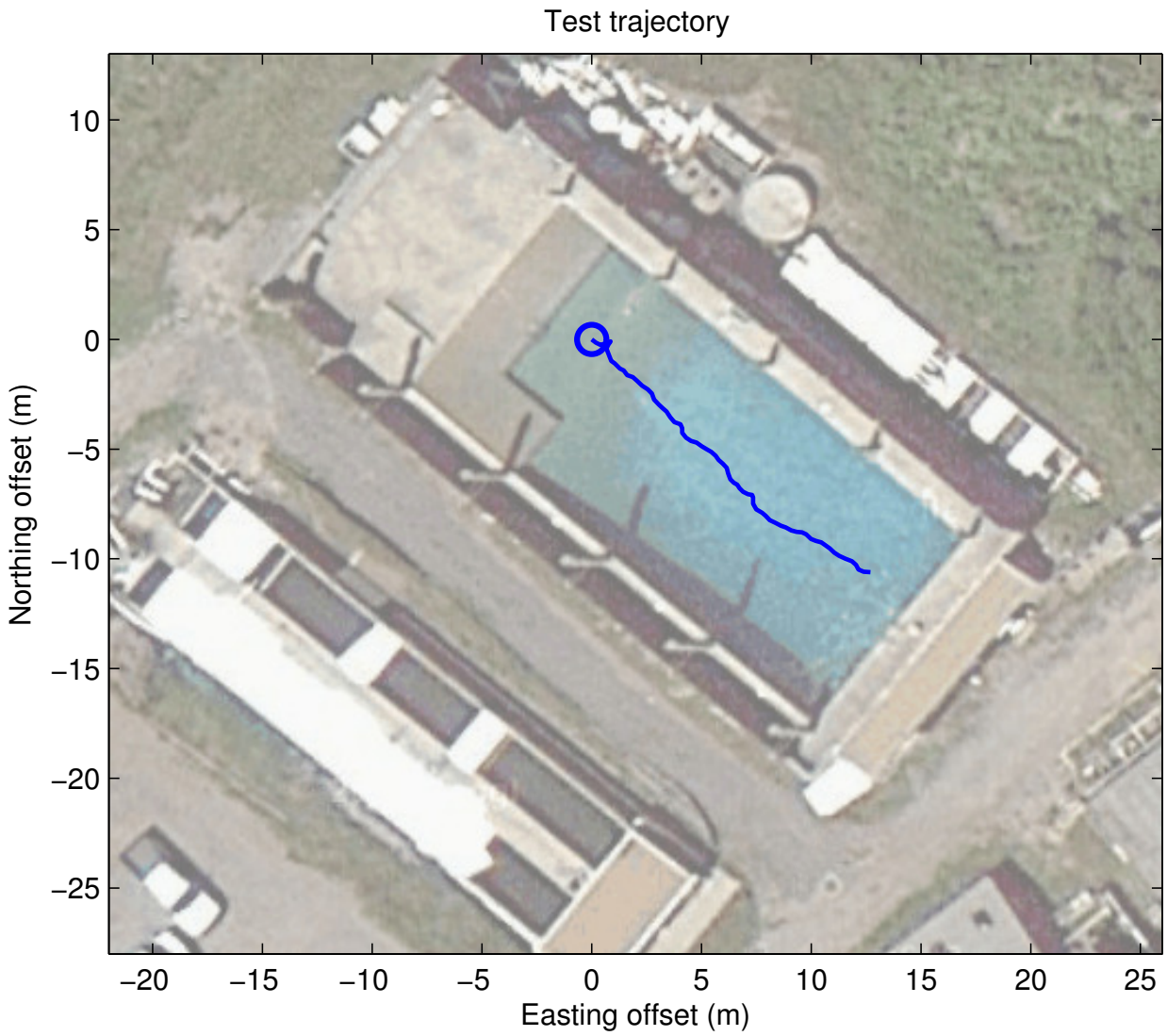

Figure 3.14: GPS track of test trajectory in Bodega Bay test tank.

3.3.7 GPS Module

Selection of the GPS module is based on cost, availability, power consumption, ease of use, and accuracy of readings. For Generation 1 and 2, we selected the Thales AC-12 OEM GPS module. This module was a satisfactory choice for the early generations, but by the time we were developing Generation 3, there were smaller, less expensive, easier to integrate modules available. The earlier generations of OEM GPS modules often had an external connector for the GPS antenna, leaving antenna selection and configuration up to the system integrator. While this flexibility might be an important advantage in some applications, in our case it

Chemistry/format Example	Specific energy	Energy density
Lithium ion Panasonic NCR18650 [88]	840 kJ/kg	2100 kJ/L
Lithium polymer Sanyo UPF673791 [100]	760 kJ/kg	1800 kJ/L
Nickel-metal hydride Panasonic HHR110AAO [87]	200 kJ/kg	620 kJ/L
Lead-acid Panasonic LC-P0612P [86]	130 kJ/kg	370 kJ/L

Table 3.2: Representative battery capacities for various chemistries.

was just another component for us to source and assemble. Early Generation 3 prototypes that used the AC-12 had serious GPS reliability issues that we finally traced to the antenna. The diameter of the Gen 3 hull is slightly smaller than that of the earlier generations; this means that the bending radius of the coaxial cable that connects the antenna to the GPS module is also slightly smaller. We were violating the bend radius limit of the antenna cable, causing signal degradation and GPS failures. We found a newer GPS module, the MediaTek MT3329, that combined the GPS electronics and antenna into a single module. Removing the GPS antenna as a separate component increased our GPS reliability and decreased the component cost and assembly time of the vehicle.

3.3.8 Compass

We incorporated an electronic compass into the Generation 3 drifter to assist with navigation. Although a GPS unit will provide heading information, it is normally not very accurate; it is much easier to use information about the local magnetic field to determine orientation. The Honeywell HMC6352 2-axis compass provides heading information at 20 Hz, which is definitely fast enough for doing feedback heading control with the motors.

3.3.9 Communications: GSM Module and 802.15.4 Module

There are three primary reasons to communicate with a drifting sensor in the field:

1. To discover the local water conditions for real-time sensing applications;
2. To track the sensor's position for retrieval, or to query its health and operational status (battery energy remaining, etc);
3. To share data between vehicles for multi-vehicle control applications or with the command center for remote actuation of the drifter.

These goals have different requirements for transmission range, bandwidth, and latency. Goals (1) and (2) above have very low bandwidth requirements; 0.01 kB/s – 0.1 kB/s, with up to 30 s latency, would be acceptable. However, these transmissions will need to be sent over distances of kilometers or greater. By contrast, goal (3) requires more bandwidth and lower latency; 2 kB/s with less than 1 s latency. For multi-vehicle control applications the vehicles can be assumed to be relatively close: 100 m is a reasonable range. These diverse requirements can be best addressed with two separate communication networks.

The range requirements of the long-range communication system pose a challenge in the estuarine environment. The vehicles themselves could be up to 10 km away from their origin. Islands, levees, trees, and buildings are all interfering obstacles. The best self-contained solution would be to erect a communication tower on-site to minimize the fading through these obstacles. Although truck-mounted portable tower solutions exist, there is a more convenient option: the civilian *Global System for Mobile Communications* (GSM) network. Using a GSM module such as the Motorola G24 [75], the drifting sensor can use the GPRS (General Packet Radio Service) of the GSM protocol suite [40] to open a TCP connection to any server on the Internet. The guaranteed minimum bandwidth of a GPRS connection is 9.6 kB/s, but the latency is not guaranteed. Empirical tests show that latency of 5 s –0.1 s is common.

Due to the larger latency, the GSM solution is inappropriate for multi-vehicle control applications. At shorter ranges, we can reasonably expect to have a clean line of sight between the vehicles, and so a low-power point-to-point radio system is appropriate. The emerging IEEE 802.15.4 standard for low-power wireless networks [60] defines protocols for low-powered radios to form mesh networks and transfer small quantities of data (appropriate for sensor networks, automation, or other embedded applications) over the 2.4 GHz ISM frequency band. Some radios that conform to a subset of the IEEE 802.15.4 drafts are branded as *ZigBee* radios. The Digi XBee-PRO ZB is a ZigBee radio that allows short-range, line-of-sight, low-latency data communications between vehicles. One feature of the XBee-PRO ZB that distinguishes it from similar modules is the on-board power amplifier, which increases the transmit power to 50 mW, extending the transmission range of the system. We have observed connectivity at distances of up to 1 km in river environments when using these modules.

3.3.10 Processor

The computational units (both high-level microprocessors and low-level microcontrollers) on board the drifter are responsible for maintaining the other electronics modules (GPS, communications, sensors) and marshalling data to fulfil the Lagrangian sensor mission. However, the computational demands of this role (parsing, storing the data, opening a communications channel, and retransmitting it to the server) are simple enough, by the standards of modern embedded processors, that there is essentially no connection between the functional requirements of these tasks and the design parameters of choosing the processors. As long

as there is any kind of embedded processor on board, fulfilling the basic Lagrangian sensor function is relatively straightforward.

The remaining functional requirements that place constraints on the processor selection are:

- Altering trajectory: Deciding how to use the propulsion is a non-trivial task,
- Data storage,
- Reliability, in the form of graceful degradation in the face of software bugs,
- Research workflow,
- Sourcing constraints and component costs.

There are many possible applications for a fleet of actuated sensor vehicles. A major goal of this research project is to produce a platform upon which new and innovative multivehicle control ideas can be developed. It is therefore difficult to specify *a priori* what computational resources will be required for these roles. Chapters 5 and 6 introduce two different directions for these control problems; hopefully there will be even more such applications in the future. We should therefore err on the side of overprovisioning the processor.

The "reliability" and "research workflow" requirements are best satisfied with a high-level processor that can run a modern operating system. By taking advantage of the process control, multitasking, and interprocess communication features available in such systems, we can both save development time and reach higher levels of operational reliability.

Generation 1, 2, and 3 of the drifters used different embedded computer modules from Gumstix, Inc. With each generation, there were more powerful models available. These modules were selected because they were inexpensive, easy to integrate, and they ran Linux, which has the high-level functions we need and supports a wide array of programming languages.

3.3.11 Microcontroller

An important feature for reliability is to separate the computational tasks that have truly hard deadlines, like motor control, and separating them from the high-level computational tasks. This separation could be logical; a properly configured real-time operating system can guarantee that hard deadlines will be met even while executing complex high-level tasks. However, it is far simpler to separate these tasks onto different computational units. Microcontrollers are a good choice for the deadline-driven low-level tasks of an embedded system. Their simple architecture and ease of low-level programming make precise timing easy, and they are usually configured with ample digital I/O ports. High-level processors often don't have enough serial ports or general-purpose IO lines to manage a large number of integrated devices; the simple ones, like sensors, can be delegated to the microcontroller.

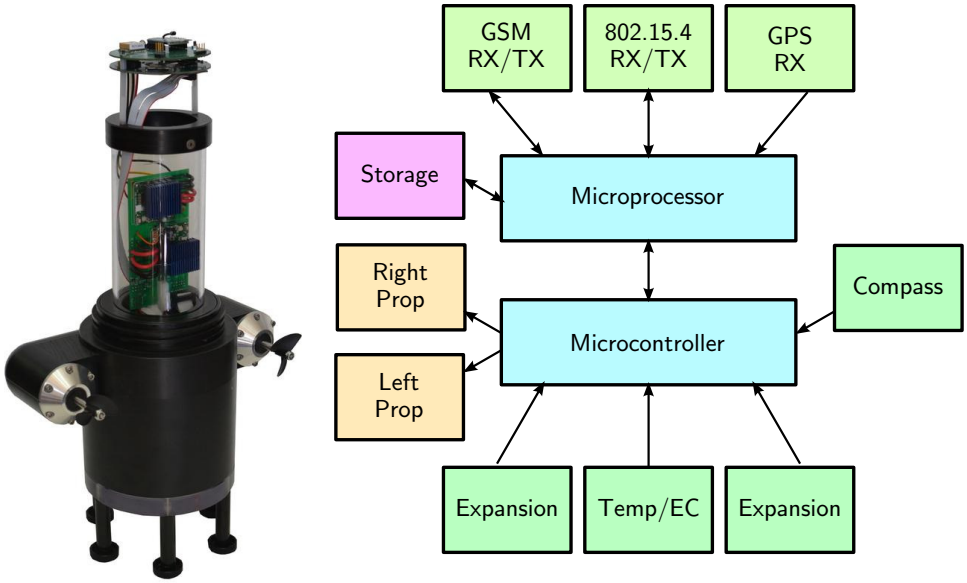

Figure 3.15: Generation 3 electronics block diagram.

We use Atmel's ATMEGA family of microcontrollers. In Generation 3 we upgraded to the XMEGA family, which is a generational iteration from ATMEGA. Figure 3.15 shows the allocation of responsibility between the microprocessor and microcontroller. Earlier generations are similar, without the propulsion units.

The controller design for heading-based navigation was implemented on the XMEGA microcontroller. A discussion of this controller design and implementation is in Section 3.5.

3.3.12 Printed Circuit Boards

Integrating all the electronics modules together was accomplished with custom designed *Printed Circuit Boards* (PCBs). The functional requirements of these boards was generally to provide physical support to the various modules, provide basic electrical support like power filtering, switching, and sensor conditioning, and to connect the analog and digital signal lines between various modules. For Generation 1 and 2, the boards were designed by student researchers and fabricated by external manufacturers, but assembling all the components and connectors was done in the lab by students. While this was perhaps cost effective for small numbers of prototypes, by the time the manufacturing quantities reached around 10 units, any gains were lost in the face of poor assembly reliability. Making the switch to having assembly houses populate the PCBs involved a major learning curve; the design and documentation requirements for outsourcing a PCB assembly job are quite significant. The cost is also non-trivial, and there is a substantial risk of wasting money and time if the

PCB layout is faulty in some way, but these risks are necessary to transition from a research lab that produces one or two prototypes to a research operation that works with tens or hundreds of units.

The scale of the PCB integration design is apparent when we consider the boards that make up the latest Generation 3 device. This consists of six circular circuit boards separated by function. The number of boards is due to the cylindrical shape of the hull, requiring multiple stacked boards to support all the components.

Shown in Figure 3.16 is a breakdown of how the circuit boards are arranged in the drifter unit. From top to bottom, we describe the responsibilities and components of each board:

- The *Top Board* is a 3.1 inch 2-layer circular PCB. It contains the GPS module, compass module, XBee-compatible pin headers, and an ATxmega128A3 microcontroller which reads the compass and can intercept communications to or from the GPS or Xbee modules. The microcontroller is given control of the power domains of the drifter and can thus extend battery life by turning off the high-consumption subsystems such as the main CPU.

- The *Overo Board* is a 3.1 inch 4-layer circular PCB. The board connects to the Top board via two 14 pin board-to-board headers and communicates to the bottom board stack through a 14 pin ribbon cable. It also receives power by a 4-conductor large-gauge cable. This board houses the GSM cell-phone modem, the Overo CPU, and indicator LEDs. A FT4232H USB-to-serial chip provides the Overo CPU with 4 serial ports with which to communicate with the drifter's various modules.

- The *ESC Board* is a rectangular PCB placed vertically in the tube supporting the upper electronics stack. It contains two commercial electronic speed controllers (ESCs) and the level-shifting circuitry to operate them. A 14 pin ribbon cable connects this to the Power board (which provides a connection to the Control board) and a 2 pin battery connector provides power.

- The *Power Board* is a triangular PCB responsible for converting the 11.1V battery voltage to 3.7V to run the electronics and measuring the power consumption via a MAX4173F chip. The voltage conversion is facilitated by a buck converter circuit controlled by a TPS5430 chip.

- The *Control Board* board contains another ATxmega128A3 microcontroller responsible for controlling the direction of the drifter and reading measurements from the water quality sensors below. Originally the compass was to be placed on this board, but we found that between the battery and motors there was too much magnetic distortion for the digital compass to function.

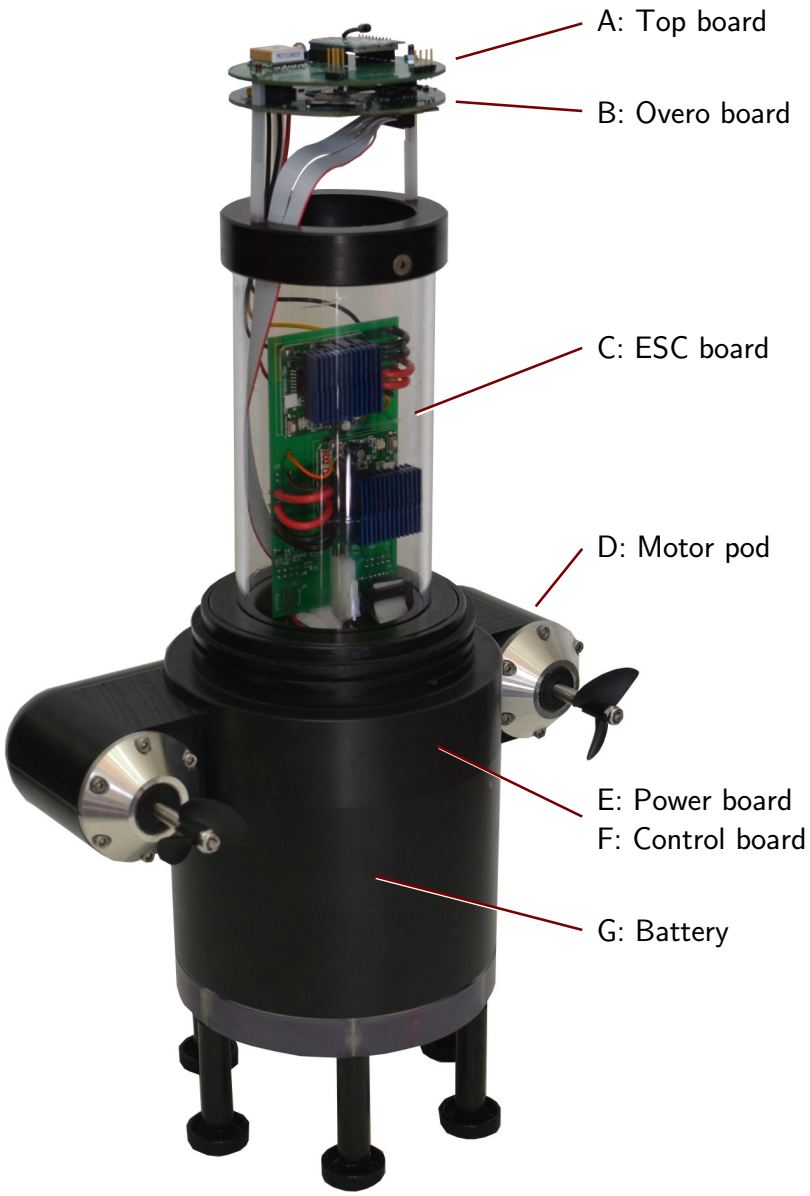

Figure 3.16: Generation 3 Drifter Hardware Components.

3.3.13 Software language

The only practical choice of language for the microcontrollers is C.

In Generation 1 and 2, the high-level software on the Gumstix processor was also written in C. In Generation 3, we designed our software to be primarily written in the Python language. This choice of language offered some important advantages over previous generations which were written in C. Most notably, since Python is a scripting language, it does not require cross-compilation targeting the Gumstix. Thus, we were able to have rapid design cycles, as each code change required only uploading the new code to the drifter. Additionally, the scripts could be edited on the drifters themselves.

On the Android drifter, Java was the best choice, as it integrates well with the Android operating system in use.

3.3.14 Software architecture

Fundamental choices in the software architecture design affect both the reliability of the drifter and the ease of research development. In particular, we set the ability to perform *software-in-the-loop* (SIL) simulation as a high priority. SIL refers to a simulation environment where the external environment is simulated, and the device itself is simulated by running a copy of its software on a simulation platform.

Software reliability was improved by separating the code into modules which would run in independent processes. The most immediate advantage of this strategy is that, if one module experiences errors and crashes, it can be restarted without affecting the rest of the system. We chose to use UNIX IPC sockets for *inter-process communications* (IPC) which enabled a straightforward implementation of a SIL simulator. To serialize and decode messages, we use Google Protocol Buffers [49].

A feature of this modular design choice is that we can write *virtual* modules which emulate the functionality of physical systems. For example, when testing the control algorithm presented in Chapter 5, we wrote a virtual GPS module providing coordinates from a dynamics simulator. The dynamics simulator includes a simulated heading-hold controller which exports a virtual motor interface and takes motor commands via this service.

The intent is that the same production code which runs on the actual drifter hardware can be connected to a simulated drifter. This enables running certain experiments without the overhead of a field operation.

3.3.15 Task periodicity

The power consumed by the electronics modules can, in theory, be reduced if the rate at which various tasks is performed is reduced. The sampling time for sensors, for example, dictates the rate at which data is generated, the amount of storage that must be provisioned, and the amount of power required to send the data to the central server.

#	Item	$	Q
1	Duct tape handle	$1	1
2	Upper PVC hull	$25	1
3	Motorola DEFY	$175	1
4	Rubber bands	$1	2
5	PVC phone holster	$15	1
6	Foam disk	$1	1
7	LiPo battery	$25	1
8	If-found placard	$1	1
9	Aluminum disk	$10	1
10	Delrin base	$50	1
	Total	$305	11

Figure 3.17: Android drifter and Bill of Materials

There are limits to the energy savings available from changing the task periods. For example, the acquisition time of a GPS module is on the order of one minute. Considering only the energy consumed by the GPS module, it makes no difference whether positions are recorded every 1 s or 10 s; the GPS module must be continuously on in order to generate measurements at either of those rates. If the application is such that GPS data could be gathered every 30 min instead, however, there could be significant energy savings realized.

The experimental regions we investigate definitely need sub-minute resolution for accurately representing their flows, so the GPS modules are active all the time. There definitely is potential, however, in being more intelligent about the use of communications modules. Our experimental protocols tended to just transmit the data in real time, all the time, instead of queueing data up to save energy with burst transmissions. There is low-hanging fruit here for energy savings.

3.4 Android drifter: A New Approach

The availability of mobile phones that could easily be configured to run custom software made a new approach to passive Lagrangian sensors possible. A modern smartphone incorporates GPS positioning capability, communications capability via GSM, and a fairly powerful processor. Smartphones running the Linux-derived Android operating system are particularly easy to use for developing custom software.

The Android drifter takes advantage of Android-based mobile phones to fill all of the functional requirements of the passive Generation 1 and 2 drifters at significantly less cost.

The reliability of these professionally engineered, mass-produced devices also exceeds that which a small research lab can produce.

Implementing the Android drifter did have a significant transition cost, in that the Python software infrastructure we had built on the previous generations was not very useful on the new platform. Re-implementation in Java was the best choice, even though there was some duplication of previously implemented functionality. This was, however, a small price to pay for the new opportunities provided by the lower cost and higher reliability device.

3.5 Heading controller design for Generation 3

Real-time control of the motors in Generation 3 is handled by an XMEGA microcontroller implementing a "heading-hold" control. Heading commands are provided by the Gumstix processor. The goal of the heading-hold controller is to drive the drifter forward along a bearing (angle relative to magnetic north). The controller generates *pulse width modulated (PWM)* signals for the ESCs which is ratiometric to the power delivered to the motors. The controller's feedback comes from the HMC6352 compass as tenths of degrees from magnetic north.

We chose to use a *proportional-integral-derivative (PID)* controller [2] to accomplish the control task. Given a desired bearing, θ_{desired}, from the centerline module, and θ_{actual} from the compass, the control law of the heading-hold controller can be expressed as:

$$\begin{aligned}
\theta_{\text{error}}(t) &= \theta_{\text{actual}}(t) - \theta_{\text{desired}}(t) \\
u_{\text{diff}}(t) &= k_d\dot{\theta}_{\text{error}}(t) + k_p\theta_{\text{error}}(t) + k_i\int_0^t \theta_{\text{error}}(\tau)d\tau \\
u_{\text{left}}(t) &= u_{\text{mid}} + u_{\text{diff}}(t) \\
u_{\text{right}}(t) &= u_{\text{mid}} - u_{\text{diff}}(t)
\end{aligned}$$

where u_{left} and u_{right} are the inputs to the PWM generator for the left and right motors and k_d, k_p, k_i and u_{mid} are tuneable constants. The values for the PID constants were determined experimentally by setting a constant desired bearing and using classic PID tuning techniques so that the drifter would travel in a straight line with minimal oscillations. We also ensured that, when giving the drifter a new bearing command, that the step response of the system was stable, a challenge particularly because of the nonlinearity at $0^\circ = 360^\circ$. In practice, we found that using the integral error term was only necessary and useful when θ_{error} was small, so we set a configurable threshold on the error, outside of which, the integrator's state is set to zero. Without this modification, the step response of the system was usually unstable.

Chapter 4

Data Assimilation for the 2D Shallow Water Equation

The Lagrangian drifting sensors described in Chapters 2 and 3 will gather hydrodynamic state information along the path they follow through the system. In order to transform this relatively sparse Lagrangian data into an overall view of the system (which is usually in the Eulerian perspective), a data assimilation technique is required. In this chapter, a variational approach to data assimilation is formulated as a *Quadratic Programming* (QP) problem, which provides a computationally efficient approach. An example of use on an early experiment conducted with the first generation of mobile floating sensor is presented.

4.1 Modelling River Phenomena

River hydraulics can be modeled with *shallow water equations* (SWE) in one or two dimensions [24]. Shallow water equations are a standard constitutive model used in the environmental engineering community and hydraulics community to model river flow; they are commonly used for simulation and control. When dealing with experimental measurements, techniques are required to incorporate them into the model. One such technique is *data assimilation*, which is the process of integrating measurements into a flow model, and which originated in meteorology and oceanography [67].

In the following, we use the SWE as our constitutive hydrodynamic model. We will present the equations, followed by a specific linearization and discretization. For legibility we suppress the arguments for dependent variables. The governing hydrodynamic equations

for the modeled system are [38, 123]:

$$\frac{\partial u}{\partial t} + \vec{u} \cdot \nabla u = -g\frac{\partial \eta}{\partial x} + F_x + \frac{1}{h}\nabla \cdot (h\nu_t \nabla u) \tag{4.1}$$

$$\frac{\partial v}{\partial t} + \vec{u} \cdot \nabla v = -g\frac{\partial \eta}{\partial y} + F_y + \frac{1}{h}\nabla \cdot (h\nu_t \nabla v) \tag{4.2}$$

$$\frac{\partial h}{\partial t} + \vec{u} \cdot \nabla h + h\nabla \cdot \vec{u} = 0 \tag{4.3}$$

Equations (4.1) and (4.2) represent momentum balance, and equation (4.3) represents mass balance. They can be obtained from fundamental principles (conservation of mass for an incompressible fluid, and Newton's second law) in a differential setting [109]. The symbol ∇ is used to denote the gradient operator and $\nabla\cdot$ is used to denote the divergence operator. The variables (x, y) are space coordinates; t is time in seconds; $\vec{u} = (u(x, y, t), v(x, y, t))$ is depth-averaged water velocity in m/s; $h = h(x, y, t)$ is water depth in meters; $b = b(x, y)$ is elevation of bottom surface in meters; $\eta = h + b = \eta(x, y, t)$ is free surface elevation in meters; g is the acceleration of gravity in m/s^2; ν_t is the coefficient of turbulence diffusion, obeying the so called k-epsilon model [38]; and $F_x = F_x(x, y, t), F_y = F_y(x, y, t)$ are friction terms

$$F_x = -\frac{1}{\cos\theta}\frac{gn^2}{h^{4/3}}u\sqrt{u^2 + v^2} \tag{4.4}$$

$$F_y = -\frac{1}{\cos\theta}\frac{gn^2}{h^{4/3}}v\sqrt{u^2 + v^2} \tag{4.5}$$

where $\theta = \theta(x, y)$ is the slope of the river bed; n is the Manning coefficient. The Manning coefficient is an empirical term that depends on the roughness of the channel bed. For this study we took the Manning coefficient to be 0.04 uniformly over the domain. This is a reasonable estimate for a major natural stream without significant brush obstruction on the banks [109]. The friction terms are derived from the classic Gauckler-Manning hydraulic equation [25], using the shallow water assumption (that the depth of the water is significantly smaller than the cross-sectional width of the channel).

4.2 Adapting the model to Quadratic Programming

Following common assumptions in fluvial hydraulics, our first simplification is to neglect the turbulence terms. We linearize the equations about a steady but non-uniform flow

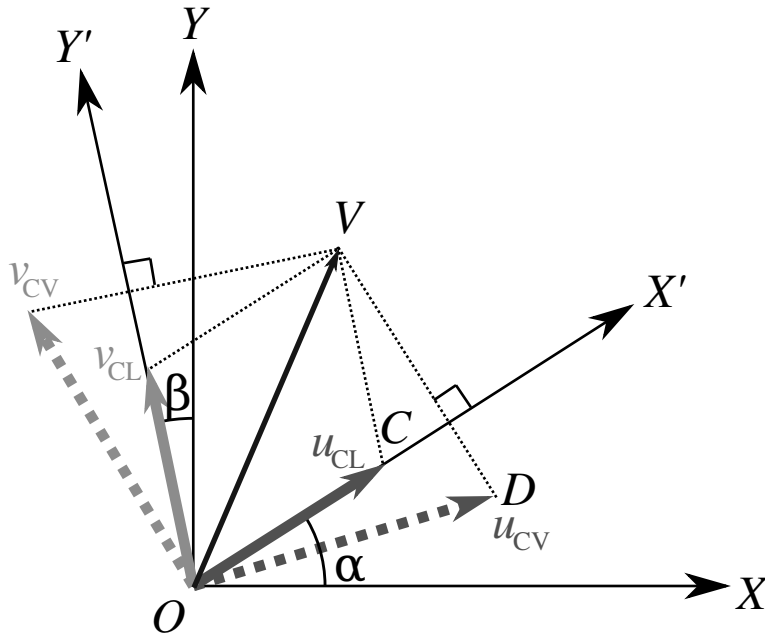

Figure 4.1: Example of non-orthogonal curvilinear axes. OX, OY: global Cartesian axes. OX',OY': local non-orthogonal curvilinear axes. OV: velocity vector. $u_{\mathrm{CL}}, v_{\mathrm{CL}}, u_{\mathrm{CV}}, v_{\mathrm{CV}}$: curvilinear and contravariant components of OV.

$U^0(x,y)$, $V^0(x,y)$, $H^0(x,y)$ that satisfies equations (4.1), (4.2) and (4.3):

$$\frac{\partial u}{\partial t} + U^0\frac{\partial u}{\partial x} + V^0\frac{\partial u}{\partial y} = -g\frac{\partial h}{\partial x} - g\frac{\partial b}{\partial x} + Cu \tag{4.6}$$

$$\frac{\partial v}{\partial t} + U^0\frac{\partial v}{\partial x} + V^0\frac{\partial v}{\partial y} = -g\frac{\partial h}{\partial y} - g\frac{\partial b}{\partial y} + Cv \tag{4.7}$$

$$\frac{\partial h}{\partial t} + U^0\frac{\partial h}{\partial x} + V^0\frac{\partial h}{\partial y} + H^0\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = 0 \tag{4.8}$$

with the choice of

$$C = \frac{1}{\cos\theta}\frac{gn^2}{H^{0\,4/3}}\sqrt{U^{0\,2} + V^{0\,2}} \tag{4.9}$$

as the linearized friction coefficient.

4.2.1 Non-orthogonal curvilinear grid

For general geometries, the river region does not line up well with the Cartesian axes. Discretizing using a Cartesian mesh would be inefficient; the grid size would have to be very fine in order to capture the spatial features properly. Generating a non-orthogonal grid to

efficiently cover the domain while also representing the boundary is a standard problem with many approaches in the literature. For this study, the domain was manually decomposed into large, approximately rectangular regions that were then algorithmically subdivided into smaller quads. This is called the "macro-element" or "multi-block" approach, and was first described in [137]. The interested reader is referred to [69] for an overview of grid generation methods. The coordinate changes are characterized by the deviation of the local axes from the Cartesian axes, called α and β, respectively (see figure 4.1) [48]. When working with water velocity in the curvilinear system, we must distinguish between the *curvilinear* (or *covariant*) velocity, whose components are parallel with the local axes, and the *contravariant* velocity, whose components are perpendicular to their complementary axes (see figure 4.1). Covariant velocity, denoted by $u_{\mathrm{CL}}, v_{\mathrm{CL}}$ is used for the momentum balance equations (4.1) and (4.2), while contravariant velocity, denoted by $u_{\mathrm{CV}}, v_{\mathrm{CV}}$ is used for the mass balance equation (4.3).

A summary of the derivations in [48] is presented here. In order to develop expressions for the curvilinear and contravariant velocity in terms of the Cartesian velocity components u and v, it is convenient to begin by expressing the velocity in polar coordinates R, θ with respect to the Cartesian axes. Expressions for the curvilinear velocity are derived by analyzing triangle $\triangle OCV$ with the Sine Rule, then solving for u_{CL} and v_{CL}:

$$\frac{R}{\sin(\frac{\pi}{2}+\alpha-\beta)} = \frac{u_{\mathrm{CL}}}{\sin(\frac{\pi}{2}-\theta+\beta)} = \frac{v_{\mathrm{CL}}}{\sin(\theta-\alpha)}$$
$$u_{\mathrm{CL}} = R\cos(\theta-\beta)\sec(\alpha-\beta)$$
$$v_{\mathrm{CL}} = R\sin(\theta-\alpha)\sec(\alpha-\beta)$$

The process for the contravariant velocity is the same, analyzing triangle $\triangle ODV$:

$$\frac{R}{\sin(\frac{\pi}{2}-\alpha+\beta)} = \frac{u_{\mathrm{CV}}}{\sin(\frac{\pi}{2}-\theta+\alpha)} = \frac{v_{\mathrm{CV}}}{\sin(\theta-\beta)}$$
$$u_{\mathrm{CV}} = R\cos(\theta-\alpha)\sec(\alpha-\beta)$$
$$v_{\mathrm{CV}} = R\sin(\theta-\beta)\sec(\alpha-\beta)$$

Using the sum and difference identities, we expand the $\{\cos,\sin\}(\theta \pm \{\alpha,\beta\})$ terms, and substitute $R\cos(\theta) = u$ and $R\sin(\theta) = v$. This leads to systems of equations which can be solved for the forward and inverse transforms between Cartesian velocity components and the two types of curvilinear components.

$$\begin{bmatrix} u_{\mathrm{CL}} \\ v_{\mathrm{CL}} \end{bmatrix} = \sec(\alpha-\beta) \begin{bmatrix} \cos\beta & \sin\beta \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\beta \\ \sin\alpha & \cos\beta \end{bmatrix} \begin{bmatrix} u_{\mathrm{CL}} \\ v_{\mathrm{CL}} \end{bmatrix}$$

$$\begin{bmatrix} u_{\mathrm{CV}} \\ v_{\mathrm{CV}} \end{bmatrix} = \sec(\alpha-\beta) \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\alpha \\ \sin\beta & \cos\alpha \end{bmatrix} \begin{bmatrix} u_{\mathrm{CV}} \\ v_{\mathrm{CV}} \end{bmatrix}$$

$$\begin{bmatrix} u_{\mathrm{CV}} \\ v_{\mathrm{CV}} \end{bmatrix} = \begin{bmatrix} \sec(\alpha-\beta) & \tan(\alpha-\beta) \\ \tan(\alpha-\beta) & \sec(\alpha-\beta) \end{bmatrix} \begin{bmatrix} u_{\mathrm{CL}} \\ v_{\mathrm{CL}} \end{bmatrix}$$

$$\begin{bmatrix} u_{\mathrm{CL}} \\ v_{\mathrm{CL}} \end{bmatrix} = \begin{bmatrix} \sec(\alpha-\beta) & -\tan(\alpha-\beta) \\ -\tan(\alpha-\beta) & \sec(\alpha-\beta) \end{bmatrix} \begin{bmatrix} u_{\mathrm{CV}} \\ v_{\mathrm{CV}} \end{bmatrix}$$

All other variables have trivial transformations, and we will abuse notation by not distinguishing them from their original forms.

The linearized shallow water equations (4.6), (4.7) and (4.8) are transformed into the curvilinear coordinates [48]:

$$\frac{\partial u_{\mathrm{CL}}}{\partial t} + U^0_{\mathrm{CL}}\frac{\partial u_{\mathrm{CL}}}{\partial x_{\mathrm{CL}}} + V^0_{\mathrm{CL}}\frac{\partial u_{\mathrm{CL}}}{\partial y_{\mathrm{CL}}} + \left(U^0_{\mathrm{CL}}\frac{\partial u_{\mathrm{CL}}}{\partial y_{\mathrm{CL}}} + V^0_{\mathrm{CL}}\frac{\partial u_{\mathrm{CL}}}{\partial x_{\mathrm{CL}}}\right)\sin(\alpha-\beta) = -g\frac{\partial h}{\partial x_{\mathrm{CL}}} - g\frac{\partial b}{\partial x_{\mathrm{CL}}} + Cu_{\mathrm{CL}} \quad (4.10)$$

$$\frac{\partial v_{\mathrm{CL}}}{\partial t} + U^0_{\mathrm{CL}}\frac{\partial v_{\mathrm{CL}}}{\partial x_{\mathrm{CL}}} + V^0_{\mathrm{CL}}\frac{\partial v_{\mathrm{CL}}}{\partial y_{\mathrm{CL}}} + \left(U^0_{\mathrm{CL}}\frac{\partial v_{\mathrm{CL}}}{\partial y_{\mathrm{CL}}} + V^0_{\mathrm{CL}}\frac{\partial v_{\mathrm{CL}}}{\partial x_{\mathrm{CL}}}\right)\sin(\alpha-\beta) = -g\frac{\partial h}{\partial y_{\mathrm{CL}}} - g\frac{\partial b}{\partial y_{\mathrm{CL}}} + Cv_{\mathrm{CL}} \quad (4.11)$$

$$\frac{\partial h}{\partial t} + U^0_{\mathrm{CV}}\frac{\partial h}{\partial x_{\mathrm{CL}}}\sec(\alpha-\beta) + V^0_{\mathrm{CV}}\frac{\partial h}{\partial y_{\mathrm{CL}}}\sec(\alpha-\beta) + H^0\left(\frac{\partial u_{\mathrm{CV}}}{\partial x_{\mathrm{CL}}} + \frac{\partial v_{\mathrm{CV}}}{\partial y_{\mathrm{CL}}}\right)\sec(\alpha-\beta) = 0 \quad (4.12)$$

These transformed equations are algebraically more involved, but from a practical perspective, simply add static trigonometric terms to the discretized scheme (4.16), (4.17), (4.18), to be derived next. They require that the velocity components be transformed back and forth between Cartesian and curvilinear axes. In particular, linearity is preserved.

4.2.2 Boundary conditions

For the boundary conditions, we imposed a condition that there be no velocity component perpendicular to the shoreline:

$$\vec{u} \cdot \vec{s}\big|_{\partial\Omega_{\text{land}}} = 0 \tag{4.13}$$

where $\vec{s} = \vec{s}(x, y)$ is a vector perpendicular to the shoreline, and $\partial\Omega$ is the boundary of the domain. No-slip conditions ($\vec{u}\big|_{\partial\Omega_{\text{land}}} = \vec{0}$) are also commonly used, but are inappropriate for a linear scheme, since shear forces arise from the non-linear terms in the original momentum equations (4.1), (4.2). This is a common boundary condition in the hydrodynamics literature, and sometimes called the "slip boundary condition"; see for example [48] and [55]. The implicit assumptions for the slip boundary condition are that the bathymetry is steep enough at the shore that the water height will not significantly affect the location of the land boundary, and that there is no significant influx or outflux of water from or into the soil. This would be an inappropriate assumption, for example, for a shallow mud flat that wets and dries during the tidal cycle [121, 129]; but for steep banked river channels such as the ones studied in this chapter, the assumption is justified.

This constraint is enforced on the curvilinear mesh by forcing the u_{CV} or v_{CV} component of the water velocity at specific nodes to zero.

The upstream velocity and downstream height boundary conditions are implicitly defined as being equal to the value at the initial condition:

$$\vec{u}(t)\big|_{\partial\Omega_{\text{upstream}}} = \vec{u}(0)\big|_{\partial\Omega_{\text{upstream}}} \tag{4.14}$$

$$h(t)\big|_{\partial\Omega_{\text{downstream}}} = h(0)\big|_{\partial\Omega_{\text{downstream}}} \tag{4.15}$$

This is an appropriate assumption for assimilation over short times compared to the tidal cycle.

4.2.3 Discretization

We use an implicit discretization scheme, consisting of backward Euler for the time derivative and centered differencing for the spatial derivatives, also known as the *Backwards Time Centered Space (BTCS) method* [54]. This scheme was chosen for computational efficiency, since it is an *implicit, single step* scheme. The number of decision variables in the optimization program is proportional to the number of time steps in the discretization, which will be seen in section 4.3.1. A multi-step scheme (where intermediate states are computed for each time step) would result in a similar proportional growth of decision variables. Implicit methods, named for their *implicit* time update equation, are not constrained by the *Courant-Friedrichs-Lewy* (CFL) stability condition on the time step. Using an explicit method would require satisfying the CFL condition, which would require more time steps, which would increase the number of decision variables in the variational assimilation problem. As will be

discussed in section (4.3.1), however, the method developed in this chapter is applicable to all single and multi-step schemes, explicit or implicit.

We use the covariant velocity variables. The mass conservation equation (4.12) uses contravariant velocity, not covariant velocity, which means an additional transform is necessary, as can be seen in (4.18).

$$\begin{aligned}\frac{u^{k+1}_{\mathrm{CL}i,j}-u^{k}_{\mathrm{CL}i,j}}{\Delta t} =& -U^{0}_{\mathrm{CL}i,j}\frac{u^{k+1}_{\mathrm{CL}i+1,j}-u^{k+1}_{\mathrm{CL}i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}} - V^{0}_{\mathrm{CL}i,j}\frac{u^{k+1}_{\mathrm{CL}i,j+1}-u^{k+1}_{\mathrm{CL}i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}}\\ &-\sin(\gamma_{i,j})U^{0}_{\mathrm{CL}i,j}\frac{u^{k+1}_{\mathrm{CL}i,j+1}-u^{k+1}_{\mathrm{CL}i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}} - \sin(\gamma_{i,j})V^{0}_{\mathrm{CL}i,j}\frac{u^{k+1}_{\mathrm{CL}i+1,j}-u^{k+1}_{\mathrm{CL}i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}}\\ &-g\frac{h^{k+1}_{i+1,j}-h^{k+1}_{i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}} - g\frac{b_{i+1,j}-b_{i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}} + C_{i,j}u^{k+1}_{\mathrm{CL}i,j}\end{aligned} \tag{4.16}$$

$$\begin{aligned}\frac{v^{k+1}_{\mathrm{CL}i,j}-v^{k}_{\mathrm{CL}i,j}}{\Delta t} =& -U^{0}_{\mathrm{CL}i,j}\frac{v^{k+1}_{\mathrm{CL}i+1,j}-v^{k+1}_{\mathrm{CL}i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}} - V^{0}_{\mathrm{CL}i,j}\frac{v^{k+1}_{\mathrm{CL}i,j+1}-v^{k+1}_{\mathrm{CL}i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}}\\ &-\sin(\gamma_{i,j})U^{0}_{\mathrm{CL}i,j}\frac{v^{k+1}_{\mathrm{CL}i,j+1}-v^{k+1}_{\mathrm{CL}i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}} - \sin(\gamma_{i,j})V^{0}_{\mathrm{CL}i,j}\frac{v^{k+1}_{\mathrm{CL}i+1,j}-v^{k+1}_{\mathrm{CL}i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}}\\ &-g\frac{h^{k+1}_{i,j+1}-h^{k+1}_{i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}} - g\frac{b_{i,j+1}-b_{i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}} + C_{i,j}v^{k+1}_{\mathrm{CL}i,j}\end{aligned} \tag{4.17}$$

$$\begin{aligned}\frac{h^{k+1}_{i,j}-h^{k}_{i,j}}{\Delta t} =& -\sec(\gamma_{i,j})\sec(\gamma_{i,j})U^{0}_{\mathrm{CL}i,j}\frac{h^{k+1}_{i+1,j}-h^{k+1}_{i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}} - \sec(\gamma_{i,j})\tan(\gamma_{i,j})V^{0}_{\mathrm{CL}i,j}\frac{h^{k+1}_{i+1,j}-h^{k+1}_{i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}}\\ &-\sec(\gamma_{i,j})\tan(\gamma_{i,j})U^{0}_{\mathrm{CL}i,j}\frac{h^{k+1}_{i,j+1}-h^{k+1}_{i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}} - \sec(\gamma_{i,j})\sec(\gamma_{i,j})V^{0}_{\mathrm{CL}i,j}\frac{h^{k+1}_{i,j+1}-h^{k+1}_{i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}}\\ &-\sec(\gamma_{i,j})\sec(\gamma_{i,j})H^{0}_{i,j}\frac{u^{k+1}_{\mathrm{CL}i+1,j}-u^{k+1}_{\mathrm{CL}i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}} - \sec(\gamma_{i,j})\tan(\gamma_{i,j})H^{0}_{i,j}\frac{u^{k+1}_{\mathrm{CL}i+1,j}-u^{k+1}_{\mathrm{CL}i-1,j}}{\Delta x_{i-1,j}+\Delta x_{i,j}}\\ &-\sec(\gamma_{i,j})\tan(\gamma_{i,j})H^{0}_{i,j}\frac{u^{k+1}_{\mathrm{CL}i,j+1}-u^{k+1}_{\mathrm{CL}i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}} - \sec(\gamma_{i,j})\sec(\gamma_{i,j})H^{0}_{i,j}\frac{v^{k+1}_{\mathrm{CL}i,j+1}-v^{k+1}_{\mathrm{CL}i,j-1}}{\Delta y_{i,j-1}+\Delta y_{i,j}}\end{aligned} \tag{4.18}$$

where the subscript indexes i and j are for the x and y grid directions, respectively; the superscript index k is the time index; Δt is the time step; $\Delta x_{i,j}$ is the distance between node (i,j) and $(i+1,j)$; and $\Delta y_{i,j}$ is the distance between node (i,j) and $(i,j+1)$. $\gamma_{i,j}$ is an abbreviation for $\alpha_{i,j}-\beta_{i,j}$.

4.3 Quadratic Programming

4.3.1 Variational framework

Our method uses *variational data assimilation*. The variables in the discretized equations (4.16), (4.17), and (4.18) are concatenated into vectors, using the standardized framework set out in [57], as follows:

X_n Concatenated vector of state variables (u, v, h) for all mesh points at time t_n.

X_B Background term vector to improve well-posedness of the problem.

Y_n Vector of observed variables at time t_n. No observations are taken at time 0.

B Covariance matrix of the background error (the vector difference between the initial state X_0 and the background term X_B).

R_n Covariance matrix of the observation error at time t_n.

H_n Observation operator, which projects the state vector X_n into the observation subspace containing Y_n.

The method is referred to as *variational* data assimilation because the method estimates the optimum initial condition functions $u(x, y, 0)$, $v(x, y, 0)$, and $h(x, y, 0)$. The objective function expressed in (4.19) is a function of the discretized state, but in the original formulation the objective is actually a *functional* of the initial conditions. This terminology serves to distinguish the method from the broad category of *sequential* methods, which typically estimate the state at the observation times [57]. It can be argued that, because of the discretization, the objective is no longer a functional and that the method is no longer variational. The terminology is useful, however, for placing our method in context with the data assimilation literature.

Our data assimilation strategy is to search for the initial state X_0 that minimizes a weighted ℓ^2 norm of the difference between the state and observation variables and the difference between the initial state and the background term X_B:

$$\mathcal{J}^0(X_0) = (X_0 - X_B)^T B^{-1}(X_0 - X_B) + \sum_{n=1}^{n_{\max}} (Y_n - H_n[X_n])^T R_n^{-1}(Y_n - H_n[X_n]) \qquad (4.19)$$

X_B, the background term, is a "first guess" about the state of the system; it is expected that it is inaccurate, and that the product of the assimilation will be a refined estimate of the state. The background term could be derived from historical data, from forecasts, from a previous assimilation, or from forward simulation based on boundary conditions (either observed or artificially generated). The need for a background term is discussed further in section 4.3.3.

The covariance matrices B and R_n affect the weight given to the background term and the observations. Choosing appropriate values for these covariances is discussed in section 4.3.3. As a simplifying assumption we take these matrices to be a scalar times the identity matrix: $b\mathbb{I}$ and $r\mathbb{I}$, respectively.

General variational data assimilation schemes treat the observation operator H_n as a non-linear operator. One feature of Lagrangian sensing using a positioning technique like GPS is that observations come with both location and velocity information. Our assimilation method is *a posteriori*, so our knowledge of the observation positions can be used to represent the observation operator as a time-varying matrix. In the context of estimation on linear systems, H_n would be called the *observation matrix* [108]. In the simplest case, where the assimilation time steps match the observation times, the H_n matrix would be a $\{0, 1\}$ matrix, with element $(i, j) = 1$ if the drifter associated with measurement i was in the cell associated with state variable j at time n. If drifter measurements are not synchronized with assimilation steps, then the values in the H_n matrix should reflect the polynomial approximation associated with the time discretization scheme. For example, for a single step method such as the backward Euler scheme, a drifter observation would be mapped into two H_n matrices using linear interpolation. This mapping can be generalized to any linear multi-step method.

The search space for the variational data assimilation is the initial condition of the solution to the linearized, discrete PDE; by the implicit definition of the boundary conditions, we are at the same time searching for the upstream velocity and downstream height boundary conditions. Appropriate choices for B and R_n mean that the cost function can be represented as a positive semidefinite quadratic term. The discretized dynamics of the flow are represented as a series of linear constraints of the form

$$EX_{n+1} = FX_n + g \tag{4.20}$$

where E, F are matrices determined by the time and space difference schemes (4.16), (4.17) and (4.18), and g is a vector capturing source terms that do not depend on the state, such as the bottom elevation. This constraint formulation allows implicit discretization schemes to be implemented, which broadens the applicability of the method significantly. Explicit schemes have a conceptually simpler process update (the E matrix is usually the identity matrix), but the time step used in these schemes is restricted by the CFL condition, and can often be inconveniently short.

The E matrix is invertible, so it would be possible to solve (4.20) for X_{n+1} and achieve a classical discrete-time dynamical system. Numerical issues make this step inadvisable. E and F are sparse matrices; E^{-1} would be non-sparse, and the scale of the matrices would introduce significant numerical error through non-sparse matrix multiplication.

4.3.2 Quadratic program

With a positive semi-definite quadratic cost function and linear constraints, the data assimilation problem can be posed as a QP problem

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned} \tag{4.21}$$

The variables in bold are from standard optimization formulations [15] and should not be confused with the variables used in the rest of this chapter. In particular, note that $\mathbf{x}$ is the vertical concatenation of all state vectors $X_0 \dots X_{n_{\max}}$, $\mathbf{P}$ and $\mathbf{q}$ are found by expanding all the terms in (4.19) and combining into a single quadratic expression, as shown here:

$$\mathbf{P} = \begin{bmatrix} B^{-1} & & & & \\ & H_1^T R_1^{-1} H_1 & & & \\ & & H_2^T R_2^{-1} H_2 & & \\ & & & \ddots & \\ & & & & H_{n_{\max}}^T R_{n_{\max}}^{-1} H_{n_{\max}} \end{bmatrix} \tag{4.22a}$$

$$\mathbf{q} = \begin{bmatrix} -X_B^T B^{-1} & Y_1^T R_1^{-1} H_1 & Y_2^T R_2^{-1} H_2 & \cdots & Y_{n_{\max}}^T R_{n_{\max}}^{-1} H_{n_{\max}} \end{bmatrix} \tag{4.22b}$$

The equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ represents the flow dynamic constraints described in (4.20), and can be expanded as:

$$\begin{bmatrix} -F & E & & & \\ & -F & E & & \\ & & \ddots & \ddots & \\ & & & -F & E \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{n_{\max}} \end{bmatrix} = \begin{bmatrix} g \\ g \\ \vdots \\ g \end{bmatrix} \tag{4.23}$$

$\mathbf{G}$ and $\mathbf{h}$ are normally zero, although we may impose heuristic inequality constraints to reduce the search space, in particular for initial and boundary conditions.

4.3.3 Background term and well-posedness of the problem

The Cauchy problem associated with a system of shallow water equations (4.1–4.3), for *subcritical* flow, with initial conditions provided everywhere, and boundary conditions (4.13–4.15), is *well posed* as defined by Hadamard [35]. This is a fairly standard result in the study of hyperbolic partial differential equations. The "subcriticality" of the flow is a condition that can formally defined, and has to do with the ratio of the velocity field and the eigenvalues of a matrix which appears when equations (4.1–4.3) are written in conservation law form. In physical terms, it means that the velocity of the water is always less than

the wave propagation speed. This is a reasonable assumption for our application. "Well-posedness" means that (a) a solution exists, (b) the solution is unique, and (c) the solution depends continuously on the initial and boundary conditions.We are interested in two of the analogous properties for the variational data assimilation problem (4.21); namely, that

1. there exists an optimal $\mathbf{x}$,
2. the optimizing $\mathbf{x}$ is unique.

A minimizer to (4.21) exists because of the convexity of the objective function ($\mathbf{P}$ is positive semidefinite) and the bounded, non-empty constraint set.

Observations are usually sparse compared to the number of mesh points; the H_i matrices are rank-deficient, and therefore the matrix $\mathbf{P}$ is positive semi-definite as opposed to positive definite. Thus, the uniqueness of an optimum $\mathbf{x}$ is not guaranteed; this is a common problem in oceanography and similar fields relying on data assimilation. However, we can argue that by incorporating the background term X_B, we restrict $\mathbf{x}$ to a set of smaller dimension than it would be otherwise.

Assume for argument that $\mathbf{x}^*$ is a feasible, optimal solution to the QP (4.21), and that $\frac{1}{2}\mathbf{x}^{*T}\mathbf{P}\mathbf{x}^* + \mathbf{q}^T\mathbf{x}^* = j^*$. For simplicity, we will drop the $\mathbf{G}\mathbf{x} \leq \mathbf{h}$ constraint. To investigate the uniqueness of $\mathbf{x}^*$, we consider the related feasibility problem

$$\begin{aligned} \tfrac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} &= j^* \\ \mathbf{A}\mathbf{x} &= \mathbf{b} \end{aligned} \tag{4.24}$$

We already have one feasible solution, so we consider $\mathbf{x} = \mathbf{x}^* + \mathbf{x}'$:

$$\tfrac{1}{2}\mathbf{x}'^T\mathbf{P}\mathbf{x}' + \left(\mathbf{x}^{*T}\mathbf{P} + \mathbf{q}^T\right)\mathbf{x}' = 0 \tag{4.25a}$$

$$\mathbf{A}\mathbf{x}' = 0 \tag{4.25b}$$

$\mathbf{x}^*$ is the *unique* optimizing solution for (4.21) if and only if (4.25a,4.25b) admit a single solution in $\mathbf{x}'$, the zero vector.

The uniqueness of the solution to (4.21) then becomes a geometric question: whether the intersection of the quadratic hypersurface defined by (4.25a) and the linear subspace defined by (4.25b) contains just a single point (the zero vector) or multiple points; and if so, how those multiple points can be characterized. Since $\mathbf{P}$ is positive *semi*definite, the quadratic hypersurface defined by (4.25a) is degenerate, and so we might expect the set of solutions to include linear subspaces; however, the block structure of $\mathbf{P}$ and $\mathbf{A}$, when a background term is used, preclude this possibility.

Assume for contradiction that $\mathbf{x}'$ is a non-zero member of a linear subspace that satisfies (4.25a,4.25b); in other words, $\lambda\mathbf{x}'$ also satisfies (4.25a,4.25b) for any scalar λ. Because (4.25a) is the sum of a linear and quadratic term, it is then necessarily true that $\mathbf{x}'^T\mathbf{P}\mathbf{x}' = 0$. Recalling the block structure of $\mathbf{P}$ from (4.22a),

$$\mathbf{x}'^T\mathbf{P}\mathbf{x}' = X_0'^T B^{-1} X_0' + \sum_{i=1}^{n_{\max}} X_i'^T H_i^T R_i^{-1} H_i X_i'$$

B^{-1} is positive definite, unlike all the other blocks of $\mathbf{P}$, so $X'_0 = 0$. But the block structure of $\mathbf{A}$, shown in (4.23), means that if $X'_0 = 0$, all the other X'_i vectors must be zero as well. This contradicts our assumption that $\mathbf{x}'$ is non-zero; and so the feasibility problem (4.25a,4.25b) does not admit any non-trivial linear subspaces.

Unfortunately, the "true" uniqueness of $\mathbf{x}^*$ is not guaranteed. (4.25a,4.25b) still admit solutions where $\mathbf{x}'^T\mathbf{P}\mathbf{x}' \neq 0$. But these solutions are not linear subspaces; for any non-zero $\mathbf{x}'$ satisfying (4.25a,4.25b), the only scalar values of λ for which $\lambda\mathbf{x}'$ also satisfies (4.25a,4.25b) are 0 and 1.

If we did not use a background term for our data assimilation problem, $\mathbf{P}$ would have no strictly positive definite block, and the contradiction proof above would not work. The set of optimal solutions to (4.21) could include linear subspaces as well as nondegenerate quadratic hypersurfaces. By adding a background term, we exclude all linear subspaces from the set of optimal solutions.

4.3.4 Choice of the covariance matrices

The covariance matrices B and R_n affect the weight given to the background term and the observations. In the absence of second-order statistics, they can be approximations representing the assumed reliability of the different sources of information. As discussed above, we have assumed that $B = b\mathbb{I}$ and $R_i = r\mathbb{I}$ for simplicity. A reasonable choice for r can be made from the accuracy specifications of the GPS module. Choosing b is less clear. If b is too small, and the eigenvalues of B^{-1} become large, the assimilation process will overemphasize the background term, and the observations will not significantly affect the final estimate. If b is too large, the regularizing effect of the background term will be insufficient to improve the convergence of the algorithm.

There is also an important relationship between the choice of B and the accuracy of the X_B term. If we had a trusted estimate of the covariance of the error between the background term and the true state, then the correct choice of B is obvious; however, this is not practically useful, since the background term is almost always a "guess". Estimating the error of the background term, when the background term is generated from guessing, simulations, or historical data, is an extremely non-trivial task.

Clearly, it would be advantageous to be able to use "bad" background terms and still get accurate assimilation results. In many ways, the background term encodes assumptions about qualitative properties of a valid solution: for example, the overall direction of water movement, the relative uniformity of water height, the fact that the water velocity is locally roughly parallel, etc. If the background term violates these assumptions, a valid solution is extremely unlikely. For example, if we used a background term with water flowing the wrong direction, we would be asking the optimization algorithm to find a valid PDE solution with no external sources that somehow performed a complete flow reversal in the short time period between the initial time and the first observation. The result would depend on the weighting of the B and R matrices, but would probably be some intermediate value, inconsistent with

both the background and the observations. In this sense, we see that the background term must be *qualitatively* accurate. Establishing the requirements for *quantitative* accuracy, and the corresponding choice of B, is an open problem.

4.3.5 Convergence of QP solutions to valid PDE solutions

A feasible solution to the quadratic program (4.21) will, by construction, satisfy the finite difference equation system (4.16) – (4.18).

Hence, in the limit case in which the discretization size converges to zero, the solution of the discretized system (4.16) – (4.18) converges to the solution of the linearized PDE system (4.6) – (4.8). This is a known result of the Lax equivalence theorem, applied to the BTCS method, which is unconditionally stable [54].

The linearized PDE system (4.6) – (4.8) is an approximation of the original PDE system (4.1) – (4.3). These linearizations are standard in fluid mechanics, and obtained procedurally as follows. A nominal flow of the nonlinear PDE is constructed, around which a first order Taylor expansion of the PDE system is derived, which leads to the *linear* PDE system. Accordingly, the error between the two models is of order $O(u^2), O(v^2), O(h^2)$, as well as the neglected terms. Extending properties of linearized hyperbolic conservation PDE systems to their nonlinear counterparts has examples in the literature; for example, [28] showed that stability properties for a one-dimensional linearized shallow water system under boundary control can be extended to the nonlinear case. Extending similar properties for the estimation problem under the conditions of this chapter (arbitrary geometry, variable boundary conditions) is beyond the mathematical scope of this study. However, it is likely that specific results could be proven for simpler geometries and idealized conditions.

Our case for the applicability of the solutions derived from our assimilation method is thus a three-step argument: (1) the QP solution is an exact solution for the discretized system; (2) the solution to the discretized system converges to the solution to the linearized PDE as the step sizes go to zero; (3) the error in the linearized model is of quadratic order compared to the fully non-linear PDE model.

4.4 Field test

4.4.1 Available data

The proposed platform is designed to solve practical problems for which several types of information are available. The following is a list of the data sources used by the data assimilation method:

- *Drifters*: The Lagrangian sensors record their position with GPS as they advect through the water. They also record a GPS velocity signal, which we use directly (as opposed to deriving velocity from the successive positions using a finite difference

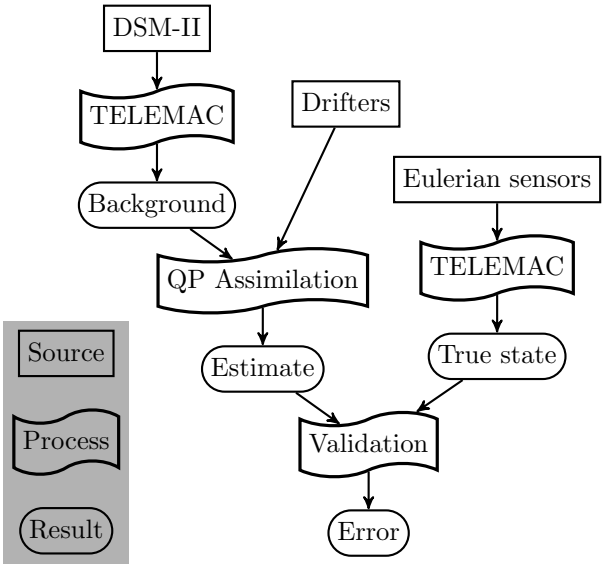

Figure 4.2: Data flow diagram used for the data assimilation using the hardware platform.

scheme). We built ten drifters; in the experiments presented here, up to eight were deployed at a time.

- *DSM-II Historical Data*: DSM-II [1] is a one dimensional model of the entire Sacramento/San Joaquin Delta. It was used to generate historical flow and height values for the background.

For validation purposes, we also gathered Eulerian data at the boundaries of the region of interest, using sensors described below. This data was used as the boundary conditions for a forward simulation using TELEMAC, a commercial hydrodynamics simulator. TELEMAC is essentially a specialized PDE solver for the shallow water equations; given the initial conditions and boundary conditions, it finds the velocity at all points in the mesh through a forward simulation of the equation. Since actual measurement of the initial conditions was unavailable, we used the standard technique of starting with an arbitrary initial condition, holding the boundary conditions steady, and running the simulation for a long time, essentially "washing away" the arbitrary initial condition. This technique is only appropriate for systems that are close to a steady state, which is a reasonable assumption for the slowly-changing river.

The Eulerian data includes the following items (see Figure 4.3):

- *Acoustic Doppler Current Profiler* (ADCP): This Eulerian sensor was installed by our group near the upstream boundary of the region of interest. It sits on the bottom of the river and measures the water velocity in the vertical column over it. This data allows estimation of the upstream flow boundary condition.

- *USGS Gauge Stations*: These Eulerian sensors measure flow and height. One sensor in the Sacramento River and one in the Georgiana Slough provide information about the downstream boundaries.

The list of data sources must also include the bathymetry and Manning parameters. The bathymetry is used in the QP assimilation (see equations (4.6), (4.7) and (4.8)). The TELEMAC forward simulations that generate the background term and validation data use both the bathymetry and the Manning parameters.

The data flow diagram in Figure 4.2 shows how the various data are used. Historical DSM-II data is used, with TELEMAC 2D [38] forward simulations, to generate the background term for the QP process. The estimate of the state of the system is generated by assimilating the drifter data. The Eulerian sensors are used with TELEMAC to generate a separate state estimate that is used for Eulerian validation.

4.4.2 Experimental strategy

Eight drifter deployments were performed from November 12 to November 16, 2007, at the junction of the Georgiana Slough and Sacramento River in California. This location was chosen for the USGS field gauges which could be used for Eulerian validation.

For each experiment, between seven and ten drifters were placed in the water by personnel in a small motorcraft. The initial positions were in a roughly straight line across the river, with approximately even spacing, but in the center of the river to avoid obstacles and shallow areas on the sides. Figure 4.4 depicts an example of the drop points used in experiment 4 on November 16. The drifters were monitored as they travelled in the river. Each experiment was planned to last between 45 and 60 minutes; in practice, some of the experiments were terminated earlier. Reasons for terminating the experiment included (*i*) drifters travelling past the junction, eliminating line of sight, (*ii*) drifters spacing out too far, making them difficult to monitor, (*iii*) miscellaneous logistical concerns.

With the development of short-range and long-range wireless communication capabilities on the drifters, the hardware infrastructure is designed to let the drifters operate autonomously, without direct line of sight supervision, allowing for experiments with expanded domains in space and time.

Figure 4.5 shows the water velocity at the ADCP versus time over the five day experimental period. The start times of the eight experiments are shown with "x" marks. The velocity time series was processed with a low-pass filter (zero-phase, cutoff frequency 7.85×10^{-5}Hz, corresponding to a period of 3.54 hour, generated by the Parks-McClellan optimum filter algorithm [58]). To better show the length of the experiments, and their relationship to the tidal cycle, the filtered velocity signal for all five days was superimposed in figure 4.6, referenced to the minor maximum of the velocity.

Post-experiment analysis showed that several drifters did not record sufficient amounts of GPS data (e.g. less than 10% of the expected amount of data); in most cases this was

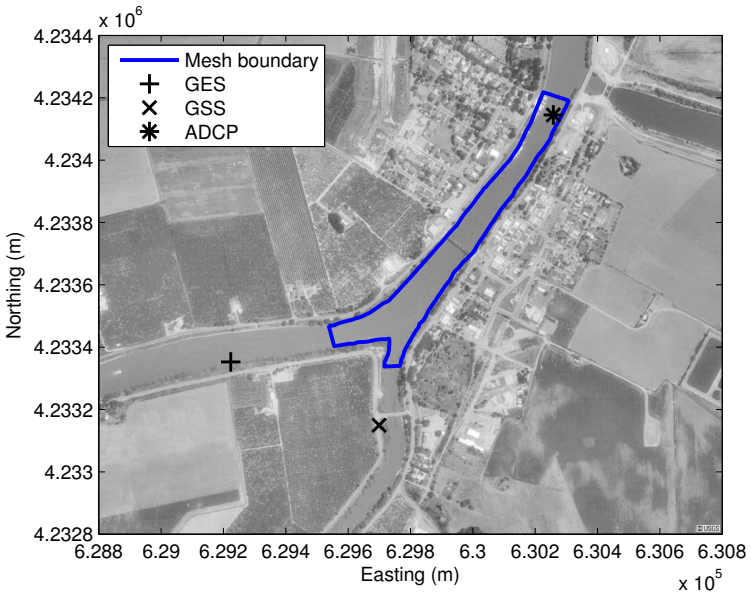

Figure 4.3: Sacramento River/Georgiana Slough with modelled area, ground stations, and sensor deployment locations. Image courtesy of USGS.

traced to antenna connection problems. This reduced the number of operating drifters at a given time to between five and eight. Only four of the eight experiments had enough data to proceed with the assimilation method.

4.4.3 Implementation of the algorithm

The drifter measurements were sampled at 30s. Each drifter measurement was assigned according to its GPS location to a specific cell of the curvilinear mesh, and the GPS velocity was converted to curvilinear coordinates. The DSM-II historical data was then used to generate boundary conditions for a TELEMAC forward simulation to generate the background term. A QP problem was formulated using the drifter measurements and the background term for the cost function, and the curvilinear, discretized, linearized PDE equations as linear constraints, as described in Section 4.2. The drifters do not gather information about the water height. The friction source term was set to zero. The QP problem was expressed using the optimization modeling language AMPL and solved with CPLEX. The optimal initial condition was extracted from the CPLEX solution, and the curvilinear velocity field was converted back to the Cartesian grid.

One feature of the QP formulation is that the number of sensors can vary with time, simply by adding or removing the necessary terms from the cost function (4.19). This is advantageous, because in practice there are often gaps in the GPS tracks of the drifters (as

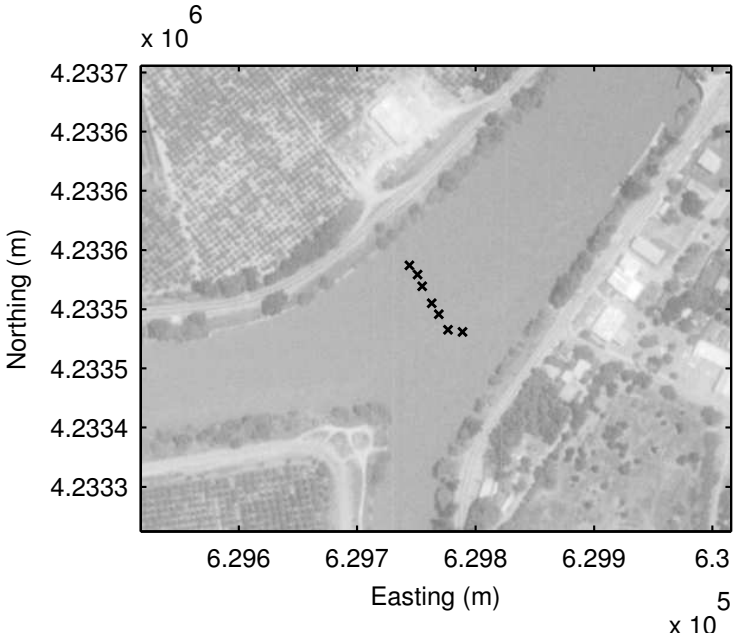

Figure 4.4: Example of drop points for drifter release in the final experiment.

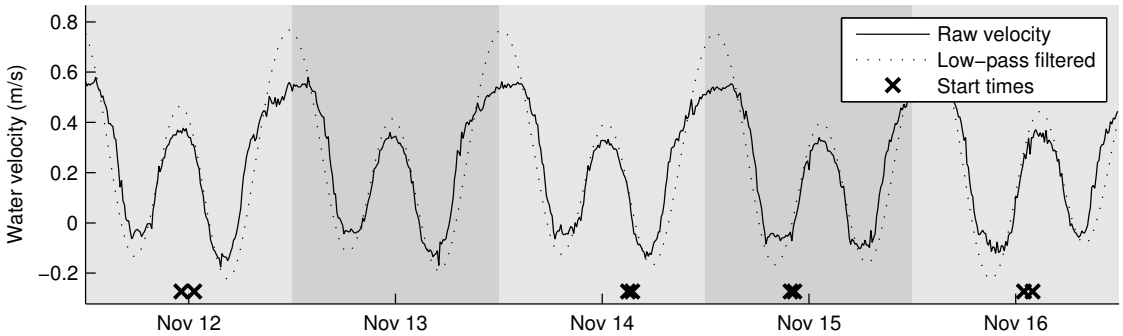

Figure 4.5: Overview of experimental timeframe.

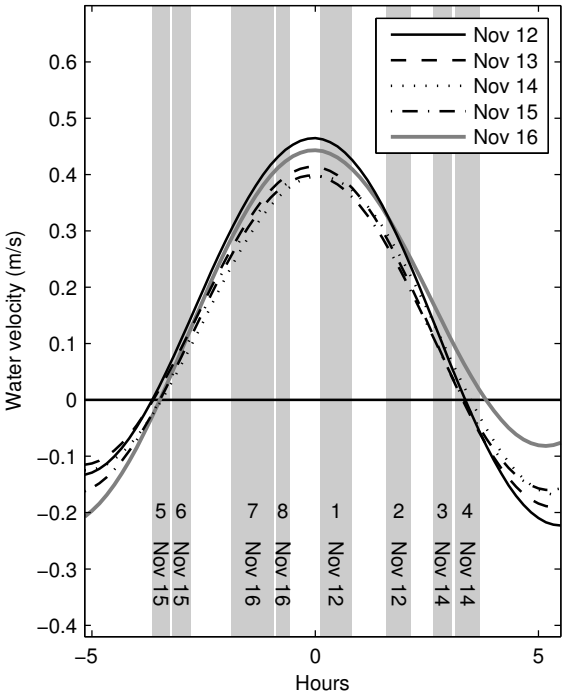

Figure 4.6: Experiment times shown relative to minor velocity peak.

they pass underneath bridges, or experience similar signal loss). Instead of trying to patch the holes in the record with a form of interpolation, the data can be passed as-is to the QP assimilation process.

4.4.4 Validation

A forward simulation of the region of interest was performed using the data from Eulerian sensors. This data was used as the boundary conditions for a SWE simulation, to generate what we will call the "true state" velocity field. This forward simulation *does* include the river bed friction term. The relative error between the true state, (u_T, v_T), and the estimated initial condition velocity field from the QP process, (u, v), was computed by dividing the ℓ^2 norm of the difference by the magnitude of the simulated field:

$$\epsilon_E = \sqrt{\frac{\sum_j (u_{Tj} - u_j)^2 + \sum_j (v_{Tj} - v_j)^2}{\sum_j u_{Tj}^2 + \sum_j v_{Tj}^2}} \tag{4.26}$$

where j is the node index.

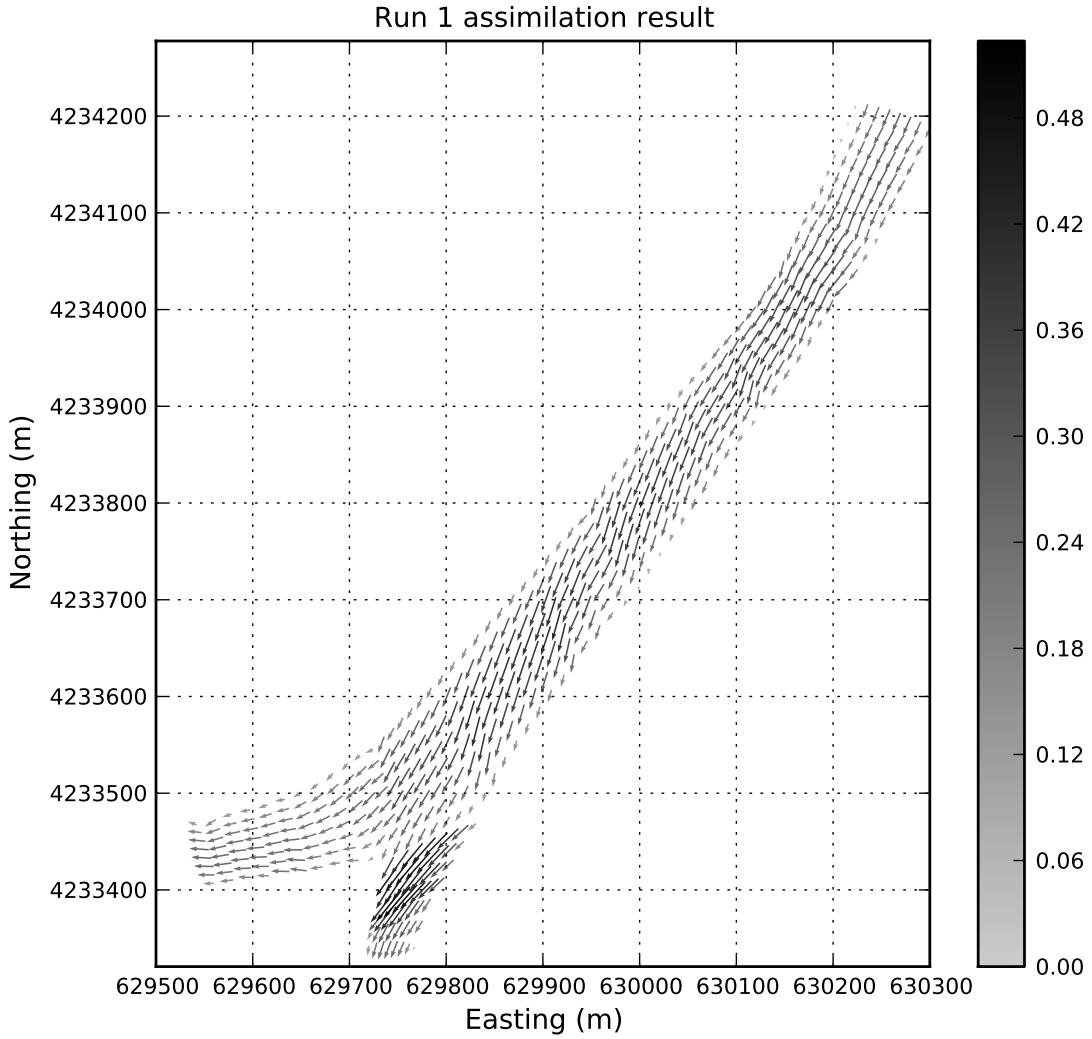

Figure 4.7: Assimilated velocity field initial condition for experiment 1. Scale is in m/s. Decimated by two for legibility.

4.4.5 Results

Figure 4.7 shows the initial flow field condition assimilated by the QP algorithm for one of the experiments. The height variable is very smooth (differing by only a few centimeters over the region) and not interesting to plot. Only one experiment is shown for space constraints. Figure 4.8 shows how the QP assimilated velocity field is closer to the true state (generated by forward simulation from Eulerian sensors) than the background term.

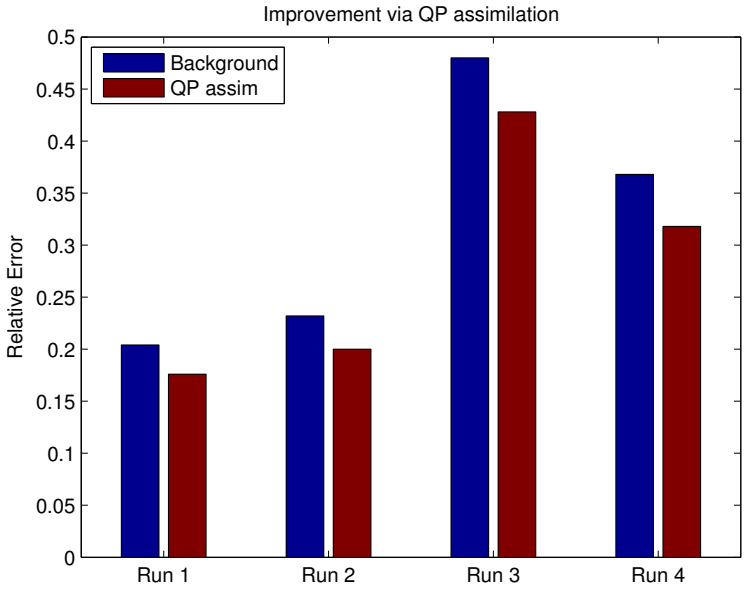

Figure 4.8: Change in relative error for each of four experiments.

Using a 2000-point mesh (16 cells across the river, 118 cells down the reach of the Sacramento segment), performing a QP assimilation of 30 minutes worth of drifter data (from 5–8 drifters) takes approximately 10 minutes on a single 2.0GHz processor.

4.5 Conclusions

In this chapter, we present a method for formulating the variational data assimilation problem for Lagrangian sensors in shallow water flows as a quadratic programming optimization problem with linear constraints. A major advantage of the quadratic programming formulation is that the constraints can express the model partial differential equation discretized with an implicit scheme. This allows our method to use longer time steps than explicit methods. Our method also assimilates on the initial conditions, in contrast to many sequential methods.

The quadratic programming assimilation method relies on a background term, as many variational data assimilation methods do, both to guarantee well-posedness and to provide a "first guess" to the system. The metric used to evaluate the assimilation performance is the improvement made in relative error versus a true state. Care was taken to ensure that the true state used distinct information; the assimilation process relied on historical data (for the background term) and Lagrangian sensor data, while the true state was simulated from local Eulerian sensors. (Both sides use the same bathymetry and Manning parameter data,

but this is not a major issue).

The flow field estimates generated by the data assimilation process described in this chapter may be useful in their own right, as an estimate of the hydrodynamic state of the river environment; they are also useful for planning the navigation of *actuated* Lagrangian sensors through the river. In Chapter 5 we will explore an optimal control technique for maintaining safe positions in rivers that uses the flow field estimates generated by data assimilation techniques like the QP method described here.

Chapter 5

Single-Vehicle Safety Control

5.1 Introduction

In order to plan the movements of actuated mobile sensors, the effect of the flowing water on the vehicle position must be taken into account. With data assimilation techniques like the one presented in Chapter 4, we can estimate the flow field that will affect our floating sensors. In this chapter, we present a control framework for single-vehicle navigation in flow environments to deal with navigation objectives and obstacles.

Early experiments with purely passive drifters demonstrated that unattended floating objects like drifters will frequently get caught on the banks of river channels or in vegetation near the banks. Figure 5.1 shows a Gen 3 drifter caught in reeds during a field experiment. There are two contributing factors to the problem of getting caught in obstacles. First, the size of the drifter with respect to vegetation and the depth of the channel. If the fluid flow carries the drifter to a point in the channel that is shallower than its draft, or a region with vegetation like that shown in Figure 5.1 where fluid can move but larger objects cannot, then the drifter will become stuck. Second, the fluid at the surface of the channel is often subject to forcing terms that represent confounding factors to the "natural" river flow. For example, a fast-moving boat will create a substantial wake that will sweep floating objects to the side of the channel. Recreational boat traffic

Figure 5.1: A Gen 3 drifter with a flag caught in reeds. Photo credit: Jean-Benoît Saint-Pierre.

in the Sacramento-San Joaquin Delta often complicate drifter operations.

Since the goal of a Lagrangian drifter is to behave like an ideal particle in the fluid flow, getting caught on an obstacle represents non-ideal behavior. One way of mitigating the obstacle problem is therefore to make the drifter closer to an ideal particle. The smaller the drifter, the less susceptible it will be to getting caught in shallow regions or in vegetation. If the drifter were neutrally buoyant, and the the drifter fleet were distributed vertically as well as horizontally, surface effects like boat wakes would not be significant. The tradeoffs involved in the design of the drifters were explored in depth in Chapter 3. A neutrally buoyant drifter approximately 1 cm large would not be challenged by the obstacles present in the Sacramento-San Joaquin Delta; however, this drifter design is not currently within our capabilities. Instead, we use active propulsion to mitigate the problem of obstacles.

Navigational objectives also call for active propulsion and autonomous control. In the case of a branching network of river channels, we may want to ensure that a certain number of drifters occupy each channel. There are many ways in which the assignment of drifters to channels could be performed; once the assignment has been performed, however, the drifters must have some way of controlling their trajectory to ensure that they remain inside the channel to which they have been assigned.

The drifters' active propulsion is limited by the energy capacity of their batteries. In addition, the use of propulsion is also constrained by the mission objective of gathering Lagrangian data. The fundamental assumption of Lagrangian sensing is that the drifter velocity matches that of the fluid in which it is immersed. This is clearly not true during active propulsion. Although it might be possible to separate the propulsion term from the underlying flow in order to recover the fluid velocity during propelled movement, this would be a very difficult system identification and estimation problem; we choose to simply discard the velocity data gathered by a drifter when any power is being applied to the propellers. We therefore want to minimize the control effort expended, not just in terms of the energy expended, but in the absolute sense of time during which any propulsion is happening at all. We chose a bang-bang controller, in which either the control effort is either zero or the maximum possible, in order to keep more sensor data.

Our approach is to use a reachability framework to calculate *minimum-time-to-reach* (MTTR) functions for two different scenarios. In the first scenario, we assume that the drifter is not using its propulsion, and we find the shortest time before the drifter reaches an obstacle like the shoreline. In the second scenario, we assume that the drifter is applying maximum propulsion, and we find the shortest time before the propelled drifter reaches the center of the river. "Shoreline" and "center" are regions defined *a priori* by annotating a satellite map of the experimental region.

We use the *Hamilton-Jacobi-Bellman-Isaacs* (HJBI) PDEs [62, 41, 7] to construct MTTR maps under the two scenarios. Isotime contours of these maps are then used to define state transition thresholds for the bang-bang controller. The gradient of the second MTTR function (active propulsion) is also used to generate the optimal heading for travel to the center of the river.

In this chapter, we describe the systems of fluid flow and disturbance used to model the drifter's movement for the HJBI functions. We then illustrate the generation of the MTTR functions from the HJBI equations. We describe the validation of the method through *Software-in-the-Loop* (SIL) testing as well as two field experiments in the Sacramento-San Joaquin Delta with Gen 3 drifter vehicles.

5.2 Reachability Techniques for Path Planning

Reachability frameworks study the feasible evolutions of a finite-dimensional state space system for bounded timescales. They are a good choice for studying motion problems for vehicles in flow fields, for two reasons: many reachability frameworks are general enough in terms of system dynamics to admit the natural flow dynamics of the river, and by studying the finite-time evolution of the feasible trajectories, they can verify whether safety constraints like obstacle avoidance can be satisfied.

One of the key mechanisms of reachability analysis is the Hamilton-Jacobi PDE and its variants. In this chapter, we use the HJBI equation in order to model external disturbances as an "oppositional player" in a differential game. The solution of a HJBI equation can be used to construct a *minimum-time-to-reach* (MTTR) function, expressing the shortest time in which a system can reach a target state or set of states while obeying defined system dynamics. One major challenge to implementing reachability techniques based on Hamilton-Jacobi equations is the so-called *curse of dimensionality*: the computational effort necessary to solve such equations scales exponentially with the number of dimensions of the system state. The curse of dimensionality was originally recognized by Bellman in the context of *Dynamic Programming* (DP) algorithms [12]; the Hamilton-Jacobi approach to reachability can be interpreted as the continuous limit of DP [7], and subject to the same limitations. Strategies for bypassing, mitigating, or enduring the curse of dimensionality are the focus of significant research attention. In this work, we handle the computational challenge by keeping the system dimension low (only two dimensions), and by pre-computing the MTTR functions for use as feedback maps by the drifters.

Path planning problems are well studied in the literature [65, 66]; the most common features of path planning problems are the need to find a trajectory for a vehicle that avoids known obstacles and reaches a desired final state. Non-trivial external dynamics, like a flowing river, requires the specialized handling of reachability tools; reachability for path planning is also known in the literature [113]. The problem studied in this chapter has two features that distinguish it:

- The river current, in general, is stronger than the propulsion of the vehicle; in other words, the velocity of the water is greater than the maximum still-water velocity of the vehicle over significant subsets of the environment. Planning trajectories must, therefore, take the external dynamics into account. The path-planning problem may be absolutely infeasible for some starting configurations; it may be impossible for

the vehicle to reach a target destination before being swept into an obstacle by the movement of the water.

- In general, the target that we are trying to reach is not a single point, but rather a subset of the system domain.

Algorithmically, several approaches can be taken to find the MTTR functions that form the foundation of the path-planning strategy. *Viability theory* [19, 99] approaches the reachability problem using set-valued functions to model the control options and external disturbances on the system; finding the *viability kernel* of a problem (the set of points guaranteed to be safe) has been shown to be equivalent to solving the MTTR for reaching a target set [46]. Path planning problems can therefore be addressed by reformulating it as a viability problem [118].

Another algorithmic approach, and the one applied in this chapter, is to compute the level sets of the HJBI solution [9, 74, 101, 83, 84]. An existing mathematical toolbox [73] was used to solve these equations numerically.

In previous work [128], the Floating Sensor Network team described the HJBI algorithm and how it solves the path-planning problem for vehicles in a flow field. In this chapter, the work is extended with greater detail in the simulation and experimental verification of the implemented algorithm.

5.3 Hamilton-Jacobi-Bellman-Isaacs Based Optimal Control

5.3.1 Model and Problem Statement

For this problem, we model the dynamics of a single drifter vehicle as a particle in 2D Euclidean space, capable of limited actuation in any direction, and subject to an external flow field:

$$\begin{aligned} &\dot{x} = w(x) + a(t) + b(t), \\ &\|a(t)\|_2 \leq \bar{a}, \\ &\|b(t)\|_2 \leq \bar{b}, \end{aligned} \tag{5.1}$$

where $x \in \mathbb{R}^2$ is the two-dimensional state vector representing the position of the drifter in meters, $w : \mathbb{R}^2 \to \mathbb{R}^2$ is the external flow field in meters per second, $\bar{a}$ is the limit of the actuation input in meters per second, and $\bar{b}$ is the limit of an unknown disturbance input in meters per second. In this chapter, $\bar{a} = 0.2$ m/s, $\bar{b} = 0.05$ m/s, and the magnitude of $w(x)$ is typically around 0.8 m/s. Note that $w(x)$ does not vary with time. In the context of a tidally-influenced river, this means we are assuming that the path planning problem is

of sufficiently short duration in time (30 minutes, for example), that the changing tide does not perturb the problem unduly.

Modelling the vehicle as a 2D problem is a choice motivated by the need to keep the number of dimensions small in order for the HJBI computation to be tractable. It is a reasonable assumption, because the cylindrical form and differential drive configuration of the drifter make it highly manoeuverable about its vertical axis; it can, effectively, apply propulsion in arbitrary directions at will, at least on the time scales that matter for this path planning problem.

We define the sets of functions that satisfy the magnitude constraints as $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{A} \triangleq \left\{a(\cdot)\colon \forall t\ \|a(t)\|_2 \leq \bar{a}\right\},$$
$$\mathbf{B} \triangleq \left\{b(\cdot)\colon \forall t\ \|b(t)\|_2 \leq \bar{b}\right\},$$

and parametrize the system trajectory in terms of time, initial condition, and the control/disturbance inputs:

$$x = x(t; x_0, a(\cdot), b(\cdot)).$$

As part of the problem definition, we have a set of undesirable positions, $D \subset \mathbb{R}^2$, representing the obstacles that must be avoided. We assume that these obstacles are all known and do not change with time. The complement of the obstacle set, $S \triangleq D^C$, gives the positions for which the system is *safe*.

The path planning problem is to find a control input $a(\cdot)$ such that

$$\forall t > 0,\ \forall x_0 \in D^C,\ \forall b(\cdot) \in \mathbf{B},\ x(t; x_0, a(\cdot), b(\cdot)) \in D^C, \tag{5.2}$$

which minimizes the time of actuation,

$$t_{\text{act}} = \int_0^\infty \mathbb{1}_{\left[a(t)\neq 0\right]} dt.$$

We will not find a formal optimum of this problem; instead we will apply a bang-bang controller, based on the intuition that the controller should either be completely off or at maximum effort in order to keep the actuation time small.

The problem as formulated is a *differential game* [62, 74] in which the inputs $a(\cdot)$ and $b(\cdot)$ work against each other to either satisfy or attempt to violate (5.2), respectively. We will show later that $b(\cdot)$ will always act in the opposite direction of $a(\cdot)$ with magnitude $\bar{b}$. Under the bang-bang controller heuristic, the overall differential game with input constraints $(\bar{a}, \bar{b})$ can be split into two games: one with the control input zero, with input constraints $(0, \bar{b})$, and one with the control input at its maximum value, with input constraints $(\bar{a} - \bar{b}, 0)$. This is a useful decomposition, but it does not hold for general differential games [62].

5.3.2 Mathematical Solutions

This section reviews the construction of MTTR functions by using level set techniques to solve the HJBI equation.

Given a target set $\mathcal{T} \subset \mathbb{R}^n$ of states we are trying to reach, subject to the dynamics described in (5.1), we can define a static cost function $V(x_0)$:

$$V(x_0) = \inf_{a(\cdot)\in\mathbf{A}} \sup_{b(\cdot)\in\mathbf{B}} \left\{ \int_0^{t^\star} l\left(x(t; x_0, a(\cdot), b(\cdot)), a(t), b(t)\right) \mathrm{d}t \right\}, \tag{5.3}$$

$$t^\star(x_0, a(\cdot), b(\cdot)) = \inf\left\{ t\colon x(t; x_0, a(\cdot), b(\cdot)) \in \mathcal{T} \right\}, \tag{5.4}$$

where $l(x, a, b)$ is a Lagrangian cost function defining the cost per unit time of being in state x while taking actions (a, b). The $t^\star$ function defines the first time that the trajectory of the system enters $\mathcal{T}$ given a starting state x_0 and the control/disturbance inputs $a(\cdot)$ and $b(\cdot)$.

If we are interested in minimizing the time to reach $\mathcal{T}$, we can define $l(\cdot, \cdot, \cdot) \equiv 1$. Equation (5.3) then becomes

$$\begin{aligned} V(x_0) &= \inf_{a(\cdot)\in\mathbf{A}} \sup_{b(\cdot)\in\mathbf{B}} \left\{ \int_0^{t^\star} 1 \cdot \mathrm{d}t \right\} \\ &= \inf_{a(\cdot)\in\mathbf{A}} \sup_{b(\cdot)\in\mathbf{B}} t^\star(x_0, a(\cdot), b(\cdot)), \end{aligned} \tag{5.5}$$

or, more concisely

$$V(x_0) = t^\star\left(x_0, a^\star(\cdot), b^\star(\cdot)\right), \tag{5.6}$$

$$\left(a^\star(\cdot), b^\star(\cdot)\right) = \left(\arg\inf_{a(\cdot)\in\mathbf{A}}, \arg\sup_{b(\cdot)\in\mathbf{B}} \right) t^\star(x_0, a(\cdot), b(\cdot)) \tag{5.7}$$

The notation above is used for convenience to denote the solution of the minmax game defined in (5.3). Under this definition, $V(x_0)$ is the MTTR function for the target set. We call $a^\star(\cdot)$ the *optimal control* and $b^\star(\cdot)$ the *worst case disturbance*.

If the target set cannot be reached from a particular initial condition, $V(x_0) = +\infty$ for that value of x_0.

The optimal control and worst case disturbance can be found directly from the gradient of V:

$$a^\star(x) = -\bar{a}\frac{\nabla V(x)}{\|\nabla V(x)\|_2}, \qquad b^\star(x) = \bar{b}\frac{\nabla V(x)}{\|\nabla V(x)\|_2}. \tag{5.8}$$

In order to compute V, we have extended the technique found in [73] for finding the MTTR function of a holonomic system: a *time-dependent HJBI* equation, which is solved

using level set methods [83]. The HJBI function is defined as [74]:

$$0 = \phi_t + \min\left[0, \bar{G}(x, \nabla\phi)\right], \qquad t \in (0, h), \tag{5.9}$$

$$\bar{G}(x,p) \triangleq \max_{\|a\|_2 \le \bar{a}} \min_{\|b\|_2 \le \bar{b}} \left\{ p^T \cdot f(x,a,b) \right\}.$$

$$\begin{cases} \phi(x,0) < 0 & x \in \text{int}(\mathcal{T}) \\ \phi(x,0) = 0 & x \in \partial\mathcal{T} \\ \phi(x,0) > 0 & \text{otherwise} \end{cases}, \tag{5.10}$$

where $f(x,a,b) = w(x) + a + b$ is the evolution of x as defined in (5.1). $\phi(x,0)$ is a function whose *zero level set* matches the boundary of $\mathcal{T}$; while it is not precisely specified by (5.10), a typical choice is a signed distance function to the boundary. Solving (5.9) for ϕ_t allows numeric solutions of $\phi(x,t)$ for $t \ge 0$. The zero level set of ϕ will move outward from $\partial\mathcal{T}$, sweeping out the set of points that can reach $\mathcal{T}$. These zero level sets define V:

$$\left\{x : V(x) = h\right\} = \left\{x : \phi(x,h) = 0\right\}$$

If we have a starting position x that is never swept by the zero level set of ϕ, that is, $\nexists h : \phi(x,h) = 0$, then $V(x) = +\infty$ and the target set is unreachable from x.

5.4 Implementation and Simulation

5.4.1 Flow Field Modeling

Generating the MTTR functions requires knowledge of $w(x)$, the flow field over the domain of interest. Building an estimate of the actual flow field is a non-trivial problem, and is one of the major goals of the Floating Sensor Network project. A future version of the system might use experimentally derived estimates of $w(x)$ to drive the control strategy of the vehicles. In the current implementation, we create a small number of simulated flow fields, representing major tidal regimes (high tide, low tide, ebb tide, slack tide, etc.) These flow field estimates were generated by REALM, a forward simulation model of the Sacramento-San Joaquin Delta [3]. As discussed previously, we assume that the "snapshot" estimates are reasonably well matched to the actual flow field, and that the flow field does not change significantly during individual path planning problems. The MTTRs for each tidal regime are loaded onto the vehicle's on-board computer, and the real-time controller chooses the appropriate MTTR set for the time of day.

Part of this work includes a simulation of the drifter/river system using a simple kinematic model of the drifter, using viscous friction and random disturbance forces. The simulated river in this model has a flow field derived from the REALM simulation system as well.

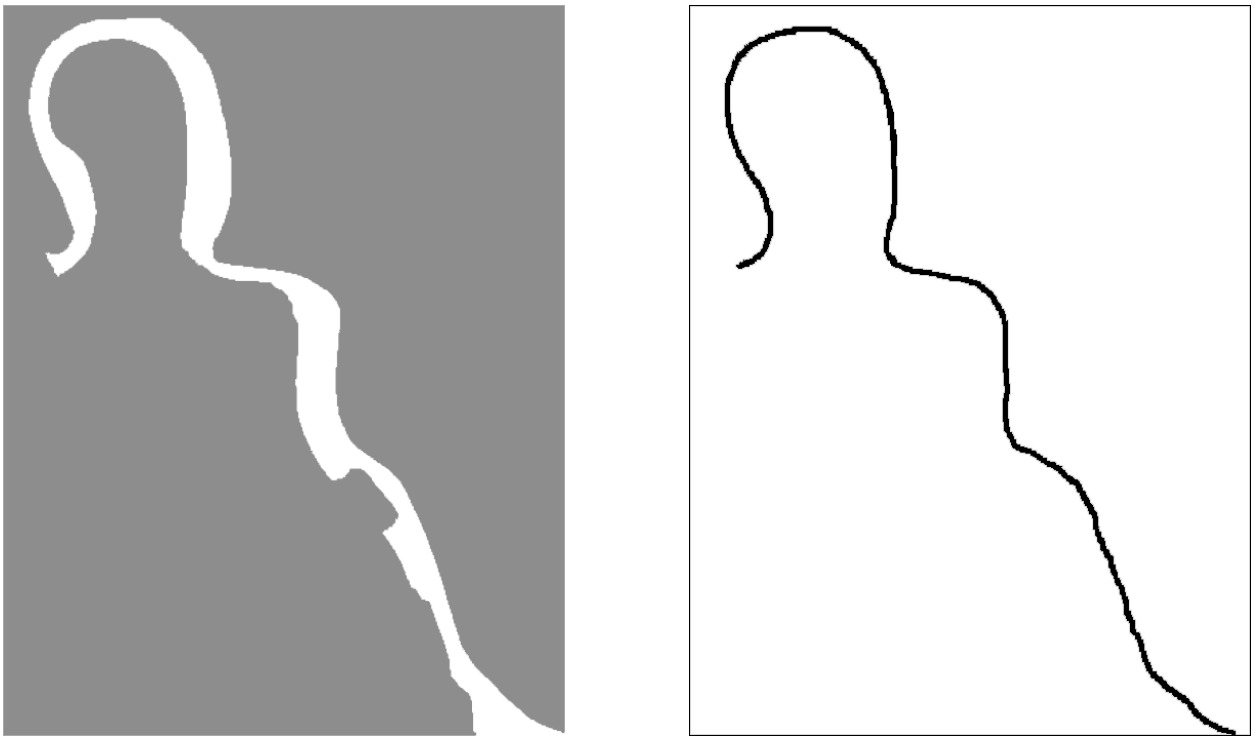

Figure 5.2: Left: $\mathcal{T}_{\text{shore}}$, the constraint set which we want the drifter to avoid. Right: $\mathcal{T}_{\text{center}}$, target set which the drifter needs to reach after touching the unsafe set.

5.4.2 Computation of Control Feedback

The real-time controller on board the drifter vehicle generates an actuation plan based on three pre-computed 2D lookup tables:

1. An MTTR function named V_{shore}, shown on the left of Figure 5.3, capturing the minimum time required for the drifter to be pushed into an obstacle by the combined influence of the river flow field and the worst-case disturbance, assuming that there is no actuation. We calculate this MTTR by setting $\mathcal{T}$, the target set, to D, the set of undesirable positions; and using the level set toolbox as outlined in Section 5.3.2, with the control authority magnitude set to $\bar{b}$ and the disturbance magnitude set to 0. In other words, we reverse the positions in the differential game; we try to find the "best" disturbance, that is, the one that will drive the unactuated drifter into the obstacle as quickly as possible, and compute the MTTR based on that scenario.

2. An MTTR function named V_{center}, shown on right of Figure 5.3, capturing the minimum time required to drive to the center of the river. The target set $\mathcal{T}$ is a user-defined narrow band in the middle of the river channel. The MTTR is found by setting the control authority magnitude and disturbance magnitude to their conventional values

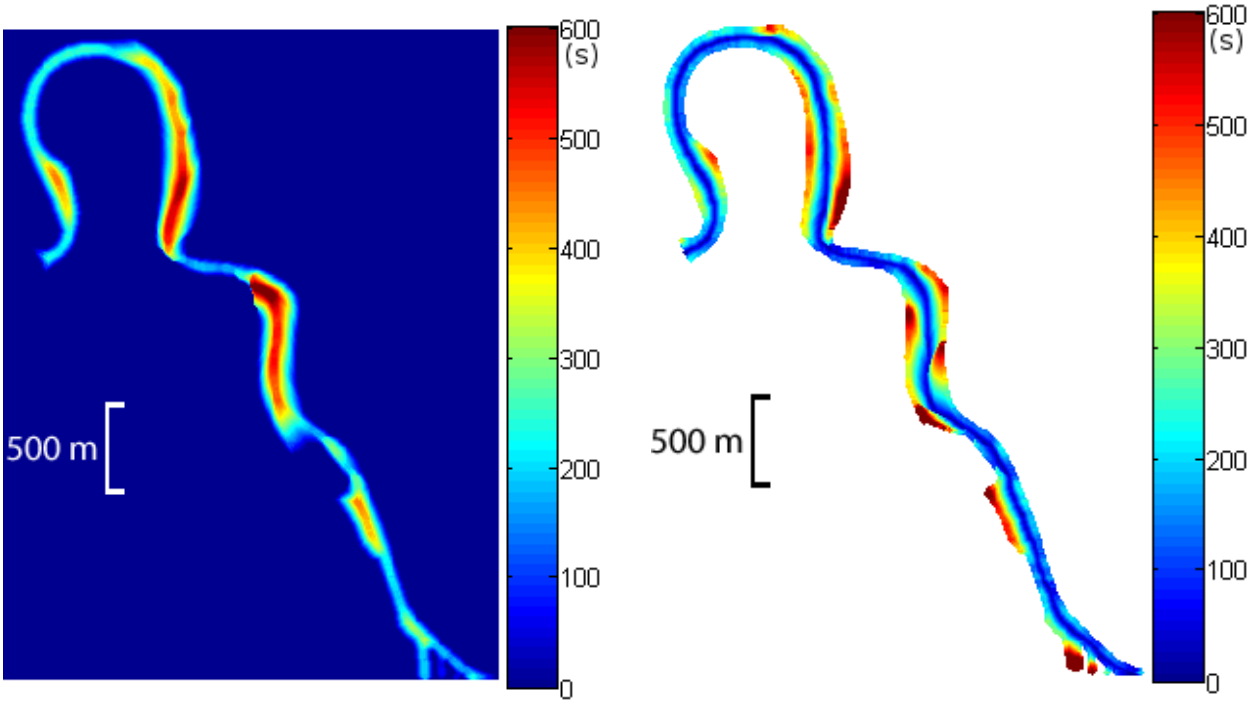

Figure 5.3: MTTR functions for the targets defined in Figure 5.2. Left: V_{shore} function. Right: V_{center} function.

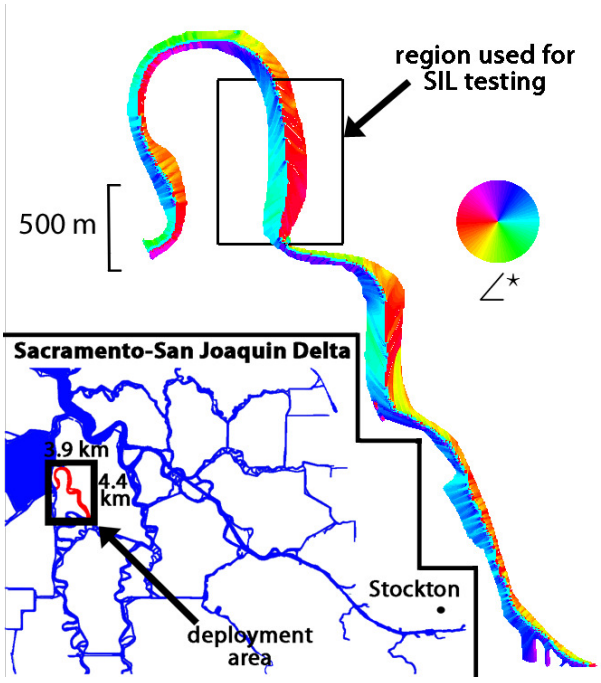

Figure 5.4: Optimal bearing output $\angle^\star$, represented with colors.

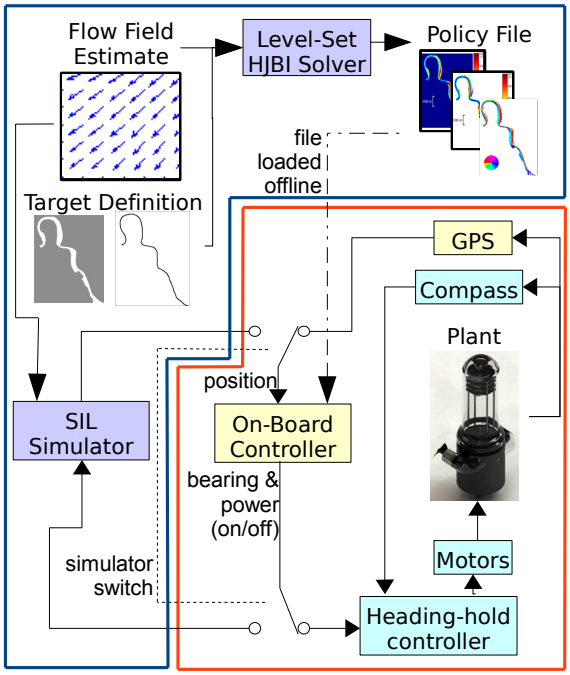

Figure 5.5: Controller implementation diagram – Orange box indicates online operations; Blue box indicates offline processes.

($\bar{a}$ and $\bar{b}$, respectively).

3. The optimum control heading toward the center of the river, named $\angle^\star(x)$. This is derived from V_{center} using the relation defined in (5.8). This function is shown in Figure 5.4.

Together, these three lookup tables form a *policy file* which is loaded onto the drifter before a field experiment. Each tidal regime (i.e. each estimated $w(x)$) requires its own policy file. We will also use different policy files to execute different navigational goals, as described in the next section.

5.4.3 Path selection

Figure 5.6 shows a section of the Sacramento River where it joins with a channel called the Georgiana Slough. The normal (outgoing) flow of the Sacramento River is from north to south. At the junction, the outgoing flow splits, with some of the water heading south down the Georgiana Slough. At high tide, the flow of water on the Sacramento often reverses; but usually, the Georgiana Slough does not reverse, and continues to pull water south from the junction. This junction suggests an experimental scenario with a navigational goal. If active drifters are deployed in the Sacramento River upstream of the junction during outgoing flow,

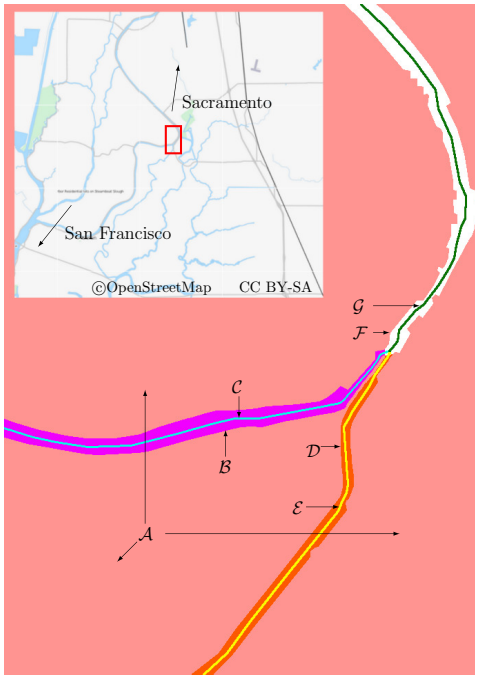

Figure 5.6: Source files for lane-splitting algorithm superimposed and color-coded; inset map showing geographic location. Each color represents a set of points and is labeled for reference.

we wish to control the drifter's actuation so as to guarantee that some drifters go down the Sacramento River and some go down the Georgiana Slough. We set aside the issue of how this assignment is to be made; in this chapter, we focus on the control technique necessary to ensure that a drifter can select one path or the other.

For each tidal regime, we calculated two policy files for the region shown in Figure 5.6. One policy file is for drifters which will go down the Sacramento River channel, and is generated with the inputs

$$\begin{aligned} \mathcal{T}_{\text{shore}} &\leftarrow \mathcal{A} \cup \mathcal{D} \cup \mathcal{E} \\ \mathcal{T}_{\text{center}} &\leftarrow \mathcal{G} \cup \mathcal{C} \end{aligned}$$

while the other policy file is for drifters which should go down the Georgiana Slough, and is generated with the inputs

$$\begin{aligned} \mathcal{T}_{\text{shore}} &\leftarrow \mathcal{A} \cup \mathcal{B} \cup \mathcal{C} \\ \mathcal{T}_{\text{center}} &\leftarrow \mathcal{G} \cup \mathcal{E} \end{aligned}$$

In each case, the "undesirable" channel is treated as an obstacle by adding it to $\mathcal{T}_{\text{shore}}$.

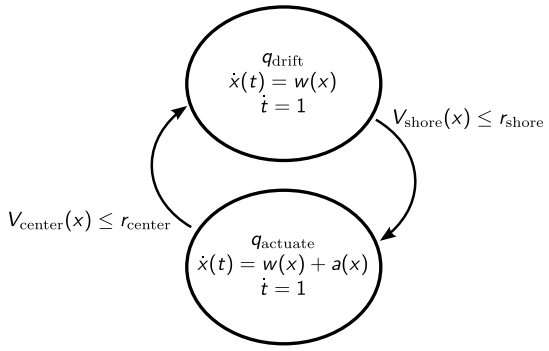

Figure 5.7: On-board controller hybrid automaton diagram.

5.4.4 On-board Controller

Our bang-bang controller switches between an "actuated" state and a "drifting" state. We can represent this controller as a hybrid automaton $H = (Q, X, R, f, \Sigma, \mathcal{U})$, where Q is the set of discrete modes, X is the domain of continuous states, $R : (Q, \Sigma, X) \mapsto (Q, X)$ is the transition function, $f_q : X \mapsto X$ are the continuous dynamics for each mode, Σ is the set of discrete events, and $\mathcal{U}$ is the set of continuous inputs [71, 23]. The resulting automaton definition is:

- $Q = \{q_{\text{drift}}, q_{\text{actuate}}\}$
- $X = \mathbb{R}^2$
- R is as documented in Figure 5.7
- f_q are shown in Figure 5.7
- $\Sigma = \{x \in \mathcal{T}, x \in \mathcal{D}\}$
- $\mathcal{U} = \mathbf{A} = \left\{a(\cdot) \colon \forall t \ \|a(t)\|_2 \leq \bar{a}\right\}$

The system transitions from one mode to the other based on defined thresholds of the MTTR functions. When in the q_{drift} mode, the V_{shore} function is monitored; if it falls below a critical value, the system switches to the q_{actuate} mode to drive away from danger. When sufficiently close to the center of the river, described by a threshold value on V_{center}, the system switches back to q_{drift} mode.

5.4.5 Simulated Results

We used a *Software-In-the-Loop* (SIL) simulation to explore the behavior of the drifters under the proposed control scheme in a simulated river with a REALM-derived flow field. The software environment is particularly well suited to exploring cases in which the "true"

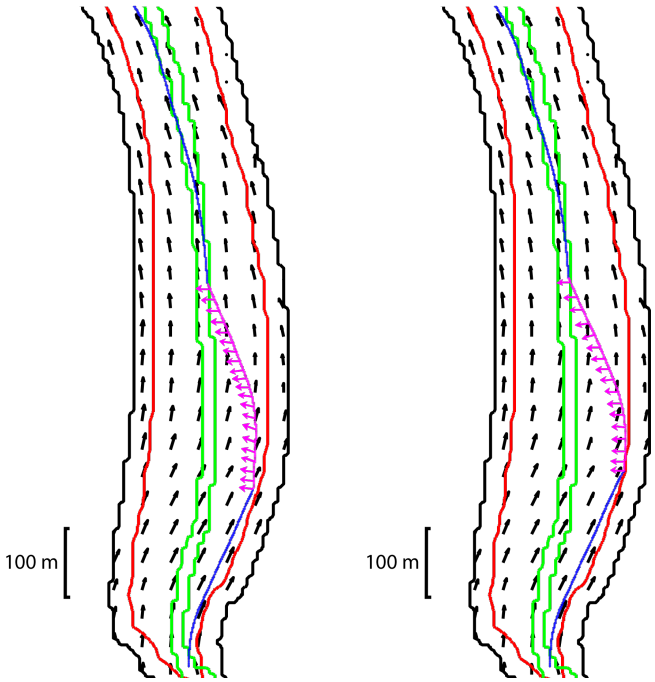

Figure 5.8: Simulated trajectories with presence or absence of REALM flow field estimate. Left: known estimate. Right: no estimate incorporated; zero flow field used instead.

river dynamics do not match the $w(x)$ function used to calculate the MTTR functions. This could easily be the case when working in new hydrodynamic systems which have not yet been well studied.

In the simulations described in this section, we also introduced a viscous force on the drifter toward the east. One possible source of such an input disturbance in the real environment would be the action of the wind. It was necessary to add such a perturbation to the simulated environment in order to force the drifter into dangerous regions, so that the actuation phase of the control scheme could be demonstrated.

Figure 5.8 illustrates two trajectories of the simulated drifter. The left is the ideal situation in which REALM provides a reasonably accurate flow field estimate. In the event that we know REALM will be inaccurate for a region, a compromise is to not use any flow field. This case is shown on the right, where the river flow is set to 0 at all points. These experiments demonstrate that the scheme is reasonably robust against inaccuracies in the MTTR calculation, particularly when the desired actuation is nearly orthogonal to the true driving flow field.

5.5 Field Operational Tests

5.5.1 Obstacle Avoidance

The feedback maps shown in Figure 5.3 were developed for a river environment in the Sacramento-San Joaquin Delta, near Franks Tract, south of the San Joaquin (approximately Latitude 38.03 N, Longitude 121.58 W. See Figure 5.4). Nine active drifters were deployed in the channel for approximately five hours. Their GPS traces show the action of the controller in keeping the drifters away from the shores. Two boat teams were responsible for monitoring the drifters and retrieving trapped units if necessary. Retrieved drifters were placed back in the river at safe locations to continue their mission. The primary goal of the experiment was to determine if the proposed controller could effectively prevent the drifters from heading into dangerous areas (the obstacle avoidance goal).

Figure 5.10 shows data from the field deployment that was gathered by one of the units. It is plotted in a similar fashion to the simulated results presented in Section 5.4.5. Flow field arrows are not displayed, as the true river flow during the experiment is unknown.

The drifter behavior is qualitatively identical to the behavior demonstrated in simulation. As in the simulation, the drifters in the field experienced an easterly wind that pushed them toward the shore. Figure 5.10 shows this drifter floating north with the river current, but also being pushed toward the eastern shoreline. Upon crossing the V_{shore} threshold (red contour), it changes to actuation mode and begins to navigate toward the center of the river. Once it reaches the V_{center} threshold (green contour), it transitions back to drifting without actuation. This is the behavior we expect given the simulator results. Small discrepancies are likely due to inaccuracies in flow magnitude in the REALM flow field used during simulation.

The field test validated the simulator results and demonstrated the fundamental effectiveness and applicability of the HJBI-based controllers. The off-line computation of the MTTR functions for use by a relatively simple lookup-driven on-line hybrid control proved to be a successful implementation of the control technique.

5.5.2 Path Selection

Our second experiment was at the junction of the Georgiana Slough and Sacramento River described in Section 5.4.3. This is also the site of the full-fleet operation described in Chapter 8. Our goal for this experiment was to divert 10 out of 30 actuated drifters down the Georgiana Slough, with the remaining 20 actively remaining in the Sacramento River. As well as this path selection task, we wished to also maintain a safe separation from the shoreline as well as some man-made structures (marinas and docks) in the region. We exaggerated the effect of the path selection mechanism by manipulating the obstacle map, $\mathcal{T}_{\text{shore}}$. Instead of simply blocking off the undesired channels, we formed two lanes down the Sacramento River upstream of the junction, so that the drifters would split into two groups travelling in parallel lines prior to the junction. This allowed us to diagnose any

Figure 5.9: Left: drifter under motor power. Right: several drifters participating in experiment.

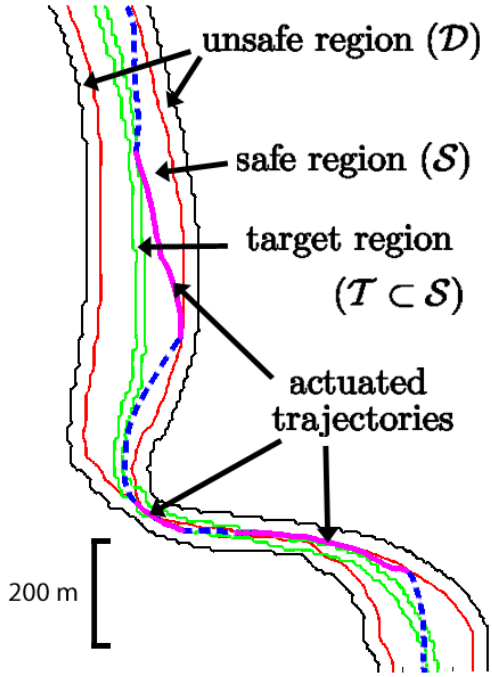

Figure 5.10: Drifter GPS Trajectory during northward tidal flow. The red line is a contour of V_{shore} and denotes the edge of the unsafe region. The green line is a contour of V_{center} and define the target region. Along the drifter trajectory, dotted lines indicate unactuated motion.

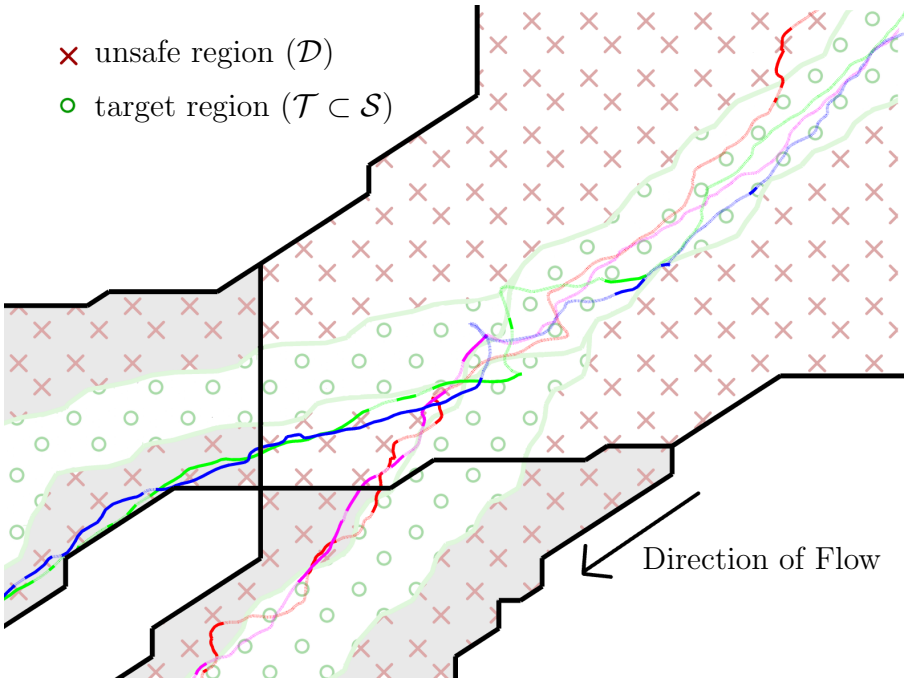

Figure 5.11: GPS Trajectories of four drifters performing the path-selection algorithm. Two of the drifters, green and blue, are tasked with proceeding down the Sacramento River, while the other two, red and magenta, are tasked with proceeding down the Georgiana Slough. Along the trajectory, the faded segments indicate unactuated motion.

malfunctioning drifters (ones that could not perform the split operation) and retrieve them before they reached the junction, where they would be in danger if their actuation was not working. In practice and in simulation, the algorithm does not require that virtual lanes be formed, only that an obstacle be drawn across unselected paths.

Figure 5.11 shows the trajectories of four drifters, two from each group, focusing on the point where the drifters start to separate into separate groups. The data represents a GPS location taken every 2 seconds by each drifter and passed through a two element moving average filter to reduce the apparent effect of GPS error. The trajectories shown in this figure demonstrate the correct behavior: the drifters are avoiding the obstacles, and separating into the desired channels.

5.6 Conclusions

This chapter has described a technique for controlling individual drifters to avoid obstacles and the shoreline, as well as enabling navigational goals like path selection. Solving the Hamilton-Jacobi-Bellman-Isaacs equations offline to construct feedback maps for on-line lookup is a successful way to mitigate the high computational cost of HJBI based reachabil-

ity techniques. We have demonstrated the applicability of the algorithm both in simulation and in several field tests. This implemented algorithm is a compelling way to autonomously control a lightly actuated sensor vehicle in a hazardous environment; this method greatly improves the feasibility of unattended inland Lagrangian sensing over purely passive solutions.

Chapter 6

Zermelo-Voronoi Partitions for Fleet Control

This chapter presents another approach to the control problem for the fleet of actuated sensor vehicles in a flow field. The *Zermelo-Voronoi partition* divides a spatial domain into zones of responsibility for individual agents in a flow field, based on the shortest time-to-reach from the agents' initial positions. Existing methods for computing the Zermelo-Voronoi partition either rely on restrictive conditions on the flow field or on general reachability algorithms requiring substantial computation. Under a less restrictive condition, namely *affine* fields, an efficient approximation algorithm is developed and demonstrated to compute the Zermelo-Voronoi partition; the properties of *centroidal* Voronoi control are shown to generalize to *pseudo-centroidal* control in the flow field case. The results are implemented using an algorithm also derived in the chapter.

6.1 Introduction

6.1.1 Voronoi Partitions

The Voronoi partition problem is an important building block for many multivehicle control schemes [5, 29, 36, 6]. A domain (usually space) is partitioned between vehicles so that each vehicle is assigned to the points to which it is *closest*. (See Section 6.2.3 for a formal definition of the Voronoi partition). A Voronoi partition is defined in relation to the metric used to evaluate distance; most examples in the literature use the Euclidean metric. Control strategies based on the Voronoi partition are usually premised on individual vehicles being given responsibility for the space in their Voronoi cell; examples include equalizing the area assigned to each vehicle or minimizing a cost function reflecting the minimum time-to-reach for the closest vehicle [29, 36]. However, the Euclidean Voronoi partition is not a useful construction when the vehicles are influenced by a flow field. In this case, the notion

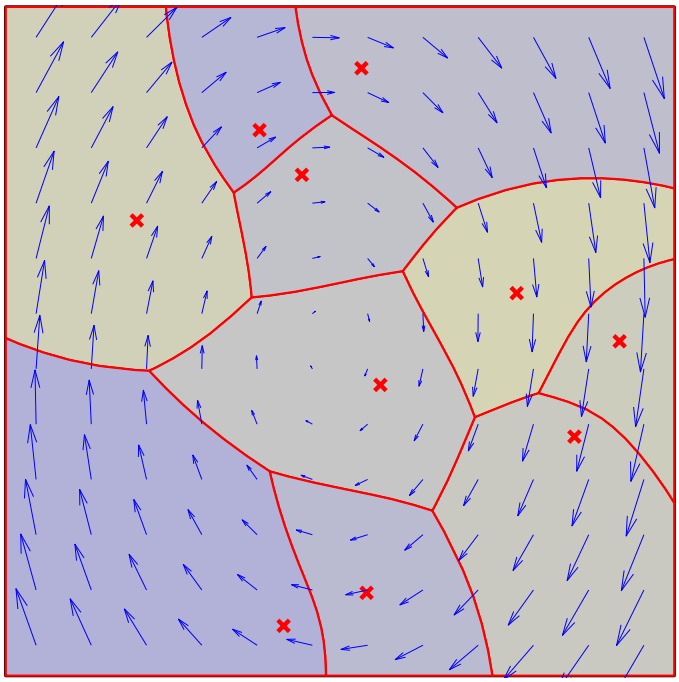

Figure 6.1: A Zermelo-Voronoi partition for vehicles (red crosses) in a flow field (blue arrows), showing the equal-time-to-reach boundaries (red lines). This partition was generated using the approximated optimal trajectory techniques described in Section 6.4.

of "distance" should reflect the asymmetric costs associated with moving upstream versus moving downstream in a flow field [6].

A *quasimetric* [107] is a function $d(x_1, x_2)$ which satisfies the usual metric properties *except* symmetry; that is, it is not generally true that $d(x_1, x_2) = d(x_2, x_1)$.

The *Zermelo quasimetric* captures the minimum time to travel between two points in a flow field. It is named after the optimal control problem posed in 1931 by Ernst Zermelo [103]. The primary contributions of the present chapter are the extension of results for the Euclidean Voronoi partition in control applications to the Zermelo-Voronoi partition, as well as an effective and efficient method for computing the Zermelo-Voronoi partition in constrained computation platforms, allowing these control strategies to be used for vehicles in flow fields.

6.1.2 Related Work

Sugihara [110] developed Voronoi partitions for vehicles in constant flow fields. Voronoi partitions in constant flow fields have been further developed by Bakolas and Tsiotras [6], who also coined the term "Zermelo-Voronoi diagram" to refer to the use of the Zermelo quasimetric. Constant-flow cases have also been studied in [64] and [98]. Nishida and

Sugihara also studied Voronoi partitions in *general* flow fields [78], using a level-set function approach to the reachability problem.

For Voronoi partitions in the conventional, zero-flow-field context, Cortés *et al.* [29] generalized the Voronoi partition in order to make it suitable for range-limited decentralized gradient descent schemes for coverage optimization. Du *et al.* [36] showed how *centroidal* Voronoi control, in which agents move toward the centroid of their Voronoi cells, solves a coverage optimization problem: minimizing the total time-to-reach over the domain.

6.1.3 Contrasting Approaches

One approach to constructing a Zermelo-Voronoi partition would be to solve the *Hamilton-Jacobi-Bellman* (HJB) partial differential equation (PDE) [74] in order to find a minimum-time-to-reach value function for each vehicle in the flow field. Given such value functions $M_i(x)$, the partition assignment function is simply

$$P(x) = \text{argmin}_i \, M_i(x)$$

This is equivalent to the approach taken in [78]. While this approach has the benefit of being general enough to handle any flow field, it has a number of disadvantages, particularly in the context of repeated Zermelo-Voronoi construction for gradient descent algorithms:

- Every time a vehicle position changes, the HJB value function must be recomputed.
- Each HJB computation is numerically intensive; decentralized, on-board computation might not be sufficient. [4, 74]
- Gradient descent algorithms require smooth derivatives of cost functions with respect to vehicle positions; the numerical solutions of the HJB value function will lead to coarse, discontinuous derivatives (alternatively, the fine resolution necessary for smooth HJB derivatives requires excessive computational effort).

This chapter focuses on developing methods with less restrictive requirements than the constant-flow requirements in [110, 6], while remaining computationally tractable and with well behaved derivatives for gradient descent algorithms. In particular, we will assume that the flow field is *affine*, and will develop a spectral decomposition approximation technique to find possible trajectories without numerical integration.

6.1.4 Chapter Structure

The rest of this chapter is organized as follows.

Section 6.2 gives a formal definition of the vehicle and environment model, and the optimization problem of locating the vehicles to minimize the expected travel time to uniformly distributed target locations.

Section 6.3 defines the Zermelo quasimetric as the "time to reach" function for vehicles in flow fields, and defines a coordinate transformation using the Zermelo quasimetric, called "transformation into trajectory space", that is a required tool for analysis of control strategies using the Zermelo-Voronoi partition.

Section 6.4 describes the approximation technique used to calculate the Zermelo quasimetric values efficiently under computation constraints.

Section 6.5 defines the "pseudo-centroid" as the analogue of the "centroid" used in Euclidean-Voronoi partition control strategies, and shows that it is the appropriate tool for analogous control strategies in flow conditions.

Section 6.6 concludes the chapter with a discussion of the consequences of the new methods on control applications and descriptions of possible open research directions.

6.2 Model and Problem Statement

6.2.1 Vehicle and Environment Model

We consider a group of n identical vehicles in a connected, bounded subset of $\mathbb{R}^2$ named D. The boundary of this domain does not represent an obstacle, simply the limits of our region of interest. In other words, if an optimal trajectory exits then re-enters D, it is still a valid trajectory. Each vehicle's position in time is represented by the vector function $x^{(i)}\colon \mathbb{R} \to \mathbb{R}^2$. For clarity we will often drop the superscript when considering a single vehicle. Each vehicle obeys the following *ordinary differential equation* (ODE):

$$\dot{x}^{(i)}(t) = F\left(x^{(i)}(t)\right) + u^{(i)}(t) \tag{6.1}$$

where $F(x)$ is the underlying environmental flow field, varying in space but not time. We confine our attention to affine fields:

$$F(x) = Ax + b \tag{6.2}$$

with $A \in \mathbb{R}^{2\times 2}$, $b \in \mathbb{R}^2$. The control input of the i^{th} vehicle is

$$u^{(i)}\colon \mathbb{R} \to \mathcal{B}_1 \tag{6.3}$$

The unit disc, $\mathcal{B}_1$, is the set $\{x \in \mathbb{R}^2 : \|x\|_2 \leq 1\}$. We also restrict D to be within the region of $\mathbb{R}^2$ where the magnitude of the flow field $F(x)$ is less than unity: this ensures small time controllability [7] of the vehicles.

6.2.2 Problem Statement

As a consequence of small time controllability, every point in D can be reached in finite time starting from any other point in D. We can define the *Zermelo quasimetric* $d_Z(x, x_0)$ as

the minimum time needed to get to x starting from x_0. The Zermelo control problem, which finds the control signal necessary to accomplish this movement, is discussed in section 6.3.1. Our overall objective is to minimize

$$\min_{x^{(i)},V^{(i)}} \mathcal{H}\left(x^{(1)},V^{(1)},\ldots,x^{(n)},V^{(n)}\right) = \sum_{i=1}^{n} \iint_{V^{(i)}} d_Z\left(x,x^{(i)}\right) \mathrm{d}x \quad \text{subject to } V^{(1)},\ldots,V^{(n)} \text{ partition } D \tag{6.4}$$

where $x^{(i)}$ represents the stationary position of the vehicles, and $V^{(i)}$ is the partition which allocates space to each vehicle. Both $x^{(i)}$ and $V^{(i)}$ are unknown and computed by the method presented in this chapter.

6.2.3 Voronoi Partitions

For the previously defined domain D, and a group of n identical vehicles at locations $x^{(i)} : i = 1 \ldots n$, a *Voronoi partition* $V^{(i)} : i = 1 \ldots n$ satisfies the condition:

$$x \in V^{(i)} \Rightarrow d(x,x^{(i)}) \leq d(x,x^{(j)}) \quad \forall j \in \left\{(1 \ldots n) \setminus i\right\} \tag{6.5}$$

where $d(\cdot,\cdot)$ is a metric on D [5]. When not explicitly specified, the Euclidean metric is assumed.

Throughout this chapter, we make the following simplifying assumptions:

- The vehicle positions $x^{(i)}$ are distinct.
- The union of all the cells $V^{(i)}$ may not completely cover D; some subset of Lebesgue measure zero may be unassigned. The one-way implication and non-strict inequality of (6.5) make the Voronoi partition non-unique, and Voronoi cells may or may not be open or closed. Alternately, we can strengthen the definition of the Voronoi partition to be *constructive*:

  $$x \in V^{(i)} \Leftrightarrow d(x,x^{(i)}) < d(x,x^{(j)}) \quad \forall j \in \left\{(1 \ldots n) \setminus i\right\} \tag{6.6}$$

  in which case the partition is unique, all Voronoi cells are open, and the union of the cells is *guaranteed* to not cover D (the boundaries between cells are not covered). Throughout this chapter, we will neglect these distinctions, and assume that the Voronoi partition is functionally unique.

The group of vehicle locations in this definition, $x^{(i)}$, are often called *generators* in the literature; in some control applications, they may not be vehicle starting locations, but rather

goal sites that must be reached by vehicles (for example in [6]). As long as $d(\cdot,\cdot)$ is a true metric (that is, symmetric), reversing the roles of vehicles and goals does not have much effect. In this chapter, we will always assume that the generators are the starting positions of the vehicles.

It is shown in [36] that, when the starting positions are fixed, the Voronoi partition minimizes (6.4); moreover, the global optimum is a *centroidal* Voronoi partition, where each vehicle is located at the centroid of its own Voronoi cell. In Section 6.5 we extend this result to the case of flow fields by defining a *pseudo-centroid* as the analog of the centroid, and showing that locating the vehicles at the pseudo-centroid of their Zermelo-Voronoi cell results in a stationary point of (6.4).

6.3 Optimal Trajectories and Trajectory Coordinate Space

6.3.1 Zermelo Optimal Control

The Zermelo problem is a classic in optimal control [4]. It consists of finding the control $u(t)$ to minimize the transit time to a particular destination, given a system (6.1) with control bound (6.3), but not necessarily restricting the flow field as in (6.2). Pontryagin's minimum principle can be applied to find a necessary condition on the control signal. We reproduce this result here; a useful reference is [16]. For the Zermelo problem:

$$\begin{aligned} \min_{u(\cdot)} \quad & \int_0^T 1 \, \mathrm{d}t \\ \text{subject to:} \quad & \dot{x}(t) = F\left(x(t)\right) + u(t) \\ & \left\|u(\cdot)\right\| \leq 1 \\ & x(0) \text{ and } x(T) \text{ given} \end{aligned} \tag{6.7}$$

the optimal control satisfies:

$$u(t) = \begin{bmatrix} \cos\theta(t) \\ \sin\theta(t) \end{bmatrix} \tag{6.8}$$

$$\begin{aligned} \dot{\theta}(t) = {} & \left.\frac{\partial F_2}{\partial x_1}\right|_{x(t)} \sin^2\theta(t) - \left.\frac{\partial F_1}{\partial x_2}\right|_{x(t)} \cos^2\theta(t) \\ & + \left(\left.\frac{\partial F_1}{\partial x_1}\right|_{x(t)} - \left.\frac{\partial F_2}{\partial x_2}\right|_{x(t)} \right) \sin\theta(t)\cos\theta(t) \end{aligned} \tag{6.9}$$

Equations (6.8) and (6.9) are reproduced from [16].

For an affine flow field:

$$\begin{aligned} F(x) = Ax + b &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} x + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \\ \dot{\theta}(t) &= a_{21} \sin^2 \theta(t) - a_{12} \cos^2 \theta(t) + (a_{11} - a_{22}) \sin \theta(t) \cos \theta(t) \\ &\equiv f_Z(\theta(t); A) \end{aligned} \tag{6.10}$$

We now make some remarks about the nature of the condition imposed in (6.10). First, a vehicle following a "Zermelo trajectory" will always use the maximum control possible, which is intuitively sensible given that the flow field does not vary with time. Second, since the ODE for $\theta(t)$ satisfies the conditions for the existence of a unique solution, the control signal for a Zermelo trajectory is characterized by only the initial heading, $\theta_0 \equiv \theta(0)$. Using standard linear systems theory, we can express the vehicle trajectories that satisfy (6.10) as a function of initial position, initial heading, and time spent travelling:

$$\begin{gathered} p(t, \theta_0; x_0) = e^{At} x_0 + \int_0^t e^{A(t-\tau)} \left(b + \begin{bmatrix} \cos \theta(\tau) \\ \sin \theta(\tau) \end{bmatrix} \right) \mathrm{d}\tau \\ \theta(t) \text{ satisfies (6.10)}, \ \theta(0) = \theta_0. \end{gathered} \tag{6.11}$$

We will call the $p(t, \theta_0; x_0)$ function the "Zermelo trajectory" function or the "parametrized trajectory" function, because the control term has been collapsed into a single parameter θ_0 (the initial control heading).

6.3.2 Zermelo Trajectories satisfy a Necessary but not Sufficient Condition for Optimality

The Pontryagin minimum principle sets a necessary but not sufficient condition for a control signal $u(\cdot)$ to be an optimal solution to the minimum time problem (6.7). In general there are two possibilities for why a trajectory that satisfies (6.10) is nevertheless not a minimum solution for (6.7):

1. The trajectory is unsuitable because it passes through an obstacle or violates some other constraint that was not built into (6.10).
2. Another trajectory satisfying (6.10) exists, and it reaches the target sooner: that is, $p(t, \theta_0; x_0) = p(t', \theta_0'; x_0)$, and $t' < t$.

For the purposes of this study, condition (1) does not arise; we do not consider domains with obstacles or other disqualifying constraints. Proving the non-existence of faster trajectories, as in condition (2), is harder; in general, it requires a complete search of all possible optimal trajectories, which is equivalent to solving the Hamilton-Jacobi-Bellman PDE for the *minimum-time-to-reach* (MTTR) function from the starting location [74, 83]. We can, however, make some useful statements about the optimality of the parametrized trajectories.

Definition 1. *For a quasimetric $d(x_2, x_1)$ and a parametrized trajectory function $p(t, \theta_0; x_0)$, a trajectory from x_0 is said to be "optimal at (t, θ_0)" if $d\left(p(t, \theta_0; x_0), x_0\right) = t$.*

Proposition 1. *For a quasimetric $d(x_2, x_1)$ and a parametrized trajectory function $p(t, \theta_0; x_0)$, if $d\left(p(t, \theta_0; x_0), x_0\right) = t$ then $\forall t' \in [0, t] : d\left(p(t', \theta_0; x_0), x_0\right) = t'$. (In other words: a trajectory that is optimal at (t, θ_0) is optimal at all earlier t').*

Proof: by contradiction. Let us assume the opposite. Suppose $\exists x', t' : p(t', \theta_0; x_0) = x'$, $d(x', x_0) < t'$. Note: $d(x', x_0) > t'$ is inconsistent with the definition of d. Let $x_d = p(t, \theta_0; x_0)$ be the original destination. We know that the original trajectory can be extended from x' to x_d: $\exists \theta_0' : p(t - t', \theta_0'; x') = x_d$. If $d(x', x_0) < t'$ then there exists a trajectory from x_0 to x_d with travel time $d(x', x_0) + (t - t') < t$. This is a contradiction. □

Proposition 2. *Given a quasimetric $d(x_2, x_1)$, an optimal trajectory function $p(t, \theta_0; x_0)$, a destination x_d, and a finite set of generators $x_i : i \in (1 \dots n)$: If $d(x_d, x_i) < d(x_d, x_j)$ for all $j \neq i$, and $p(t_d, \theta_d; x_i) = x_d$, then $d\left(p(t, \theta_d; x_i), x_i\right) < d\left(p(t, \theta_d; x_j), x_i\right)$ for all $j \neq i, t \in [0, t_d]$.*

Informally, this means that if x_d belongs to x_i's Voronoi cell, so does every point on the optimal trajectory from x_i to x_d.

Proof (by contradiction). Let us assume the opposite. Suppose there exists an intermediate point $x' = p(t', \theta_d; x_i), t' \in [0, t_d]$, and a generator $j \neq i$ such that $d(x', x_j) \leq d(x', x_i)$. By Proposition 1, $d(x', x_i) = t'$.

By construction, $d(x_d, x_i) = d(x', x_i) + d(x_d, x')$. By the triangle inequality, $d(x_d, x_j) \leq d(x', x_j) + d(x_d, x')$. Combining these expressions results in $d(x_d, x_j) \leq d(x_d, x_i)$, which is a contradiction. □

6.3.3 Trajectory Coordinate Space

We will use the trajectory function $p(t, \theta_0; x_0)$, defined in (6.11), to create a coordinate transformation.

Fixing x_0 as a parameter, we will use p to transform the Cartesian space of D into "trajectory space" based on t, θ_0. The integrals in (6.4) will be evaluated in this new coordinate system:

$$\iint_V d_Z\left(x, x_0\right) \mathrm{d}x = \iint_{V'} t \left| \det\left(\mathrm{J}\, p(t, \theta_0; x_0)\right)\right| \mathrm{d}\theta_0 \, \mathrm{d}t \tag{6.12}$$

where $\mathrm{J}\, p$ is the Jacobian of $p(t, \theta_0; x_0)$ with respect to its first two variables.

Proposition 3. $\det\left(\mathrm{J}\, p(t, \theta_0; x_0)\right) \neq 0$ *for all $t > 0$, $\theta_0 \in [0, 2\pi]$ such that $p(t, \theta_0; x_0) \in D$.*

Proof. Let $\mathrm{J}\, p = \begin{bmatrix} \nabla_t p & \nabla_{\theta_0} p \end{bmatrix}$.

$$\det(\mathrm{J}\, p) = 0 \Leftrightarrow k\nabla_t p = \nabla_{\theta_0} p \text{ for some scalar } k$$

because $\|\nabla_t p\| \neq 0$ by small-time controllability. However, $k\nabla_t p = \nabla_{\theta_0} p$ would be a contradiction with the known optimality of p.

Because p is has continuous partial derivatives, the domain of interest is connected, and $\det(\mathrm{J}\, p) \neq 0$ in the domain, the sign of $\det(\mathrm{J}\, p)$ is constant everywhere in the domain. We will assume that $\det(\mathrm{J}\, p) > 0$ and drop the absolute value brackets in (6.12). □

Proposition 4. *Given x_0 and V, we can construct $V' \subset \mathbb{R}_+ \times [0, 2\pi]$ such that $p(t, \theta_0; x_0)$ is a legal coordinate transformation from V' to $V \backslash \{x_0\}$.*

Proof. The necessary conditions are [30, 63]:

1. $p\colon V' \to V\backslash\{x_0\}$
2. p is bijective
3. p is differentiable with continuous partial derivatives everywhere in V'
4. $\det(\mathrm{J}\, p) \neq 0$ everywhere in V'

We can construct V' so as to satisfy conditions (1) and (2). Condition (3) is trivial from the definition of p in (6.11), since V' does not include $t = 0$. Condition (4) was shown in Proposition 3. Since $t = 0$ is omitted from V' to satisfy (3), so must x_0 be omitted from V. □

6.4 Approximation Techniques

6.4.1 Classification of Optimal Trajectories

Affine fields can be classified using the eigenvalues of A. Each category of field has a different type of optimal trajectory, that is, solutions to (6.10).

Using trigonometric identities, $f_Z(\theta; A)$ can be rewritten as:

$$f_Z(\theta; A) = m + r\sin(2\theta + \phi)$$

where:

$$m = \frac{a_{21} - a_{12}}{2}$$

$$r = \frac{\sqrt{(a_{11} - a_{22})^2 + (a_{21} + a_{12})^2}}{2}$$

$$\phi = \begin{cases} \sin^{-1}\left(\frac{a_{21}+a_{12}}{2r}\right) & r \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

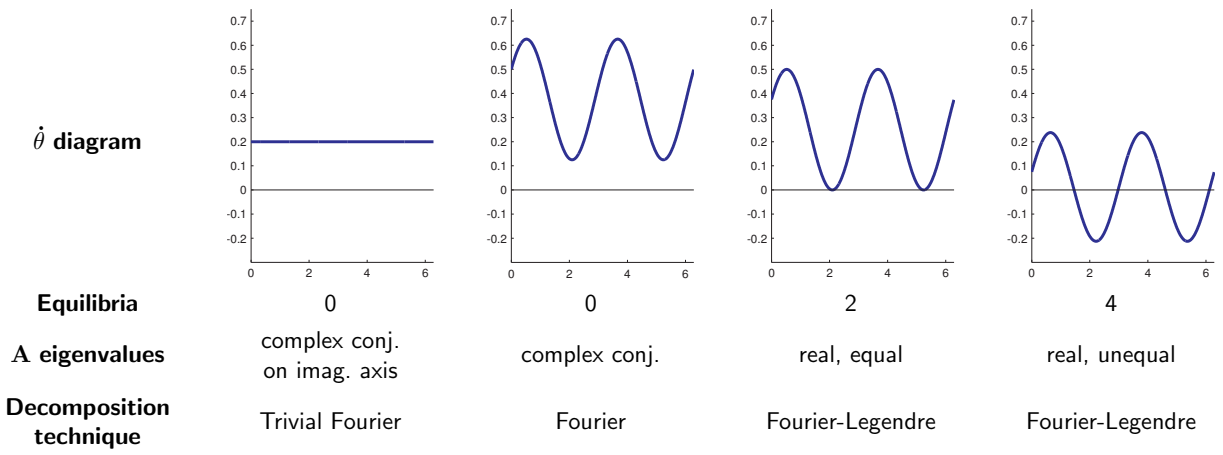

Figure 6.2: Classification of $\dot{\theta}$ ODEs based on eigenvalues of the A matrix. Diagrams are of $\dot{\theta}$ vs θ; each one is a sinusoid of period π, but the mean, amplitude, and equilibria ($\dot{\theta} = 0$ intercepts) vary.

N.eq.	Solution
0	$\theta(t) = \tan^{-1}\left(-\alpha + \beta' \tan(\omega' t + K')\right)$
2	$\theta(t) = \tan^{-1}\left(\frac{K'' - (a_{11} - a_{22})(1 - tK'')}{1 - tK''}\right)$
4	$\theta(t) = \tan^{-1}\left(-\alpha - \beta \coth(\omega t + \frac{1}{2}\log K)\right)$

Table 6.1: Exact solutions to ODEs.

In this form, it is apparent that $\dot{\theta}$ as a function of θ is a sinusoid of period π. The different types of trajectories, and therefore the different classifications of field, arise from differing numbers of equilibria; that is, different numbers of solutions to $f_Z(\theta; A) = 0$. Figure 6.2 shows four different categories of field, based on whether there are 0, 2, or 4 equilibria. A special case, where $r = 0$, is also included, because exact solutions to (6.11) are possible in this case.

The ODE in (6.10) can be solved analytically, which is one of the contributions of this chapter. The solution involves hyperbolic or conventional trigonometric functions depending on the eigenvalues of A, and so it is useful to split the solution by the categories expressed above.

The constants $\alpha, \alpha', \beta, \beta'$ in Table (6.1) are derived from the elements of A. The constants K, K', K'' derive from A as well as θ_0. There are some corner cases not covered in Table (6.1) when various elements of A are zero.

6.4.2 Approximate Trajectory Function

While the analytic solutions in Table 6.1 have some utility (for example, in determining the periodicity of the heading in the zero equilibrium case), using the analytic solutions directly in (6.11) is problematic, because they do not have closed, analytic integral forms. In order to achieve the ease of computation that was set in Section 6.1.3 as the goal, we must find a way of evaluating (6.11) without numerical integration.

We are therefore interested in finding *approximations* to the analytic solutions of Table 6.1 that have the desired properties (closed form, easy to compute integral forms). Our approach is to use two different spectral decomposition techniques, depending on the category of the trajectory. Trajectories in the zero-equilibrium category are periodic, and so conventional Fourier decomposition into sums of sinusoids is possible. The two-equilibrium and four-equilibrium case each involve trajectories that asymptotically approach stable headings. A sum of real exponentials is, intuitively, a reasonable way to approximate such functions. Formally, we use a Fourier-Legendre decomposition of a sample trajectory that has been remapped from $t \in (-\infty, \infty)$ to $t' \in (-1, 1)$ using a bi-infinite exponential map M:

$$M \colon \mathbb{R} \to (-1, 1)$$

$$M(t) = \begin{cases} 1 - e^{-t} & t \geq 0 \\ -1 + e^{t} & \text{otherwise} \end{cases}$$

$$M^{-1}(t') = \begin{cases} -\log(1 - t') & t' \geq 0 \\ \log(1 + t') & \text{otherwise} \end{cases}$$

We perform the spectral decomposition and approximation on the control input term:

$$u(t) = \begin{bmatrix} \cos\theta(t) \\ \sin\theta(t) \end{bmatrix}$$

For the two different approaches (Fourier and Fourier-Legendre) the approximations of $u(t)$ are:

$$\tilde{u}_{\mathrm{F}}(t) = \sum_{i=-N/2}^{N/2} e^{ji\omega(t + t_{\mathrm{sh}}(\theta_0))} \begin{bmatrix} c_i \\ s_i \end{bmatrix} \tag{6.13}$$

$$\tilde{u}_{\mathrm{FL}}(t) = \sum_{i=0}^{N} L_i(M(t + t_{\mathrm{sh}}(\theta_0))) \begin{bmatrix} c_i \\ s_i \end{bmatrix} \tag{6.14}$$

where L_i is the i^{th} Legendre polynomial. c_i and s_i are the decomposition coefficients for the $\cos(\theta)$ and $\sin(\theta)$ components of u, respectively. In the zero-equilibrium case, all optimal trajectories can be viewed as time-shifted versions of a nominal trajectory; since trajectories

can be characterized by their initial heading, different trajectories are achieved using an initial-heading-dependent time shift term $t_{\text{sh}}(\theta_0)$. This is also true for the two-equilibrium and four-equilibrium case, except that there must be one nominal trajectory for each attracting basin of $\dot{\theta}$; hence, the two-equilibrium case will have two nominal trajectories, and two versions of (c_i, s_i), and so on for the four-equilibrium case.

6.4.3 Algorithm for Finding the Zermelo-Voronoi Partition

Algorithm 1 gives a pseudocode outline of how to use the approximated trajectory function $p(t, \theta_0; x_0)$ to find a polyline approximation to the Zermelo-Voronoi partition. Function ComputeZVPartition takes the affine parameters A, b, a polyline outline of the domain D, and a list of vehicle positions $\{x\}$, and returns a list of (id, boundary) pairs, where "boundary" is a closed polyline in trajectory space (that is, in (t, θ_0) pairs), describing the boundary of each vehicle's Zermelo-Voronoi cell.

The first step in the process is to find the "equal time to reach" boundary between each pair of vehicles, using FindOneBoundary. This is analogous to finding the perpendicular bisector between two generators when finding the Euclidean-Voronoi partition. Because $d_Z(x, x^{(i)})$ is continuous in $x \in D$, and a quasimetric, this boundary is guaranteed to exist. The second step is to use UnifyBoundaries for each vehicle, to combine all the pairwise boundaries involving that vehicle into a single closed boundary representing its Zermelo-Voronoi cell.

The individual cells are restricted to the domain D by including the "domain boundary" in the pairwise boundary list using FindDomainBoundary. The algorithm assumes that the domain boundary is monotonically oriented for all generators inside D, that is,

$$\forall x_0 \in \text{int}(D) : \theta_0 \text{ is monotonic when} (t, \theta_0) = p^{-1}(\partial D(s); x_0)$$

where ∂D is the boundary of D. The analogous property for the Euclidean context is simply that D is convex. It is difficult to establish a simpler condition for the Zermelo case. We assume that for simple D, like rectangles, this property holds, at least for the region inside D where the vehicles actually go; if this property does not hold, more careful handling of the domain boundary in FindDomainBoundary and UnifyBoundaries would be necessary, but the computation would still be feasible.

Function FindOneBoundary uses Newton-Raphson zero-finding procedures to find a seed point that both vehicles can reach at the same time. The pairwise boundary consists of triples $(t, \theta_{01}, \theta_{02})$, representing the coordinates in trajectory space of common points; all these points must satisfy the condition

$$p(t, \theta_{01}; x_1) = p(t, \theta_{02}; x_2)$$

The boundary is extended by finding the nullspace of the Jacobian of the "endpoint distance" function

$$e(t, \theta_{01}, \theta_{02}) = p(t, \theta_{01}; x_1) - p(t, \theta_{02}; x_2)$$

This nullspace is dimension 1, and represents a perturbation that can be made to the $(t, \theta_{01}, \theta_{02})$ triple while preserving the "common point" property. ExtendBoundary finds this nullspace, takes a small step in this direction, then uses a Newton-Raphson procedure to correct the small gap between the endpoints caused by higher order terms. It repeats this procedure until it intersects the domain boundary.

The step size parameter in ExtendBoundary controls the resolution of the polyline approximation to the true Zermelo-Voronoi boundaries. The zero-finding techniques used in FindOneBoundary and ExtendBoundary are the motivation to use the approximated trajectory function described in Section 6.4.2. Since $p(t, \theta_0; x_0)$ is just a sum of exponentials, it is fast to compute compared to the numeric integration over $[0, t]$ that would be necessary if the analytic solution were used. In addition, $\frac{\partial p}{\partial t}$ and $\frac{\partial p}{\partial \theta_0}$ are well-behaved and just as fast to compute, which permits the use of Newton-Raphson techniques to find zeros instead of more complicated techniques required when the partial derivatives are noisy.

Algorithm 2 contains some supporting functions for the Zermelo-Voronoi procedure. Several simple functions, like Min, Outside, Sort, and Clip, are not explicitly described.

6.5 The Pseudo-Centroid as Stationary Point of the Cost Function

Du *et al.* [36] showed that any minimizer of the cost function (6.4) must be a *centroidal Voronoi partition*; that is, the assignment of $V^{(i)}$ to $x^{(i)}$ must satisfy the Euclidean-Voronoi condition, and that each generator $x^{(i)}$ must be at the centroid of its cell $V^{(i)}$. Cortés *et al.* [29] showed how to design a decentralized control scheme to minimize (6.4) by having each vehicle use a gradient descent scheme using its *individual* cost function

$$\mathcal{F}_{V^{(i)}}(x^{(i)}) = \iint_{V^{(i)}} d\left(x, x^{(i)}\right) \mathrm{d}x \tag{6.15}$$

In this section, we extend these results to the Zermelo-Voronoi case. In subsection 6.5.1 we show that the assignment partitions V_i must be the Zermelo-Voronoi partition in order to be optimal. This argument is very similar to [36] and is included for completeness. In Section 6.5.2 we show how to compute the analogous gradient of (6.15) for the Zermelo-Voronoi case. While the Euclidean-Voronoi partition has a link between optimization and geometry, in that the centroid of a Voronoi cell must be a minimizer of (6.15), the minimizer in the Zermelo-Voronoi case does not have an intuitive geometric description. We call the minimizer the *pseudo-centroid* in order to highlight the analogy to the Euclidean-Voronoi case.

Algorithm 1 Compute the Zermelo Voronoi partition

function ComputeZVPartition($A, b, D, \{x\}$)
▷ A, b: affine parameters. D: domain. $\{x\}$: vehicle positions.
$p \leftarrow$ MakeApproximationFunction(A, b) ▷ $p : (t, \theta_0, x_0) \to x$
$n \leftarrow [\{x\}]$ ▷ n: number of vehicles.
$B_{\text{pairwise}} \leftarrow \{\}$ ▷ initialize empty collection of "pairwise" boundaries
for $i \leftarrow 1, (n-1)$ **do**
for $j \leftarrow (i+1), n$ **do**
$B_{\text{pairwise}} \leftarrow B_{\text{pairwise}}, \Big((i,j), \text{FindOneBoundary}(p, D, x^{(i)}, x^{(j)})\Big)$
end for
$B_{\text{pairwise}} \leftarrow B_{\text{pairwise}}, \Big((i, \varnothing), \text{FindDomainBoundary}(p, D, x^{(i)})\Big)$ ▷ add extent of domain
end for
$B_{\text{combined}} \leftarrow \{\}$ ▷ initialize empty collection of "combined" boundaries
for $i \leftarrow 1, n$ **do**
$B_i \leftarrow \text{SelectBoundaries}(i, B_{\text{pairwise}})$ ▷ Filter: boundaries where i participates
$B_{\text{combined}} \leftarrow B_{\text{combined}}, \big(i, \text{UnifyBoundaries}(B_i)\big)$
end for
return B_{combined}
end function
function FindOneBoundary(p, D, x_1, x_2)
$T = (t, \theta_1, \theta_2) \leftarrow \text{SearchOneCommonPoint}(p, D, x_1, x_2)$
$n \leftarrow \text{NullDirection}(T, p, x_1, x_2)$
$H_+ \leftarrow \text{ExtendBoundary}(p, D, T, x_1, x_2, n)$
$H_- \leftarrow \text{ExtendBoundary}(p, D, T, x_1, x_2, -n)$
return $\big\{\text{Reverse}(H_-), T, H_+\big\}$
end function
function UnifyBoundaries($\{B\}$)
$B_{\text{unified}} \leftarrow \{\}$
$L_\theta \leftarrow \text{AllAngles}(\{B\})$ ▷ L_θ: list of all θ_{01} elements in all boundaries
for $\theta_{01} \leftarrow L_\theta$ **do**
$L_t \leftarrow \{\}$
for $i \leftarrow 1, [\{B\}]$ **do**
$L_t \leftarrow L_t, \text{LinInterpTime}(\theta_{01}, B_i)$
end for
$B_{\text{unified}} \leftarrow B_{\text{unified}}, (\text{Min}(L_t), \theta_{01})$
end for
return B_{unified}
end function

Algorithm 2 Supporting functions for Algorithm 1

function FINDDOMAINBOUNDARY(p, D, x_1)
  $B_D \leftarrow \{\}$
  for $x_d \leftarrow D$ **do** ▷ D is a polyline (list of coordinates)
    $(t, \theta_0) \leftarrow$ SEARCHTRAJECTORY(x_d, x_1)
    $B_D \leftarrow B_D, (t, \theta_0, \varnothing)$ ▷ The "other" θ_0 value has no meaning
  end for
  return SORT(B_D) ▷ Sort by θ_{01}, the second item of each triple
end function
function SEARCHONECOMMONPOINT(p, x_1, x_2)
  $x' \leftarrow$ MIDPOINT(x_1, x_2) ▷ Simple if D is convex. More complicated otherwise.
  $(t_1, \theta_{01}) \leftarrow$ SEARCHTRAJECTORY(x', x_1)
  $(t_2, \theta_{02}) \leftarrow$ SEARCHTRAJECTORY(x', x_2)
  $(t_1, \theta_{01}, t_2, \theta_{02}) \leftarrow$ NEWTONRAPHSON($((p(t_1, \theta_{01}, x_1) - p(t_2, \theta_{02}, x_2)), (t1 - t2)), (t_1, \theta_{01}, t_2, \theta_{02})$)
  return $(t_1, \theta_{01}, \theta_{02})$ ▷ $t_1 = t_2$ so it doesn't matter which one we use
end function
function EXTENDBOUNDARY(p, D, T, x_1, x_2, n)
  $T' \leftarrow T + \epsilon n$ ▷ ϵ is a tunable step size
  $T' \leftarrow$ NEWTONRAPHSON($p(t, \theta_{01}, x_1) - p(t, \theta_{02}, x_2), (t, \theta_{01}, \theta_{02}) = T'$)
  if OUTSIDE(T', D) **then**
    return CLIP(T', T, D) ▷ Reached the edge of the domain: no more recursion
  end if
  $n' \leftarrow$ NULLDIRECTION(T', p, x_1, x_2)
  if $n \cdot n' > 0$ **then** ▷ Continue extending with tail-recursion; be sure to keep moving in the same direction
    return T', EXTENDBOUNDARY(p, D, T', x_1, x_2, n')
  else
    return T', EXTENDBOUNDARY($p, D, T', x_1, x_2, -n'$)
  end if
end function
function NULLDIRECTION($(t, \theta_{01}, \theta_{02}), p, x_1, x_2$)
  $J \leftarrow \begin{bmatrix} \nabla_t(p(t, \theta_{01}, x_1) - p(t, \theta_{02}, x_2)) & \nabla_{\theta_0} p(t, \theta_{01}, x_1) & -\nabla_{\theta_0} p(t, \theta_{02}, x_2) \end{bmatrix}$
  $n \leftarrow \mathcal{N}(J)$ ▷ $\dim(\mathcal{N}(J))$ should be 1
  return n
end function

6.5.1 Zermelo-Voronoi Partition as optimal

We will now show that

$$\begin{aligned}\left(x^{(1)}, V^{(1)}, \ldots, x^{(n)}, V^{(n)}\right) \in \operatorname{argmin} \mathcal{H} \Rightarrow \\ \left(V^{(1)}, \ldots, V^{(n)}\right) \text{ is a Voronoi partition of } D \\ \text{with generators } \left(x^{(1)}, \ldots, x^{(n)}\right).\end{aligned} \tag{6.16}$$

with $\mathcal{H}$ defined as in Equation (6.4). Let $\left(\hat{V}^{(1)}, \ldots, \hat{V}^{(n)}\right)$ be a Voronoi partition of D with generators $\left(x^{(1)}, \ldots, x^{(n)}\right)$, and let $\left(V^{(1)}, \ldots, V^{(n)}\right)$ be a partition of D that does not satisfy the conditions to be a Voronoi partition. We can construct "assignment functions" $\hat{a}\colon D \to \{1, 2, \ldots, n\}$ and $a\colon D \to \{1, 2, \ldots, n\}$:

$$\begin{aligned}\hat{a}(x) = i \Leftrightarrow x \in \hat{V}_i \\ a(x) = i \Leftrightarrow x \in V_i\end{aligned}$$

For any $x \in D$, it is true that

$$d_Z\left(x, x^{(\hat{a}(x))}\right) \leq d_Z\left(x, x^{(a(x))}\right)$$

This inequality holds strictly for a measurable subset of D if $\left(V^{(1)}, \ldots, V^{(n)}\right)$ is non-Voronoi.

6.5.2 Pseudo-Centroid as optimal location within cell

The pseudo-centroid of a Zermelo-Voronoi cell must be found numerically, through a gradient descent algorithm to find the minimizer of (6.15), the individual cost function. The implicit definition of Zermelo-Voronoi cell V means that numeric solutions are unavoidable; however, we can exploit the closed form of the approximated trajectory function to avoid finding the gradient of (6.15) by finite differences.

Proposition 5. *If we have an individual cost function $\mathcal{F}_V(x_0)$ as defined in (6.15), where $d_Z(x, x_0)$ is the Zermelo quasimetric under affine dynamics (6.1,6.2,6.3), then the gradient with respect to the starting position x_0 is:*

$$\nabla \mathcal{F}_V(x_0) = \iint_{V'} e^{A^T t}\left(\frac{\partial p}{\partial t} + tA^T\left(\begin{bmatrix}\frac{\partial p_2}{\partial \theta_0} \\ -\frac{\partial p_1}{\partial \theta_0}\end{bmatrix}\left(\det(\mathrm{J}\, p)\right)^{-1} + \frac{\partial p}{\partial t}\right)\right)\det(\mathrm{J}\, p)\,\mathrm{d}t\,\mathrm{d}\theta_0 \tag{6.17}$$

where V' is the image of V in the trajectory coordinate transformation described in Section 6.3.3.

Proof. We start by constructing the directional derivative $\nabla_v \mathcal{F}_V$ of the objective function, using the coordinate transform defined in Section 6.3.3.

$$\begin{aligned}\nabla_v \mathcal{F}(x_0) = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \Big(&\iint_{V'} t \det \big(\mathrm{J}\, p(t, \theta_0; x_0 + \epsilon v) \big) \, \mathrm{d}\theta_0 \, \mathrm{d}t \\ &- \iint_{V''} t \det \big(\mathrm{J}\, p(t, \theta_0; x_0) \big) \, \mathrm{d}\theta_0 \, \mathrm{d}t \Big)\end{aligned} \tag{6.18}$$

Note that the domain of integration for the integrals are denoted V' and V'' to reflect the fact that they are based on different coordinate transformations.

Now if we are in a linear/affine field defined by A and b:

$$p(t, \theta_0; x_0 + \epsilon v) = p(t, \theta_0; x_0) + \epsilon e^{At} v \tag{6.19}$$

$$\begin{aligned}\det \big(\mathrm{J}\, p(t, \theta_0; x_0 + \epsilon v) \big) &= \det \big(\mathrm{J}\, p(t, \theta_0; x_0) \big) \\ &+ \epsilon \begin{bmatrix} \frac{\partial p_2(t, \theta_0; x_0)}{\partial \theta_0} & -\frac{\partial p_1(t, \theta_0; x_0)}{\partial \theta_0} \end{bmatrix} A e^{At} v\end{aligned} \tag{6.20}$$

$$j_1 \equiv \det \big(\mathrm{J}\, p(t, \theta_0; x_0) \big) \tag{6.21}$$

$$j_2 = \begin{bmatrix} \frac{\partial p_2(t, \theta_0; x_0)}{\partial \theta_0} & -\frac{\partial p_1(t, \theta_0; x_0)}{\partial \theta_0} \end{bmatrix} A e^{At} v \tag{6.22}$$

$$j_1 + \epsilon j_2 \equiv \det \big(\mathrm{J}\, p(t, \theta_0; x_0 + \epsilon v) \big) \tag{6.23}$$

j_1 and j_2 are shortcuts for readability. Now define indicator functions based on p:

$$I_V(t, \theta_0; x_0) \equiv \begin{cases} 1 & \text{if } p(t, \theta_0; x_0) \in V \\ 0 & \text{otherwise} \end{cases} \tag{6.24}$$

Using these indicator functions we can expand the domain of the two integrals in (6.18) to a larger, common domain, and combine them into one integral. We can confine t to $[0, T_{\max}]$ because D is bounded.

$$\nabla_v \mathcal{F}_V(x_0) = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{T_{\max}} \int_0^{2\pi} t \Big(I_V(; x_0 + \epsilon v)(j_1 + \epsilon j_2) - I_V(; x_0) j_1 \Big) \, \mathrm{d}\theta_0 \, \mathrm{d}t \tag{6.25}$$

Now create some new subsets of $\mathbb{R}^2$ based on V, t, v, ϵ:

$$V_0(t, \epsilon, v) = \Big\{ x : x \in V \wedge x - \epsilon e^{At} v \in V \Big\} \tag{6.26}$$

$$V_+(t, \epsilon, v) = \Big\{ x : x \notin V \wedge x - \epsilon e^{At} v \in V \Big\} \tag{6.27}$$

$$V_-(t, \epsilon, v) = \Big\{ x : x \in V \wedge x - \epsilon e^{At} v \notin V \Big\} \tag{6.28}$$

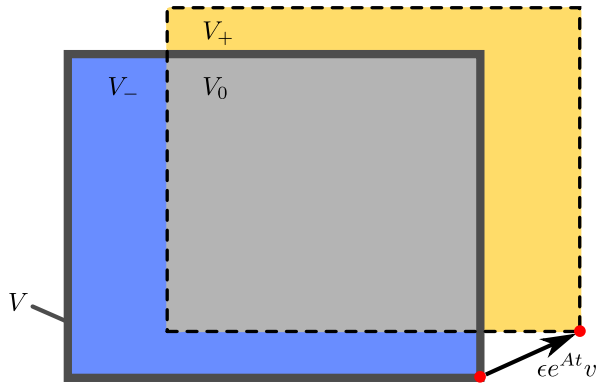

Figure 6.3: Example of how the sets V_0, V_+, V_- relate to V and $\epsilon e^{At} v$

See Figure 6.3. Informally, V_0 is the set of points that are matched by $I_V(t, \theta_0; x_0)$ and $I_V(t, \theta_0; x_0 + \epsilon v)$; V_+ is the set of points that are only matched by $I_V(t, \theta_0; x_0 + \epsilon v)$; and V_- is the set of points that are only matched by $I_V(t, \theta_0; x_0)$. The following relations hold:

$$\begin{aligned} I_{V_0(t,\epsilon,v)}(t, \theta_0; x_0) + I_{V_-(t,\epsilon,v)}(t, \theta_0; x_0) \\ = I_V(t, \theta_0; x_0) \\ I_{V_0(t,\epsilon,v)}(t, \theta_0; x_0) + I_{V_+(t,\epsilon,v)}(t, \theta_0; x_0) \\ = I_V(t, \theta_0; x_0 + \epsilon v) \end{aligned}$$

We will now drop all the arguments of these indicator functions for readability. Substitute these new indicator functions into (6.25).

$$\nabla_v \mathcal{F}_V(x_0) = \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{T_{\max}} \int_0^{2\pi} t \Big(\left(I_{V_0} + I_{V_+} \right) (j_1 + \epsilon j_2) - \left(I_{V_0} + I_{V_-} \right) j_1 \Big) \, \mathrm{d}\theta_0 \, \mathrm{d}t \tag{6.29}$$

Cancel out the $I_{V_0} j_1$ terms:

$$\begin{aligned} \nabla_v \mathcal{F}_V(x_0) &= \lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \int_0^{T_{\max}} \int_0^{2\pi} t \Big(\epsilon I_{V_0} j_2 + \epsilon I_{V_+} j_2 + I_{V_+} j_1 - I_{V_-} j_1 \Big) \, \mathrm{d}\theta_0 \, \mathrm{d}t \\ &= \lim_{\epsilon \to 0^+} \int_0^{T_{\max}} \int_0^{2\pi} t \Big(I_{V_0} j_2 + I_{V_+} j_2 + \tfrac{1}{\epsilon} I_{V_+} j_1 - \tfrac{1}{\epsilon} I_{V_-} j_1 \Big) \, \mathrm{d}\theta_0 \, \mathrm{d}t \end{aligned} \tag{6.30}$$

Now we check each term's behavior in the limit.

$$\lim_{\epsilon \to 0^+} \int_0^{T_{\max}} \int_0^{2\pi} t I_{V_0} j_2 \, \mathrm{d}\theta_0 \, \mathrm{d}t = \int_0^{T_{\max}} \int_0^{2\pi} t I_V j_2 \, \mathrm{d}\theta_0 \, \mathrm{d}t \tag{6.31}$$

$$\lim_{\epsilon \to 0^+} \int_0^{T_{\max}} \int_0^{2\pi} t I_{V_+} j_2 \, \mathrm{d}\theta_0 \, \mathrm{d}t = 0 \tag{6.32}$$

$$\lim_{\epsilon \to 0^+} \int_0^{T_{\max}} \int_0^{2\pi} t \tfrac{1}{\epsilon} \left(I_{V_+} - I_{V_-} \right) j_1 \, \mathrm{d}\theta_0 \, \mathrm{d}t = \int_{\partial V} t \hat{n}^T e^{At} v \, \mathrm{d}s \tag{6.33}$$

Of these three limit expressions, (6.33) is the only non-obvious one. Convert back to Cartesian coordinates (very easy). Finding area of $I_{V_+} - I_{V_-}$ becomes equivalent to finding signed area between boundary and boundary+$\epsilon e^{Ad_Z(x,x_0)}v$. Assume $e^{Ad_Z(x,x_0)}$ is smooth enough that it can be taken as constant locally. Then signed area becomes $\int_s \epsilon e^{Ad_Z(x(s),x_0)}v \cdot \hat{n}\,\mathrm{d}s$. $\hat{n}$ is the outward normal vector to the boundary of V, and s is the scalar parameter for the boundary.

Combining this all:

$$\begin{aligned}\nabla_v \mathcal{F}_V(x_0) &= \iint_{V'} t j_2 \,\mathrm{d}\theta_0 \,\mathrm{d}t + \int_{\partial V} t\hat{n}^T e^{At} v \,\mathrm{d}s \\ &= \iint_{V'} e^{A^T t} t A^T \begin{bmatrix} \frac{\partial p_2}{\partial \theta_0} \\ -\frac{\partial p_1}{\partial \theta_0} \end{bmatrix} \mathrm{d}t \,\mathrm{d}\theta_0 \\ &\quad + \int_{\partial V} d_Z\left(x(s), x_0\right) e^{Ad_Z(x(s),x_0)} v \;\cdot \hat{n}\,\mathrm{d}s\end{aligned} \tag{6.34}$$

Now we have an expression with two integral terms; first, an area integral in trajectory space; second, a boundary integral in Cartesian space. We will change the boundary integral to an area integral with the divergence theorem, then transform it into trajectory space and combine the two terms.

Using the divergence theorem:

(6.35)

The transformation to trajectory space is simple (and $\nabla d_Z = \frac{\partial p}{\partial t}$ in trajectory space). The term v can be factored out on the right, and we can convert from the directional derivative to the gradient (if $\nabla_v \mathcal{F}_V = g^T v$ then $\nabla \mathcal{F}_V = g$).

Once transformed back to trajectory space, the resulting expression for the gradient is

$$\nabla \mathcal{F}_V(x_0) = \iint_{V'} e^{A^T t}\left(\frac{\partial p}{\partial t} + tA^T\left(\begin{bmatrix} \frac{\partial p_2}{\partial \theta_0} \\ -\frac{\partial p_1}{\partial \theta_0} \end{bmatrix}\left(\det(\mathrm{J}\,p)\right)^{-1} + \frac{\partial p}{\partial t}\right)\right)\det(\mathrm{J}\,p)\,\mathrm{d}t\,\mathrm{d}\theta_0$$

□

6.5.3 Lloyd's Algorithm

Lloyd's algorithm, developed in the 1960s, is a simple way of finding a configuration of Voronoi generators where each generator is at the centroid of its Voronoi cell. It was shown in [36] that such a configuration is a local minimizer of the cost function (6.4). A quick sketch of Lloyd's algorithm is given in Algorithm 3.

Algorithm 3 Lloyd's algorithm for centroidal Voronoi configuration

function LLOYD($D, \{x\}$)
▷ D: domain. $\{x\}$: generators.
  $\{V\} \leftarrow$ VORONOICELLS($D, \{x\}$)
  for $i \leftarrow 1, [\{x\}]$ **do**
    $x_i \leftarrow$ CENTROID(V_i)
  end for
  Repeat until convergence
end function

Under the Zermelo quasimetric, the minimizing location within a Zermelo-Voronoi cell does not coincide with the geometric centroid, as it does in the Euclidean case. There is such a minimizing location, however, which we call the *pseudo-centroid*. Finding the pseudo-centroid within a cell is accomplished with a gradient descent algorithm on the individual cost function (6.15), using the gradient as calculated in Proposition 5.

6.5.4 Demonstration of Lloyd's Algorithm

Figures 6.4–6.7 show an example of the Lloyd's algorithm approach to control with five vehicles in an affine flow field. In Figure 6.4, the five vehicles have an arbitrary layout that does not minimize the global cost function. Figure 6.5 shows the paths they follow according the algorithm outlined in Section 6.5.3, and Figure 6.6 shows their position when the algorithm terminates (and the Zermelo-Voronoi partition of the space). Figure 6.7 shows the evolution of the cost function with time (the total as well as the cost functions for each of the five vehicles).

6.6 Conclusions

By generalizing the Voronoi partition from the standard Euclidean metric to the case of the Zermelo quasimetric, we can extend multivehicle control techniques based on the Voronoi partition to the case of vehicles influenced by an environmental flow field. Computing the Zermelo-Voronoi partition in the general case will require computationally intensive techniques such as the Hamilton-Jacobi-Bellman PDE. By restricting the flow to the category of affine fields, analytic solutions to the optimal control problem can be exploited to approximate the optimal trajectories, leading to faster solutions for the Zermelo-Voronoi partition. One of the side benefits of the approximation techniques used is the availability of clean partial derivatives of the cost functions, which are useful for implementing the analogues of the Voronoi-based control techniques.

The primary motivation for the study of Voronoi-based multivehicle control techniques is

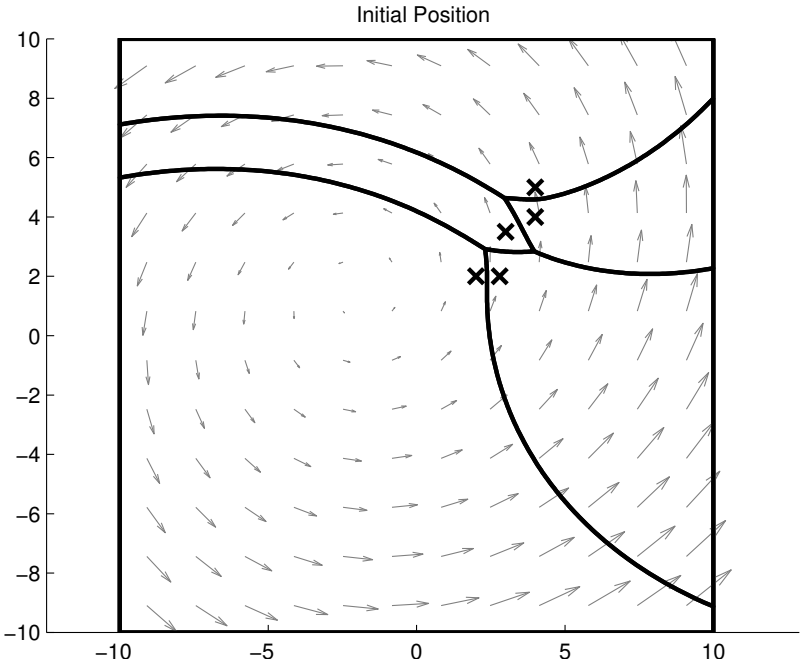

Figure 6.4: Initial position of vehicles and Zermelo-Voronoi partition for demonstration of Lloyd's algorithm.

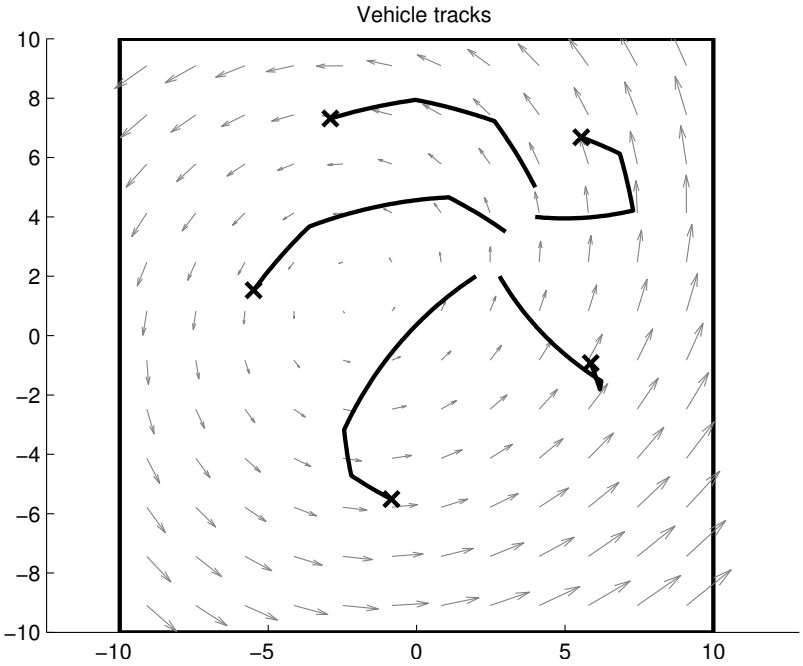

Figure 6.5: Paths followed by vehicles. The final positions are marked with an X.

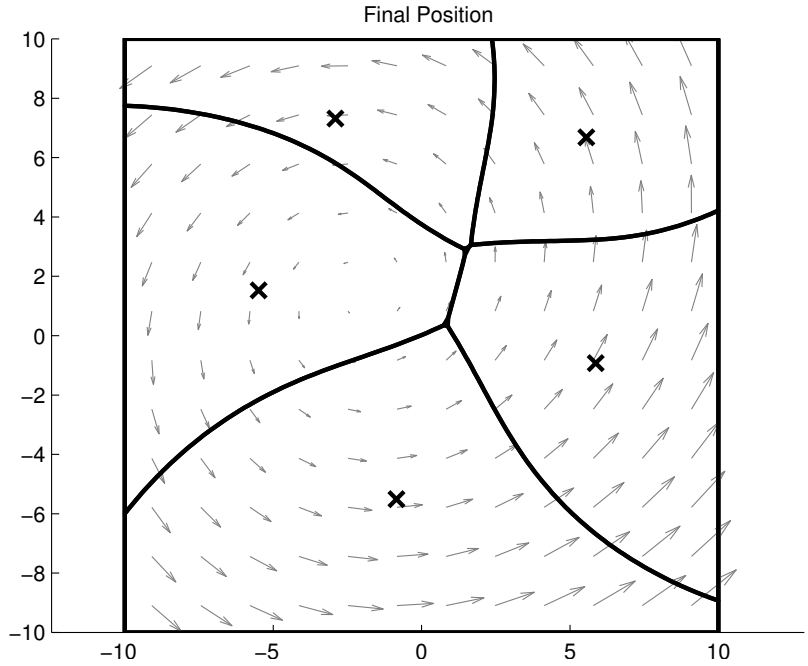

Figure 6.6: Final position of vehicles and Zermelo-Voronoi partition.

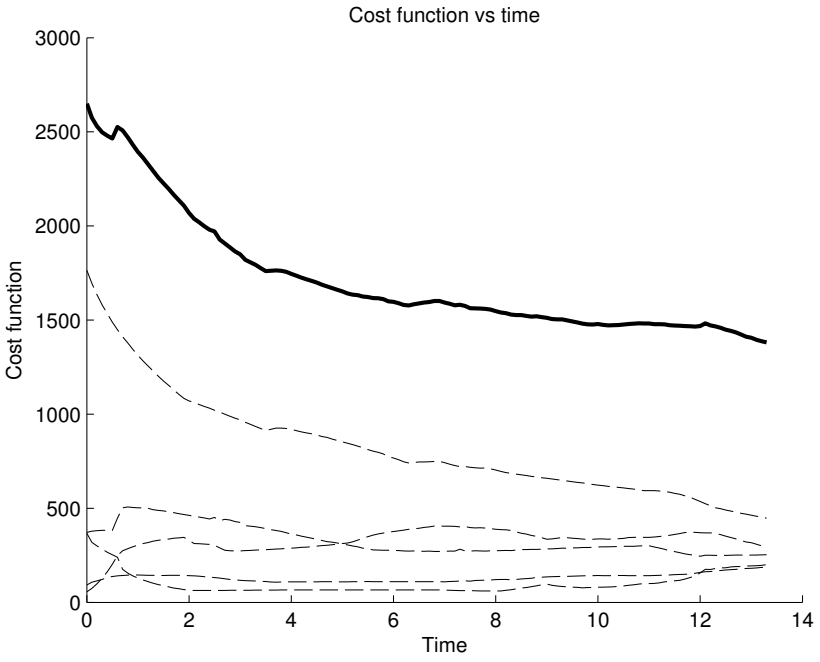

Figure 6.7: Evolution of cost functions (solid line: total cost function; dashed lines: individual vehicle cost functions).

their potential for decentralization. In [29], a variant of the Voronoi partition for decentralization was defined. Under such a variation, determining a vehicle's Voronoi cell does not require sharing information with every other vehicle in the fleet, but rather with a reduced subset based on proximity. The present work has extended the Voronoi multivehicle control techniques to the flow field case, but only in the centralized sense. For example, Algorithm 1 checks the boundaries between every pair of vehicles. One future avenue for research is the extension of the Voronoi variants in [29] to the flow field scenario.

One major drawback of the approximation techniques presented in Section 6.4 is the complicated nature of the ODE solutions for the optimal control. The two major families of solutions (zero-equilibria and four-equilibria), along with the numerically rare two-equilibria special case, make implementing the approximated trajectories complicated. Were this approach to be tried with higher-dimensional problems, the number of solution families would grow combinatorially. While the solutions themselves would not be more complex (they would still be basic spectral decompositions), actually implementing all of the different possibilities would get more difficult. In a way, this approach has shifted complexity from *computational*, in the easy-to-implement but computationally heavy PDE approaches, to *analytic*, in the computationally cheap but difficult to implement approximation techniques presented here.

Chapter 7

Decentralized Time Synchronized Channel Hopping Scheduling for Mobile Sensor Communications

Actuated mobile sensors in an unstructured environment present a demanding set of requirements for wireless communications. In particular, fleet control schemes like the Zermelo-Voronoi algorithm described in Chapter 6 require point-to-point communication between sensors in the environment in order to share position information and determine the optimum location for members of the fleet. This chapter presents a new scheduling algorithm for an existing low-power wireless radio stack in order to adapt this technology for the communication needs of mobile sensors with dynamic connectivity.

Time Synchronized Channel Hopping (TSCH) is an existing Medium Access Control scheme which enables robust communication through channel hopping and high data rates through synchronization. It is based on a time-slotted architecture, and its correct functioning depends on a schedule which is typically computed by a central node. This chapter presents a scheduling algorithm for TSCH networks which both is distributed and which copes with mobile nodes. Two variations on scheduling algorithms are presented. Aloha-based scheduling allocates one channel for broadcasting advertisements for new neighbors. Reservation-based scheduling augments Aloha-based scheduling with a dedicated timeslot for targeted advertisements based on gossip information. A mobile *ad hoc* motorized sensor network with frequent connectivity changes is studied, and the performance of the two proposed algorithms is assessed. The performance analysis uses both simulation results and the results of a field deployment of floating wireless sensors in an estuarial canal environment. Reservation-based scheduling performs significantly better than Aloha-based scheduling, suggesting that the improved network reactivity is worth the increased algorithmic complexity and resource consumption.

7.1 Introduction

As discussed in Chapter 3, the Generation 3 drifter carry two communication systems: a GSM module for transmissions to a central server, and a low-power, low-range IEEE802.15.4-2006 [59] radio for communication between nodes.

The GSM communication channel is both expensive (both monetarily and in terms of energy consumption) and unreliable (due to variable GSM coverage, particularly on the water). One strategy for delivering data from individual nodes to a remote server is to have one or several nodes with good GSM connection act as *ad hoc* sink nodes. Nodes connected by IEEE802.15.4-2006 links that do not have their own GSM connections available can send their data to one of the sinks, which retransmits it *via* GSM to the server. Since it is not known *a priori* which nodes have GSM connectivity, the design objective for the IEEE802.15.4-2006 network must be to maximize point-to-point connectivity.

We define the *physical connectivity graph* to be the ensemble of wireless links "good enough" to be used for communication at a given instant in time. We define the *logical connectivity graph* to be the set of links scheduled to be used at the same instant.

Due to the water currents, the mobility of the nodes means that the physical connectivity between nodes changes significantly over time. Global connectivity is not guaranteed. Therefore, centralized schemes for determining a communication schedule are poor fits for the problem. Our goal is to develop an algorithm which schedules intermittent bi-directional links between neighboring nodes as these links become available. We assess candidate schemes by evaluating how close the logical connectivity gets to the physical connectivity; that is, how many of the possible links are actually scheduled by the protocol.

Time Synchronized Channel Hopping (TSCH) is a *Medium Access Control* (MAC) scheme which enables robust communication through channel hopping and high data rates through synchronization. It is based on a time-slotted architecture, in which a schedule indicates to the nodes on which timeslot and on which channel to transmit/receive data to/from which neighbor. TSCH has been standardized by the the IEEE802.15.4e Working Group [60], and has been published as an amendment to the IEEE802.15.4 standard. In this chapter, the terms "timeslot" and "slot" are used interchangeably.

TSCH only defines the mechanism and makes no recommendation on how the schedule should be built. Typically, nodes report their communication needs (expressed in terms of throughput, reliability and latency) to a central scheduler, which computes a schedule and injects this into the network. This technique has proven perfectly adequate for static networks such as industrial control *Wireless Sensor Networks* (WSNs). A distributed solution seems more appropriate for mobile networks. In those type of networks, each topological change would have to be reported to the central scheduler, which would have to re-compute a schedule and inform the nodes about the change. This is sometimes infeasible since this central scheduler may be disconnected from parts of the network.

This chapter presents two related distributed scheduling algorithms to be used on top of a TSCH MAC protocol. These algorithms are designed for the scheduling needs of

IEEE802.15.4-2006 radios in applications with high mobility. In particular, these algorithms are purely decentralized. The first algorithm, "Aloha-based scheduling", uses advertisements on a specific channel to discover neighbors and initiate schedule negotiations. The second algorithm, "Reservation-based scheduling", augments the Aloha-based algorithm with a gossip mechanism that distributes the scheduling information to more nodes, speeding up the negotiation of a common schedule. In order to assess the performance of the scheduling algorithms, we present two metrics: "relative connectivity", a static metric which evaluates how many feasible neighbors from the physical connectivity graph have been added to the schedule; and "link duration", a dynamic metric that evaluates the lifetime of a link in the logical connectivity graph compared to its lifetime in the physical connectivity graph. We have evaluated the two algorithms in both a simulated environment and with a field experiment. Our field experiment features an interleaved implementation of the two algorithms, which allows us to compare their performance directly, without having to replicate the physical connectivity in separate experiments. By comparing the performance of the algorithms under different network density conditions, we can infer the importance of the different features of the two approaches, which gives insight into the design of future protocols.

The remainder of this chapter is organized as follows. Section 7.2 provides a comprehensive overview of MAC protocol approaches and standardization activities, and highlights the need for a distributed scheduling algorithm for TSCH. Section 7.3 then details the two scheduling algorithms proposed in this chapter, called "Aloha-based scheduling" and "reservation-based scheduling". A simulation environment is described in Section 7.4, and an implementation and field experiment described in Section 7.5. Performance of the two algorithms in simulated and real environments is explored in Section 7.6. Finally, Section 7.7 concludes this chapter and presents directions for future work.

7.2 Time Synchronized Channel Hopping

There are two main approaches for regulating access to a shared wireless medium: contention-based and reservation-based approaches. Any derived MAC protocol is based on one of those two approaches or a combination thereof.

Contention-based protocols are fairly simple, mainly because neither global synchronization nor topology knowledge is required. In a contention-based approach, nodes compete for the use of the wireless medium and only the winner of this competition is allowed to access to the channel and transmit. Aloha and *Carrier Sense Multiple Access* (CSMA) are canonical representative schemes of contention-based approaches. They do not rely on a central entity and are robust to node mobility, which makes them intuitively a good candidate for dynamic mobile networks.

Preamble sampling is a low-power version of contention-based medium access, widely popular in WSNs. All nodes in the network periodically sample the channel for a short amount of time (at most a few milliseconds) to check whether a transmission is ongoing.

Nodes do not need to be synchronized, but all use the same check interval. To ensure all neighbors are listening, a sender prepends a preamble which is at least as long as the check interval. Upon hearing the preamble, nodes keep listening for the data that follow. The optimal check interval, which minimizes the total energy expenditure, is a function of the average network degree and the load of the network. A check interval of 100 ms is typical. Numerous efforts have proposed ways to optimize the sampling [93], reducing the preamble length by packetization [17] or by synchronizing the nodes [134].

Despite their success, contention-based protocols suffer from degraded performance in terms of throughput when the traffic load increases. In addition, the uncoordinated nature of their resource allocation prevents them from achieving the same efficiency as ideal reservation-based protocols. Finally, frequency agility is hard to achieve by such protocols, as nodes are not synchronized.

Reservation-based protocols require the knowledge of the network topology to establish a schedule that allows each node to access the channel and communicate with other nodes. The schedule may have various goals such as ensuring fairness among nodes or reducing collisions by preventing nodes from transmitting at the same time. *Time Division Multiple Access* (TDMA) is a representative example for such a reservation-based approach.

In TDMA, time is divided into slots which are grouped into superframes which repeat over time. A schedule is used to indicate to each node when it has to transmit or receive, to/from which neighbor. Provided the schedule is correctly built, transmissions do not suffer from collisions, which guarantees finite and predictable scheduling delays and also increases the overall throughput in highly loaded networks.

Many approaches to MAC for WSNs combine some elements of contention-based protocols, especially for neighbor discovery or other startup tasks, with reservation-based scheduling for improved performance once neighbors are known. For example, in the PEDAMACS protocol [39], nodes transmit randomly using CSMA in order to discover the network topology and collect the topology information at a central node, which then computes all schedule information for all nodes in the network and distributes it. After this centralized schedule has been distributed, communication is governed by the schedule. In the TRAMA protocol [96], each TDMA superframe contains "random-access" frames, where neighbors are discovered and local topological information is shared, and "scheduled-access" frames, where nodes determine which of their two-hop neighbors has priority using a hash of frame number and node ID. In the Dozer protocol [18], new nodes use CSMA-like arbitration to respond to the beacon packets transmitted by nodes that have already joined the network; authority for setting the schedule is based on the tree hierarchy that emerges as "child" nodes associate with the older "parent" nodes. The SMACS protocol [104] uses a contention-based exchange of "invitation" and "response" packets to establish links between neighbors and to negotiate a transmit/receive schedule for that link for future communications. These four examples show the variety of approaches to scheduling that have been explored, from centralized (PEDAMACS) to purely decentralized (SMACS and TRAMA). The two algorithms presented in this chapter belong to the family of purely decentralized scheduling algorithms,

and are designed specifically for the scheduling requirements of TSCH networks, and in particular the challenges of scheduling on a mobile network with connectivity that changes frequently.

The reliability of a wireless link is mainly challenged by external interference and multi-path fading. Previous works [125, 124] show how channel hopping combats both of these, respectively. If a transmission fails, the sender retransmits the packet on a different frequency channel. Because this frequency change causes the wireless environment to be different, the retransmission has a higher probability of being successful than if it were retransmitted on the same channel.

Channel hopping was first applied to WSNs in a proprietary protocol called *Time Synchronized Mesh Protocol* (TSMP) [92]. In TSMP, nodes in the network are synchronized on a slotted time base. An individual timeslot is long enough for a sender to send a data frame, and for a receiver to acknowledge correct reception (a timeslot of 10 ms is common). L consecutive slots form a superframe, which repeats over time. A schedule of length L timeslots indicates, for each timeslot, whether the node is supposed to transmit or receive, to/from which neighbor and on which channel. TSMP runs on IEEE802.15.4-2006 [59] compliant radios, which offers 16 frequency channels in the 2.4GHz ISM band. A central scheduler is used to compute a schedule, which is then injected and used in the network.

TSMP makes a subtle distinction between channel and frequency. The former is used in the schedule: node A schedules a link to node B on a given timeslot, and a given channel. This means that every superframe, node A will have the opportunity to use that link. The latter is the frequency nodes A and B communicate on. Nodes use the *Absolute Slot Number* (ASN) to keep track of which timeslot they are in. It is an ever-increasing number which is incremented at each timeslot, and which is shared by all nodes in the network as part of the synchronization procedure. TSMP uses the following function to obtain the frequency used for transmission from the channel in the schedule and the ASN. % is the modulo operator; 16 indicates that there are 16 available channels.

$$\text{frequency} \;=\; (\text{channel} \;+\; \text{ASN}) \;\%\; 16$$

As a consequence, even when a link always appears at the same channel in the schedule, the operation described above ensures that communication happens in a channel hopping manner, thereby increasing the reliability of the link.

TSMP, which combines time synchronization and frequency agility, has been shown to achieve reliabilities of over 99.999% [34]. Its core idea has been standardized for industrial applications in WirelessHART [53, 51, 105] and ISA100.11a [61]. In 2009, was introduced in the draft standard IEEE802.15.4e under the name “Time Synchronized Channel Hopping”. The draft was finalized in 2012 as an amendment to the current IEEE802.15.4-2011 standard.

All of the above standards rely on a central controller to compute a schedule for the network to use. The goal of this chapter is to propose a distributed alternative, targeted at mobile nodes.

c	a channel
i, j, k, n	slot numbers
L	number of slots in a superframe
$S = \{S_0, S_1, \ldots, S_{L-1}\}$	state for each slot
$C = \{C_0, C_1, \ldots, C_{L-1}\}$	data channel for each slot
$N = \{N_0, N_1, \ldots, N_{L-1}\}$	neighbor for each slot (can be NULL)
$P = \{(r, c)_1, (r, c)_2, \ldots\}$	list of potential neighbors (contains identifier and channel)
$D = \{(r, c)_1, (r, c)_2, \ldots\}$	list of neighbors self is connected to

Table 7.1: Variables used in this chapter.

7.3 Distributed Scheduling Algorithms

7.3.1 Goal and Metrics

The goal of the proposed schedule is to achieve full connectivity, which is achieved when each node of the neighbor has established a bidirectional link to each of its physical neighbors. A bidirectional link is established between nodes A and B when, in the superframe, there is at least one slot scheduled from A to B, and one from B to A. The unreliability of the wireless link and the movement of the nodes are challenges the scheduling algorithm needs to cope with.

If a link is present in the physical graph, it is *feasible*; if a link is present in the physical but not in the logical graph, it is said to be *unscheduled*; a link which still appears in the logical graph after it has disappeared from the physical graph is called *stale*. We use the ratio between the scheduled and feasible links as a metric for the static goodness of the scheduling algorithm.

Node mobility causes links to come and go. A link therefore has a finite lifetime, or *link duration*. To take advantage of a link, the scheduling algorithm needs to establish a logical link as soon as the physical link appears, and unschedule it as soon as it disappears from the physical graph. We quantify the dynamic goodness of the scheduling algorithm by comparing the link duration between the physical and logical graphs.

Results presented in Section 7.4 are normalized against the optimal case, that is, the physical connectivity graph. The variables to be used in this chapter are listed in Table 7.1.

To be able to communicate, two nodes need to schedule a slot to one another. They hence need to communicate to agree which slot in the superframe to use, and which channel. We present two variants of the proposed scheduling mechanism. Aloha-based scheduling (Section 7.3.2) is a simple, canonical algorithm, in which neighbor nodes opportunistically discover each other and establish links. Reservation-based scheduling (Section 7.3.3) builds upon that. By adding an explicit reservation channel, nodes discover each other faster, which is desirable in the presence of mobile nodes.

7.3.2 Aloha-based scheduling

For each of the L slots in the superframe, the algorithm maintains a state S_i, a channel C_i, and a neighbor N_i. There are five states: "*Aloha*", "*Transmit Connection Request*", "*Receive Connection Request*", "*Transmit Data*", "*Receive Data*". A slot is assigned a channel C_i and a neighbor N_i only in the latter four states.

The *Aloha* state is the default. When establishing a unidirectional link from A to B, the scheduling algorithm causes a slot in A's schedule to transition from *Aloha* to *Transmit Connection Request*, to *Transmit Data*. Similarly, the same slot in B's schedule transitions from *Aloha* to *Receive Connection Request*, to *Receive Data*. When both A and B's slots are in the *Transmit Data* and *Receive Data* state, respectively, data packets can be transmitted from A to B, once per superframe if exactly one slot is scheduled in the superframe. While communicating, A monitors whether its data packets are acknowledged; B monitors whether it receives data at all. If for five consecutive superframes no data are successfully transmitted, the slot returns to the *Aloha* state; connection is then lost. To ensure these statistics are up to date, if a sender has no data to send on a given slot, it sends an empty "keep-alive" message.

Three types of packets move through the network:

- **Advertisement** packets contain a list of *Receive Connection Request* slots of the sender node. This can be used by neighbors to know where it can be reached to establish a link. Each entry is a tuple (s, c) of slot and associated channel. Advertisements are broadcast and always exchanged on channel 0.

- **Connection Request** packets are sent in response to Advertisements; they are unicast on one of the slots announced in the Advertisement (at the announced channel, always different from channel 0).

- **Data** packets flow over the slot when a link is established. Their content is determined by the application, but their successful transmission is monitored by the scheduling algorithm to detect stale links. An empty data packet is used as a keep-alive. Data packets are always sent on a channel different from channel 0.

Note that there are L slots in a superframe, each of which can be used for an independent link. That is, a independent state machine is running for each slot. IEEE802.15.4-2006 compliant radios can transmit on 16 independent frequency channels. We dedicate channel 0 exclusively to Advertisements, and channels 1-15 exclusively to Connection Requests and Data packets.

Pseudocode listings for the two proposed algorithms are given below. The Aloha-based algorithm is described in Algorithm 4. The reservation-based algorithm has different behaviors during time slot 0 and other slots; its slot 0 behavior is given in Algorithm 5, while the behavior at other times is given in Algorithm 6.

Algorithm 4 presents Aloha-based scheduling in pseudo-code. It is executed by every node in the network. Upon startup (lines 1–5), all the slots are set to the *Aloha* state. The main loop (lines 6–51) iterates at each slot; different actions are taken according to the state of the slot. When in an *Aloha* slot, a node listens for Advertisements 90% of the time (on channel 0, lines 17–26), while 10% of the time it transmits an Advertisement (lines 10–15).

Sending Advertisement packets. The idea of sending an Advertisement is for a node to announce different rendezvous slot/channel tuples so that interested nodes can establish a link to it. When sending an Advertisement, a node converts all of its *Aloha* slots to the *Receive Connection Request* state, and assigns each of those a random channel other than channel 0 (lines 10–14). It puts that list in an Advertisement which it sends on channel 0. It then waits to be contacted on one of the *Receive Connection Request* slots it just announced.

Receiving Connection Request packets. When reaching a slot in the *Receive Connection Request* state (lines 29–36), a node listens to the channel it has previously randomly picked and announced in its Advertisement. If it does not receive anything (line 35), it converts that slot back to *Aloha* state. If it does receive a Connection Request (lines 31–33), it converts that slot to *Receive Data* state and records the identifier of the requester.

Receiving Advertisement packets. When receiving an Advertisement (lines 19–25), a node learns about the presence of a neighbor and is given the opportunity to contact it. If it has no slot scheduled to that neighbor, it picks one of the slots announced in the Advertisement where itself is in the *Aloha* state, that is, it picks a rendezvous slot and channel. In case there are multiple slots which satisfy these requirements, it picks one of them randomly. It changes the state of that slot in its scheulde to *Transmit Connection Request* (line 22), records the channel announced in the Advertisement (line 23), and the sender of that packet (line 24).

Transmitting Connection Request packets. When reaching a slot i in the *Transmit Connection Request* state (lines 38–44) a node sends a Connection Request to the neighbor recorded in N_i, at the channel recorded in C_i (line 38). If it receives an acknowledgment, it puts that slot in the *Transmit Data* state, and the logical link is established. If the Connection Request is not established (e.g. due to a collision or nodes moving apart), the slot is reset to the *Aloha* state.

7.3.3 Reservation-based scheduling

The reservation-based scheduling protocol behaves like the Aloha-based protocol, with the following additions:

- Slot 0 is a permanent rendezvous slot, that is, only Advertisements can be exchanged. Unlike other slots, Advertisements can be exchanged on any of the 16 available channels, in slot 0. Each node picks a channel on which it listens for Advertisements. Using slot 0 as a reservation slot gives nodes more opportunities to establish links to one another.

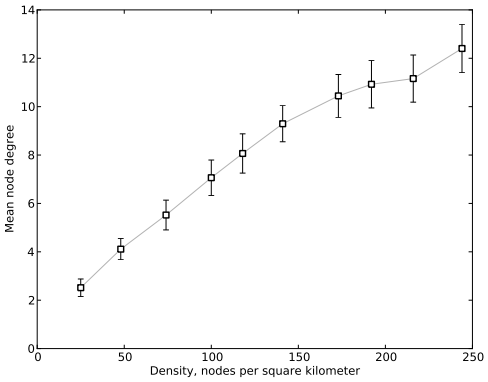

Figure 7.1: Mean node degree vs density of nodes in simulated environment.

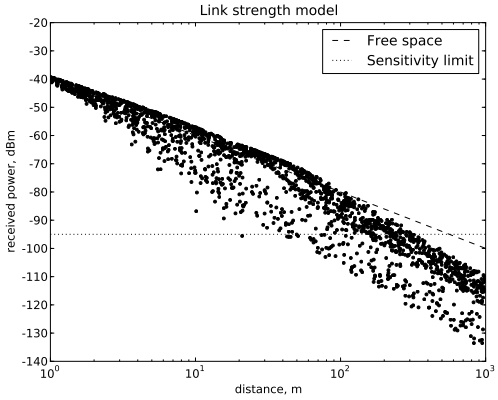

Figure 7.2: Received power from randomly chosen locations in simulated environment.

- In their Advertisements, nodes also include the list of the neighbors they are connected to, and the channel those neighbors are listening on in slot 0. This means that nodes learn about their two-hop neighbors.
- Each node maintains a list P of potential neighbors and the channel they are listening on in slot 0. This information is obtained by listening to Advertisements. Each node also maintains a list D of neighbors it is currently connected to. The scheduling algorithm tries to get as many nodes from P to D.
- A node only announces the *even* slots in its Advertisement. When the state of even slot i becomes *Transmit data* (resp. *Receive data*), the state of odd slot $i + 1$ is implicitly changed to *Receive data* (resp. *Transmit data*). This means that links are scheduled in pairs, one in each direction, establishing only bidirectional links.

Algorithms 5 and 6 present reservation-based scheduling in pseudo-code. Algorithm 5 contains initialization, the main loop, and the behavior for slot 0, while Algorithm 6 contains the behavior for all other slots.

7.4 Simulation environment

We use a Python-based simulator to model the mobility and RF propagation characteristics for a fleet of 25 mobile nodes. The superframe size was chosen to be 17 slots. The size must be co-prime with 16 in order to gain the benefits of the channel offset scheme; a relatively small superframe size was chosen to ensure that the scheduling constraints would be significant.

Algorithm 4 Aloha-based scheduling.

for each slot i in $0 \dots L-1$ **do**
    $S[i]$ =*Aloha*; $N[i]$ =NULL; $C[i]$ =NULL
end for
loop
    Go to the next slot i
    if $S[i]$ ==*Aloha* **then**
        if uniform(0,1)< 0.1 **then**
            Find the set $\{j\}$ of all other slots with state $S[j]$ ==*Aloha*
            for each of these slots **do**
                $S[j]$=*Receive Connection Request*
                $C[j]$ =uniform_integer(1,15)
            end for
            Send Advertisement listing these slots and their channels $\{(j, C[j])\}$, on channel 0
        else
            Listen for an Advertisement on channel 0
            if Advertisement $\{(j, C[j])\}$ received **then**
                Find own set of slots $\{k\}$ which are of state $S[k]$ ==*Aloha*
                if $\{k\} \cap \{j\}$ is not empty **then**
                    Choose common slot n in $\{k\} \cap \{j\}$ randomly
                    $S[n]$ =*Transmit Connection Request*
                    $C[n]$ set to the receiving channel for that slot, read from Advertisement
                    $N[n]$ set to the node that sent the Advertisement
                end if
            end if
        end if
    else if $S[i]$ ==*Receive Connection Request* **then**
        Listen for a Connection Request to self on channel $C[i]$
        if valid Connection Request received **then**
            Send Acknowledgment
            $S[i]$ =*Receive Data*
            $N[i]$ set to the ID of the requesting node
        else
            $S[i]$ =*Aloha*
        end if
    else if $S[i]$ ==*Transmit Connection Request* **then**
        Send Connection Request on channel $C[i]$ to node $N[i]$
        if Acknowledgment received **then**
            $S[i]$ =*Transmit Data*
        else
            $S[i]$ =*Aloha*; $N[i]$ =NULL
        end if
    else if $S[i]$ ==*Receive Data* **or** $S[i]$ ==*Transmit Data* **then**
        if no successful communication for 5 consecutive superframes **then**
            $S[i]$ =*Aloha*; $N[i]$ =NULL
        end if
    end if
end loop

Algorithm 5 Reservation-based scheduling: Initialization and Slot 0 behavior.

```
1: for each slot i in 0...L-1 do
2:     S[i] =Aloha
3:     N[i] =NULL
4: end for
5: C[0] =uniform_integer(0,15)
6: P = {}
7: D = {}
8: loop
9:     Go to the next slot i
10:    if i == 0 then
11:        if P is not empty and uniform(0,1)< 0.1 then
12:            Choose (j, c) randomly from neighbors of interest in P
13:            Transmit Advertisement to node j on channel c
14:            if Acknowledgment received then
15:                set state of all advertised slots to S[k] =Receive Connection Request
16:            end if
17:        else
18:            Listen for an Advertisement on channel C[0]
19:            if Advertisement received then
20:                Send Acknowledgment
21:                If neighbor of interest, choose common slot n (similar to Algorithm 4)
22:                S[n] =Transmit Connection Request
23:                N[n] = the ID of the node that sent the Advertisement
24:                C[n] = the receiving channel for that slot in the Advertisement
25:            end if
26:        end if
27:    else
28:        execute Algorithm 6
29:    end if
30: end loop
```

7.4.1 Propagation Model

The design objective for the RF propagation model is to create a deterministic model which captures the variance of the distance-to-received-power relationship observed in empirical studies of static spatial configurations [136], while also providing plausible spatial correlation of link strength. Approximately 30% of the simulated environment is covered with obstacles. The radiated power from a transmitting antenna is attenuated by an inverse square law as it moves through "obstacle-free" space, but is attenuated by an inverse fourth power law as it moves through "obstacle" space. This "higher power attenuation" scheme is inspired by empirical models of the effect of foliage on line-of-sight transmission [79]. The foliage model and density of obstacles is intended to represent an outdoor estuarial environment similar to that encountered by the Floating Sensor Network project. The multipath effect of the signal reflecting off the ground is modeled. The reflection is assumed to result

Algorithm 6 Reservation-based scheduling, behavior for slots other than 0.

```
1: if S[i] ==Aloha then
2:     if uniform(0,1)< 0.1 then
3:         Find the set {j} of all other even slots with S[j] ==Aloha
4:         for each j do
5:             S[j]=Receive Connection Request
6:             C[j] =uniform_integer(1,15)
7:         end for
8:         Send Advertisement listing these slots channels {(j, C[j])} and all tuples in D on channel 0
9:     else
10:        Listen for an Advertisement on channel 0
11:        if Advertisement {(j, C[j])} received then
12:            Add new possible neighbors to P using the information in the Advertisement
13:            Find own set of slots {k} with state S[k] ==Aloha
14:            if {k} ∩ {j} is not empty then
15:                Choose common slot n in {k} ∩ {j} randomly
16:                S[n] =Transmit Connection Request
17:                N[n] = the ID of the node that sent the Advertisement
18:                C[n] = the receiving channel for that slot in the Advertisement
19:            end if
20:        end if
21:    end if
22: else if S[i] ==Receive Connection Request then
23:    Listen for a Connection Request for self on channel C[i]
24:    if valid Connection Request received then
25:        Send Acknowledgment
26:        S[i] =Receive Data; S[i + 1] =Transmit Data
27:        N[i] and N[i + 1] set to the ID of the requesting node
28:    else
29:        S[i] =Aloha
30:    end if
31: else if S[i] ==Transmit Connection Request then
32:    Send Connection Request on channel C[i] to node N[i]
33:    if Acknowledgment received then
34:        S[i] =Transmit Data; S[i + 1] =Receive Data
35:        Put (N[i], C[i]) in D
36:        Remove N[i] from P if present
37:    else
38:        S[i] =Aloha
39:    end if
40: else if S[i] ==Receive Data or S[i] ==Transmit Data then
41:    if no successful communication for 5 consecutive superframes then
42:        S[i] =Aloha
43:        move N[i] from D to P
44:        N[i] =NULL
45:    end if
46: end if
```

in a 180° phase change and no attenuation.

The size of the simulated enviroment is modified as needed to yield desired node densities. The minimum and maximum densities are 25 and 250 nodes per square kilometer. Figure 7.1 shows the mean node degree (number of neighbors in the physical connectivity graph) for the different simulated densities. The bars represent the 95% certainty interval for the estimate of the mean.

7.4.2 Co-channel interference model

The interfering effects of two nodes transmitting on the same channel at the same time (usually called a "collision") is one of the main constraints on the decentralized schedule.

The IEEE802.15.4-2006 standard specifies required jamming resistance for interference coming from an adjacent channel (1 channel away) or an alternate channel (2 channels away), but does not specify a required resistance to interference on the same channel. The Texas Instruments CC2420 2.4 GHz IEEE802.15.4-2006 compliant transceiver [117] has a specified co-channel rejection of -3 dB; in other words, if node A receives a transmission from node B with p dBm power, and a simultaneous transmission from node C with $(p-3)$ dBm power, the transmission for B will be received correctly and the transmission from C rejected. We use this model for our simulation. Adjacent and alternate channel interference are not modeled in this simulation.

7.4.3 Node Mobility Model

Each node is modeled as a mobile device moving at a constant speed in the environment described above. The speed of each node is drawn from a uniform distribution over $[0.8, 1.2]$ m/s. Each node transmits at 0 dBm (1 mW) using an isotropic antenna. The height of the antenna from the ground (used for the multipath calculations) is drawn from a uniform distribution over $[0.7, 1.3]$ m for each node. Node motion is controlled by a random waypoint procedure: nodes select a cardinal direction randomly, then a distance to move in that direction. When they reach their destination, they repeat the selection process. The nodes are confined to a square area with dimensions determined by the desired node density.

Figure 7.2 shows the received power for randomly located transmitter and receiver nodes in the simulated environment.

7.5 Experimental Setup

On November 19 2010, an implementation of the TSCH algorithms presented in Section 7.3 was tested using ten Berkeley FSN drifters in the Grant Line Canal near Tracy, California.

Frame	Function
0–2	ASN synchronization
3 – 58	Physical graph discovery
63 – 79	Aloha algorithm
82 – 98	Reservation algorithm

Table 7.2: Superframe structure.

The algorithms were implemented on Texas Instruments eZ430-RF2500 platforms, which consist of an MSP430 16-bit 16-MHz micro-controller and a CC2500 radio chip. The radio chip was programmed to communicate on the frequencies of the IEEE802.15.4-2006 standard, on the 2.4GHz frequency band.

Each drifter was equipped with an eZ430-RF2500 platform. Distributed synchronization of these nodes was facilitated by a *pulse per second* (PPS) signal generated by the GPS unit on board the drifter, which provides a 1 Hz synchronization pulse with 25 ns jitter. The memory footprint of the implemented algorithms is 6 kB of flash memory and 500 B of RAM memory[1].

Both synchronization algorithms as well as a physical connectivity discovery mechanism were executed concurrently by the nodes by using a "master" superframe of 100 frames, and scheduling various operations within that framework, as shown in Table 7.2. The idea is to gather baseline physical connectivity data and to run both algorithms simultaneously to allow for fair comparison of their performance. Each slot is 10 ms long; the superframe repeats every second.

- The physical graph discovery phase consists of each node deterministically broadcasting on each channel in sequence (i.e. there are no collisions). When not transmitting, a node listens for its peers and record which node was heard, on what slot, and on what frequency channel. Because there are 10 drifters and 16 channels, it takes 160 physical graph discovery slots to completely survey the connectivity. With 56 slots per superframe dedicated to physical discovery, we obtain a full image of the physical connectivity every 3 superframes, that is, every 3 seconds.

- During the Aloha algorithm phase, the nodes execute the scheduling algorithm presented in Section 7.3.2. During the reservation algorithm phase, the nodes execute the scheduling algorithm presented in Section 7.3.3. These algorithms are executed independently from each other and from the physical graph discovery phase. As in the simulation-based study, both phases are 17 slots long.

We use the results of the physical graph discovery as an estimate of the instantaneous connectivity in order to evaluate the algorithmic performance of the Aloha and reservation algorithms.

[1]The firmware source code is available at `http://wsn.eecs.berkeley.edu/svn/ezwsn/`

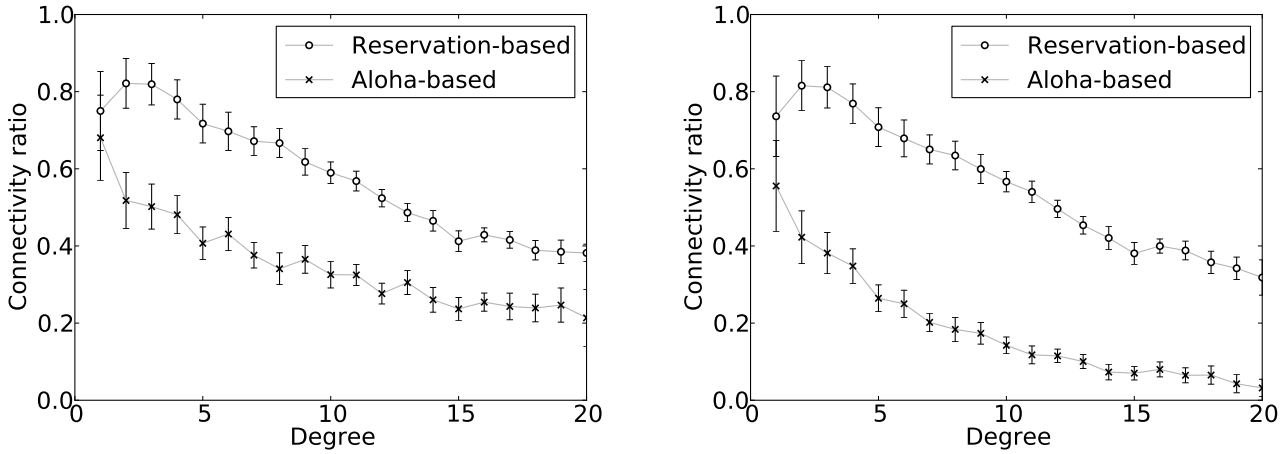

Figure 7.3: Unidirectional (left) and bidirectional (right) mean connectivity ratio vs degree in simulated environment.

The results of the discovery phase, and the state/neighbor/channel tables from each TSCH algorithm, were output from the eZ430-RF2500 motes and recorded using the data logging capabilities of the FSN drifter. Seven hours of data were recorded, resulting in over 250000 records of connectivity and algorithm state[2].

The Berkeley FSN drifters acted as passive floating sensors, being carried by the water current at approximately 0.3 m/s. Variations in the channel velocity profile caused their relative positions to change during the experiment. Overall, connectivity was not as highly dynamic as the simulation environment.

7.6 Results

7.6.1 Static Metric: Relative Connectivity

The static connectivity test in the simulated environment proceeds as follows:

1. Simulate 25 mobile nodes for 60 seconds;
2. pick a node and a superframe at random;
3. from the physical connectivity graph, count the number of unique edges incident to that node over the superframe (that is, the number of one-hop neighbors connected for at least 1 slot during the superframe); this is the degree of the node;

[2]The gathered traces are made freely available at `http://wsn.eecs.berkeley.edu/connectivity/`

4. from the logical connectivity graph, find the number of outbound edges (for the unidirectional test), or find the number of neighbors with both an outbound and inbound edge (the bidirectional test);

5. the ratio of the logical connection count to the node degree is the *connectivity ratio* for the node. A connectivity ration of 0.8 indicates that a logical link is present 80% of the cases a physical link is. A connectivity ratio of 1 is the best possible case.

To process the experimental results, the procedure was similar: a node and superframe were picked at random from the experimental logs, and the calculation of the connectivity ratio proceeded as in the simulation case.

Figure 7.3 shows the mean connectivity ratio versus the node degree for 1250 simulations, for both unidirectional and bidirectional connections. The bars represent the 95% confidence interval in the estimate of the mean.

In simulation, the reservation-based algorithm outperforms the Aloha-based algorithm at almost all node degrees (the confidence intervals overlap for degree 1). The reservation-based algorithm has more resources allocated to neighbor discovery, and a successful advertisement/connection request exchange results in a bidirectional connection. For both algorithms, increased node degree results in decreased relative connectivity ratios. More local nodes mean more collisions between Aloha advertisements, which reduces the effectiveness of neighbor discovery, and more cases of multiple nodes responding to an advertisement, resulting in collisions and lost connectivity. The superframes also fill up when more neighbors are present; since the superframe size is 17 slots, a node cannot have bidirectional links with more than eight neighbors. The difference between the Aloha-based and reservation-based algorithm performance at high node degrees, however, demonstrates that both collisions and saturation must be significant.

In the experimental results, shown in Figure 7.4, a different relationship between the Aloha and reservation performance is observed. For the unidirectional case, the reservation algorithm dominates at lower network degrees, as in the simulation results, but under more connected conditions, the reservation algorithm performance suffers. This phenomenon is not well explained by the analysis applied to the simulation results. In the bidirectional case, we see a change in the performance of both algorithms at different network densities, but the results are too close to judge that one algorithm is outperforming the other. In both cases, the overall trend (higher density leading to lower connectivity ratio) is consistent with the simulation results. The regime where the Aloha algorithm outperforms the reservation algorithm in the unidirectional case remains unexplained.

7.6.2 Dynamic Metric: Link Durations

The dynamic link duration test proceeds as follows:

1. Simulate 25 nodes for 60 seconds;

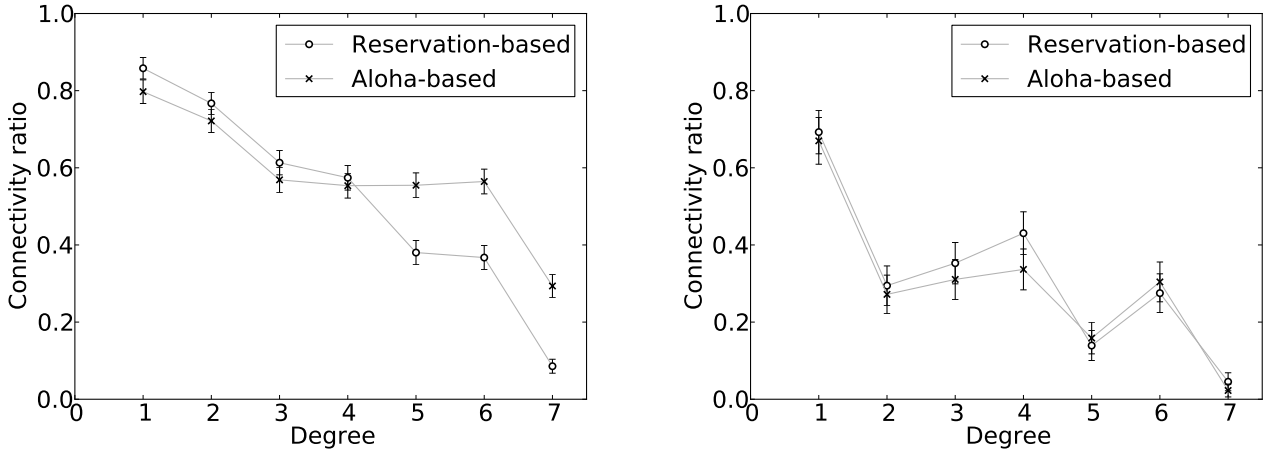

Figure 7.4: Experimental unidirectional (left) and bidirectional (right) mean connectivity ratio vs degree.

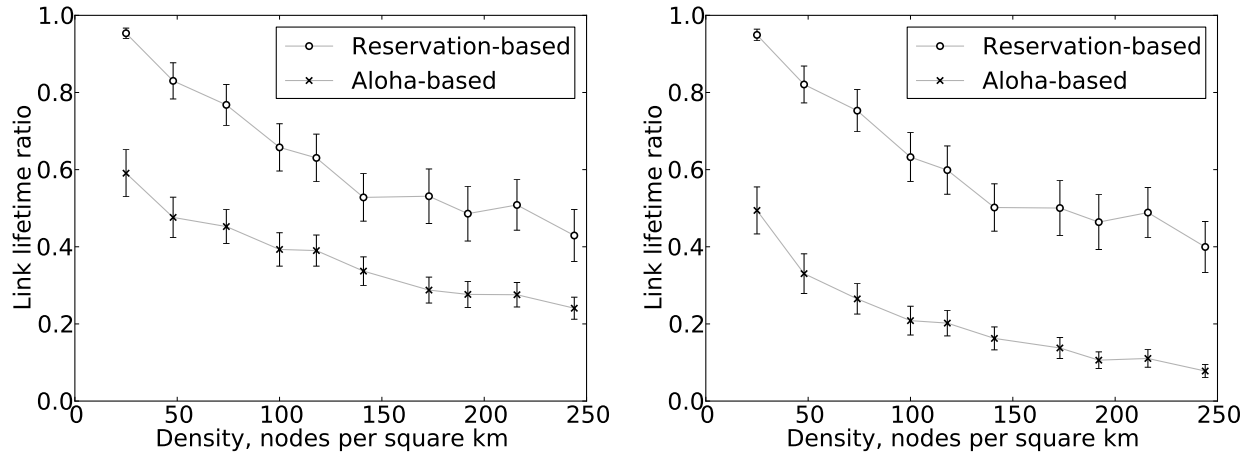

Figure 7.5: Unidirectional (left) and bidirectional (right) mean link lifetime ratio versus density in simulated environment.

2. pick a node and a superframe at random;

3. pick one of the edges on the physical connectivity graph incident to that node at random; this is the link we will test;

4. count the number of consecutive superframes (forward and backward in time) in which this link is in the physical connectivity graph; this is the *physical link duration*;

5. count the number of superframes in which the link exists in the logical connectivity graph, either as a unidirectional link (the original node to the destination) or as a

bidirectional link; this is the *logical link duration*;

6. the ratio of the logical link duration to the physical link duration is the *link lifetime ratio*. A link lifetime ratio of 0.8 indicates that the algorithm has scheduled a logical link 80% of the time a physical link is present. That is, if two nodes are within radio range for 10 s, they can exchange data for 8 s. A link lifetime ratio of 1 is the best possible case.

For the experimental results, the procedure is the same, except that the random node and superframe are drawn from the experimental logs.

Figure 7.5 shows the mean link lifetime ratio versus the density of the nodes in the simulated environment for 1250 simulations. The bars represent the 95% confidence interval for the estimate of the mean. The degree of the node is not well defined over many superframes, as the physical and logical connectivity change. While the static connectivity test could use the node degree as the independent variable, for the dynamic link duration test we use the node density as a surrogate. See Figure 7.1 for the relationship between the mean node degree and node density.

The dynamic performance in simulation also shows that the reservation-based algorithm outperforms the Aloha-based algorithm. Again, the Aloha-based algorithm is at a disadvantage, because its advertisement/connection request transactions build unidirectional links, not bidirectional links. At low densities, the ratio between the algorithms' performances is roughly 2, which suggests that the unidirectional/bidirectional allocation difference dominates in this regime. But at higher densities, the difference between the two algorithms widens, which means other effects must be significant as well.

The saturation effects at work in the connectivity tests are also significant in the dynamic case. Links can be broken by co-channel interference, if another pair of nodes begins transmitting at the same channel/slot as an existing link. Nodes that have many active links also have less vacant slots available to form new links. Saturation effects alone cannot explain the decreased performance at high density, however, since the Aloha-based algorithm's performance decreases significantly more than the reservation-based algorithm's performance.

The reservation-based algorithm benefits when advertisements are exchanged frequently, because information about connected neighbors is carried by the advertisement packets. The reactivity of the reservation-based algorithm therefore increases at higher densities, as nodes learn about possible new neighbors more quickly. Because the advertisements in the reservation-based algorithm carry more information than the advertisements in the Aloha-based protocol, the reservation-based algorithm gains relative performance at higher node densities.

Although the dynamic link survival time test can be applied to the experimental data, the experiment was conducted at essentially a single density condition. We therefore do not have values of the dynamic test at different densities, and cannot explore the density-link time relationship as in Figure 7.5. The results of the dynamic link survival time test are summarized in Table 7.3.

Algorithm	Link type	Survival time ratio and 95% Confidence Interval
Aloha	unidirectional	0.53 ± 0.17
Reservation	unidirectional	0.42 ± 0.16
Aloha	bidirectional	0.57 ± 0.19
Reservation	bidirectional	0.61 ± 0.17

Table 7.3: Experimental dynamic link time results.

The dynamic lifetime test shows strong performance for both algorithms, under either the unidirectional or bidirectional case, with no statistically significant difference in the mean lifetime ratios. Although the value of the mean lifetime ratio is consistent with those observed in Figure 7.5, having both the Aloha and reservation algorithms perform (practically) identically is inconsistent with our observations in the simulated system. A major difference between the two scenarios is the distribution of link lifetimes in the physical connectivity graph. In the simulated environment, the connectivity is highly dynamic, and the short simulation time (60 seconds) places an upper bound on the link lifetime. In the experimental setup, link lifetimes ranged from as short as 1 s to several hours long. When the connectivity is not as dynamic, the increased reactivity of the reservation-based algorithm is not an advantage, and the algorithms have similar performance.

7.7 Conclusions and Future Work

In this chapter, we present what is, to our knowledge, the first scheduling algorithm for Time Synchronized Channel Hopping networks which both is distributed and which copes with mobile networks. The two variant algorithms are based on an advertisement and rendezvous scheme: nodes continuously advertise their presence to allow neighbor nodes to discover and contact one another. An inactivity threshold mechanism is used to tear down previously established links.

The algorithms are tuned for a network of 25 drifter nodes randomly moving inside a lake or river. Simulation results show, under realistic propagation and mobility models, the efficiency of the algorithms. Figures 7.3 and 7.5 support the conclusion that the reservation-based algorithm outperforms the Aloha-based algorithm in practically all density conditions. Experimental results (Figure 7.4 and Table 7.3) do not show a significant advantage to one algorithm or the other; the maor difference between the experimental set-up and the sumulated system was the rate at which links formed and dropped, which suggests that in an environment with highly dynamic connectivity, including networks of mobile nodes, devoting additional resources to neighbor discovery and coordination pays off.

The goal of the scheduling algorithms presented in this chapter is to establish two-way connections between neighbor nodes, subject to the constraints of the superframe structure and the physical connectivity. We did not make assumptions about what kind of data is

sent over the links; the latency, throughput, and reliability requirements are not specified. These scheduling algorithms could be adapted to meet either predetermined or dynamic provisioning requirements. For example, a pair of nodes that need to exchange a large amount of data might wish to schedule more than one transmission slot per superframe.

Many WSN applications are highly energy-constrained. Our scheduling algorithms, as described here, require the radios to constantly either receive or transmit. This may consume too much power for some applications. An obvious modification is to reduce the duty cycle of the Aloha coordination activities; the algorithms could be implemented exactly as written, while only performing Aloha listen/transmit actions on a subset of the idle slots. The obvious trade-off is between the energy consumed for Aloha coordination versus the reactivity of the network to changes in the physical connectivity graph. Further work will focus on characterizing the rate of change of the connectivity graph, and determining a method for balancing power consumption and reactivity. Comparing the performance of these algorithms to previously proposed algorithms like TRAMA will also yield insight into the trade-offs made when designing algorithms for static versus mobile connectivity.

Chapter 8

Field Deployments of Heterogeneous Fleets of Active and Passive Drifters

Chapters 3 through 7 have explored specific components of the hydrodynamic state estimation system outlined in Section 2.1. This chapter presents a large-scale field experiment which demonstrated and validated these components: the design of the active and passive drifters, the actuation and control that satisfied navigation and safety goals, the communication systems to transmit data from the field, and the data assimilation techniques to process and present this information. This experiment represents the capstone of the Floating Sensor Network development process for the Generation 3 and Android drifters.

On May 9, 2012, the Floating Sensor Network team deployed 28 active Generation 3 drifters and 68 passive Android drifters in the Sacramento River near its junction with the Georgiana Slough, near the town of Walnut Grove, California. The operation demonstrated the communication, obstacle avoidance, navigation, and data-gathering capabilities of the Floating Sensor Network fleet, and gathered flow data for use in demonstrations of an online Ensemble Kalman Filter based assimilation using a high-performance computing cluster. A smaller-scale pilot experiment was conducted earlier, on April 12, 2012 (28 days previously, with roughly similar tidal conditions). This chapter describes the experimental method, gives an overview of the server and assimilation infrastructure, and presents the results of fleet movement analysis and preliminary hydrodynamics assimilation results.

8.1 Floating Sensor Network Fleet

The design and construction of the Gen 3 active drifters was described in detail in Chapter 3. An overview of the Android drifter design was provided in Section 3.4. For the most part, the drifters as deployed during these field experiments were standard implementations of the designs described in Chapter 3. Two important aspects specific to these operations are the communications infrastructure and the active control scheme.

Figure 8.1: Drifter fleet on the Walnut Grove Public Dock on May 9, 2012 prior to deployment. Photo credit: Jérôme Thai.

8.1.1 Communications Infrastructure

Figure 8.2 shows the communication links between various elements of the system. Data collected by the active and passive drifters is communicated back to the database server using the *General Packet Radio Service* (GPRS) of GSM. The Android smartphone on board each passive sensor provides the necessary GPRS functionality. As described in Section 3.3.9, our original design for the active drifters included two communication modules: a Motorola G24 OEM GSM module for direct communication with the server, and a Digi XBee-PRO 802.15.4 module for short-range communication with other drifters and the field team. Reliability issues prevented us from using the G24 GSM module, however, and so the active drifters communicate solely through the XBee module.

In order to bridge between the 802.15.4 short-range networking and the database servers, we built 10 specialized Android drifters carrying a XBee-PRO module as well as an Android smartphone. These devices, called "Relays", were put in static locations around the experimental environment. They did not gather data themselves, but simply collected the data

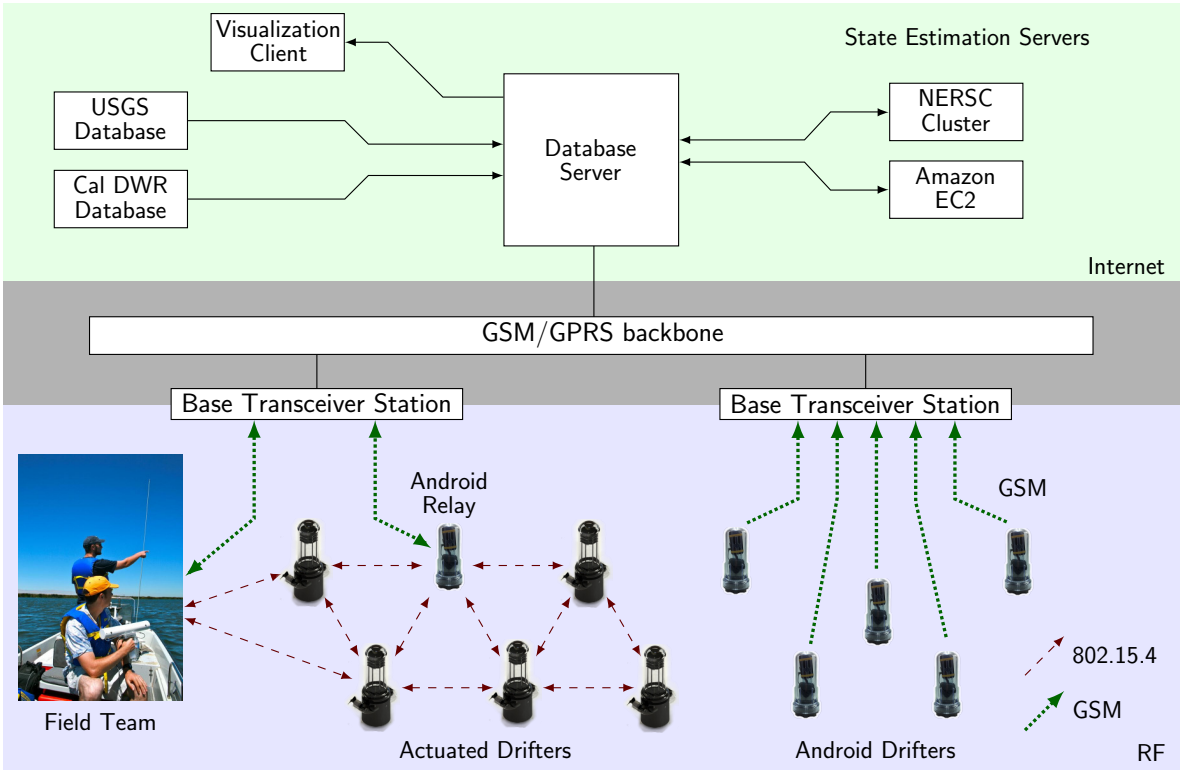

Figure 8.2: Communication architecture, showing the flow of data from drifters in the field to the database server and computation servers via the GSM service.

from the active drifters and transmitted it to the database server via GSM.

Field teams carried laptop computers with GSM modules and XBee-PRO modules. The active drifters can be sent commands via their XBee modules for diagnosis and troubleshooting. Capabilities include enabling and disabling the motors, running or terminating processes on the Gumstix, and querying various values like mission state or sensor readings. These commands can be send directly over the XBee link, or can be sent indirectly over GSM through the database server and the Android relays. This command capability is for development and debugging purposes. During the April 12 and May 9 experiments, no commands were sent to the active drifters; they operated autonomously.

8.1.2 Active Control Scheme

The Gen 3 active drifters used the reachability-based safety and navigation control scheme described in Chapter 5. By using different policy files on different drifters as described in Section 5.4.3, half the fleet was assigned to travel down the Sacramento River, while half the fleet was assigned to travel down the Georgiana Slough.

8.2 Assimilation Method and Back-end Infrastructure

The Ensemble Kalman Filter [42, 43] is a sequential data assimilation method related to the Kalman Filter and Extended Kalman Filter. The estimated state of a system, and the covariance of that state estimate, is tracked as an *ensemble* of individual states. Each member of the ensemble is individually propagated through a forward model of the underlying system, and corrected with new measurement data as it arrives. Figure 8.3 compares the estimation cycle of the Ensemble Kalman Filter with the classical Kalman Filter.

A summary of the Ensemble Kalman Filter algorithm follows.

1. Initialization: Generate an ensemble of N states, $\left\{\xi_{0|0}^{(i)}\right\}$, representing the initial estimate of the system.

2. Time update:

$$\xi_{k|k-1}^{(i)} = F\left(\xi_{k-1|k-1}^{(i)}\right) + w_{k-1}^{(i)}$$

$$\theta_{k|k-1} = \frac{1}{N}\sum_{i=1}^{N} \xi_{k|k-1}^{(i)}$$

$$E_{k|k-1} = \begin{bmatrix} | & & | \\ \xi_{k|k-1}^{(1)} - \theta_{k|k-1} & \cdots & \xi_{k|k-1}^{(N)} - \theta_{k|k-1} \\ | & & | \end{bmatrix}$$

   where F is the forward system model, $\left\{w_{k-1}^{(i)}\right\}$ are samples from the model noise process, $\theta_{k|k-1}$ is the pre-measurement estimate at time step k, and $E_{k|k-1}$ is the matrix of deviations from the mean.

3. Measurement update:

$$\Gamma_{k|k-1} = \frac{1}{N-1} E_{k|k-1} E_{k|k-1}^T$$

$$K_k = \Gamma_{k|k-1} H_k^T \left(H_k \Gamma_{k|k-1} H_k^T + R_k\right)^{-1}$$

$$\xi_{k|k}^{(i)} = \xi_{k|k-1}^{(i)} + K_k\left(y_k - H_k \xi_{k|k-1}^{(i)} + \epsilon_k^{(i)}\right)$$

   where $\Gamma_{k|k-1}$ is the covariance of the pre-measurement estimate ensemble, K_k is the Kalman gain for time step k, H_k is the observation operator (mapping each element of the observation vector to the state estimate), R_k is the covariance of the observation errors, y_k is the observation vector at time k, $\left\{\epsilon_{k-1}^{(i)}\right\}$ are samples from the observation noise process, and $\left\{\xi_{k|k}^{(i)}\right\}$ is the ensemble representing the post-measurement estimate at time step k.

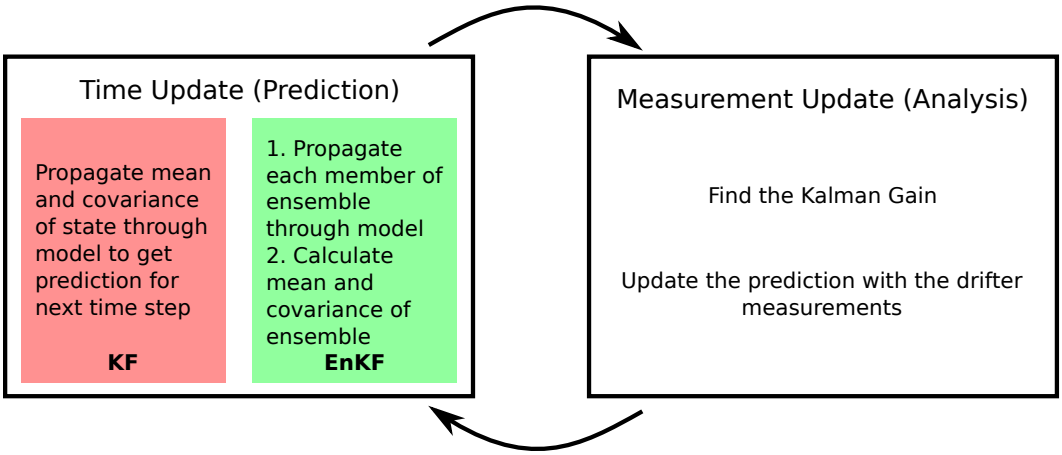

Figure 8.3: Ensemble Kalman Filter estimation scheme compared to Kalman Filter.

In this implementation, the forward model of the system is the REALM hydrodynamic model [3]. Simulating the forward evolution of each member of the ensemble represents a significant computational load. Fortunately, like many Monte Carlo computations, this phase of the algorithm is "embarrassingly parallel". Each forward simulation has no dependencies on the other members of the ensemble, and so each forward simulation can be delegated to a node of a computational cluster with very little data interchange or communication.

The *National Energy Research Scientific Computing Center* (NERSC), located at Lawrence Berkeley National Laboratory, maintains and operates Carver, a high-performance computing cluster of 1202 nodes, each consisting of 8 Intel Nehalem processors with 4 GB of memory [76]. An overview of the data flow from the drifters in the field to the Carver computational nodes is given in Figure 8.4. The drifters make TCP connections to a single FSN computer, which aggregates their incoming sensor data and sends it to a Postgres database within Carver. The database acts as a central repository for gathered data and assimilation results. In addition to the Lagrangian data collected by the drifters, relevant data from USGS and California *Department of Water Resources* (DWR) sensor stations is collected and stored.

As mentioned earlier, the individual forward simulations are executed on individual cores of the Carver cluster. The rest of the algorithm is executed on a single "Server" node. Results from the assimilation process are stored on the same database server, and queried by the visualization application, which can be accessed on the Web by any browser.

8.3 Experimental Procedure

Figure 8.5 shows the spatial domain of the experiment. April 12 was a pilot deployment for the major May 9 experiment. Tidal conditions for the two runs were similar. Figure 8.6

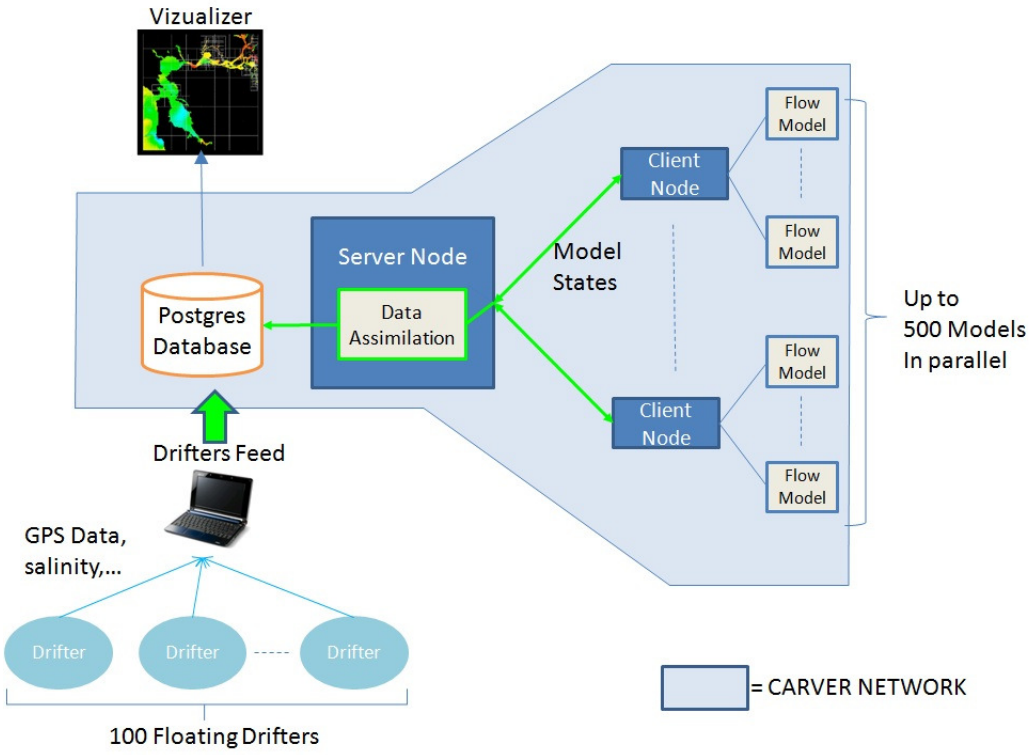

Figure 8.4: Dataflow from drifters in the field to the Carver cluster at NERSC.

show the conditions as measured at a USGS field station in the region, demonstrating that the tidal conditions on the two days were similar (as is to be expected, because they were 28 days apart). The flow was approximately 0.46 m/s (1.5 ft/s) in the outgoing (from northeast to southwest) direction. This is the non-inverted tidal condition. The original plan was to deploy all the drifters from the Walnut Grove Public Dock (label A in Figure 8.5), allow them to propagate through the junction, retrieve them at downstream points B and C, then recycle them at point D and E while time permitted. Unfortunately, on May 9 there was a significant underwater construction operation happening at the junction (box F), requiring a mid-experiment change of plans: drifters were initially released from A and picked up around F, then redeployed at D and E, then cycled from B–D and C–E. Table 8.1 shows the deploy and release schedule on the two experimental days.

8.4 Assimilation Results

Figure 8.9 shows one of the first assimilation results based on the May 9 drifter data along with existing Eulerian sensors. The water velocity over the experimental domain at 1 PM (about halfway though the experiment) is show. The spacing of the flow vectors is related to

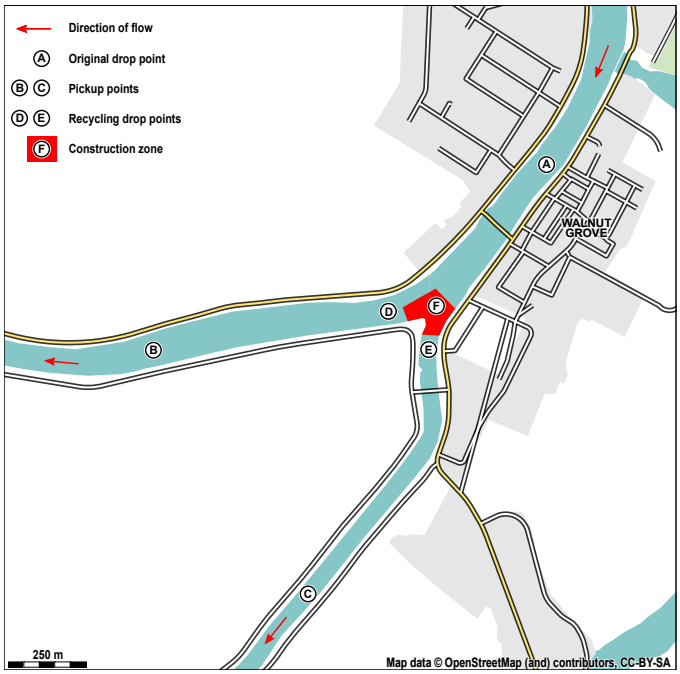

Figure 8.5: Map of experimental region near Walnut Grove, California.

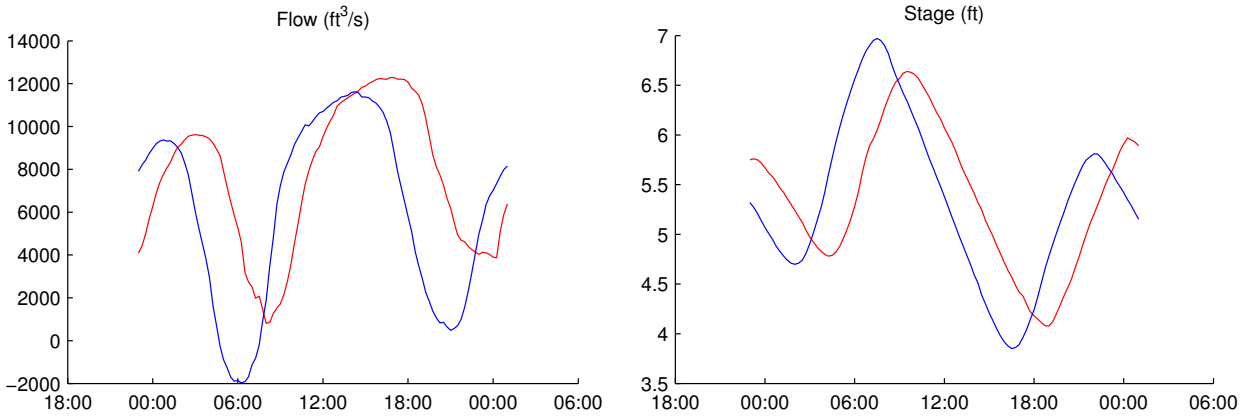

Figure 8.6: Water conditions at the GES field station on the experiment days. Red line: April 12. Blue line: May 9.

	April 12	May 9
0630	Leave Berkeley	Leave Berkeley
0700		
0730		
0800	Arrive Walnut Grove	Arrive Walnut Grove
0830		
0900		
0930		
1000	Start releasing all drifters	
1030	Finish releasing all drifters	
1100	Drifting downstream	Start releasing all drifters
1130	All drifters retrieved	Finish releasing all drifters
1200	Second Gen 3 deployment	All retrieved; re-release Androids at D/E
1230	All drifters retrieved	Cycling continues
1300	Start cycling Androids	Cycling continues
1330	Cycling continues	Cycling continues
1400	Cycling continues	Cycling continues
1430	Retrieve all drifters	Retrieve all drifters
1500	All boats depart	All boats depart
1530	Leave Walnut Grove	Leave Walnut Grove

Table 8.1: Timeline of the experimental procedures.

the REALM computational mesh. The Ensemble Kalman Filter assimilation process on the NERSC Carver cluster is still in development; these early results show the feasibility of the assimilation process. Future work will focus on improving the reliability of the assimilation algorithm and working towards real-time assimilation of data as it is gathered in the field.

8.5 Fleet Operation Analysis

Figure 8.10 shows a detailed view of one active drifter moving through the Sacramento River downstream of the junction with the Georgiana Slough. Active propulsion segments triggered by proximity to the shoreline can be seen in the red track segments. As the drifter floats downstream (from right to left on the map), it repeatedly approaches the south bank of the Sacramento River, triggering propulsion for obstacle avoidance. Note that even though the drifter applied propulsion to return to the center of the river as quickly as possible, the continued influence of the flow field caused it to move at a roughly 45° angle to the straight-line shortest distance.

Designing and provisioning an active sensor fleet depends on a good estimate of the required energy expenditure during field operations. During the design of the Gen 3 drifter,

Figure 8.7: Throwing a Gen 3 drifter from the dock into the Sacramento River. Photo credit: Roy Kaltschmidt.

we estimated that we would need to use the motors on a 10% duty cycle in order to keep the drifters away from obstacles and fulfill navigational goals (see Section 3.3.6 for more details). This target was purely an estimate of the required propulsion effort for a successful mission. We can examine the record of propulsion use on April 12 and May 9 to validate this assumption.

Figure 8.12 shows the fraction of the fleet involved in different types of moves over the course of the mission. Every propulsion action was classified as a "navigation" move if it started in the "dead zone" between the two navigation lanes (in other words, if it started from a region that would have been acceptable if it were not for the modifications to the feedback targets for navigational purposes), while all other moves were classified as "obstacle avoidance". Notice that both May 9 and April 12 have similar patterns of use over time, as the fleet moves from the initial deployment location at the Walnut Grove dock, to the lane split region downstream of the bridge, then to the junction of the Sacramento River and Georgiana Slough. In aggregate, on the April 12 operation, 5.7% of fleet time was spent on obstacle moves, while 4.9% of fleet time was spent on navigation moves. During the May 9 operation, 4.4% of fleet time was spent on obstacle moves, while 6.3% of fleet time was spent

Figure 8.8: Two boats crews retrieving Android drifters near the Sacramento River/Georgiana Slough junction. Photo credit: Roy Kaltschmidt.

on navigation moves. With total propulsion duty cycles of 10.8% and 10.7%, these results validate our design assumptions regarding the energy expenditure of the fleet operations.

In Figure 8.13, the position of the fleet within the river channel is quantified by taking the mean of the MTTR values for the drifters' positions over time. The active drifters and passive drifters are aggregated separately to illustrate the difference in their movement through the channel. As described in Section 5.4.3, two different sets of MTTR feedback maps were used; active drifters were assigned to the Georgiana Slough or Sacramento River branch of the junction through the assignment of policy file. Finding the MTTR values for the passive drifters was done in post-processing; of the two possible policy files, the "best case" file was applied at each time step to each passive drifter.

The solid lines in Figure 8.13 show the average minimum-time-to-obstacle of the two drifter ensembles. This is taken as a proxy for how close the drifters come to the shoreline and other obstacles such as marinas and docks. In the active drifter time series, the triggering of propulsion for the navigational move is seen in the early drop in time-to-shore values. Notice that the aggregate time-to-shore value for the passive fleet is below 300 s for a substantial

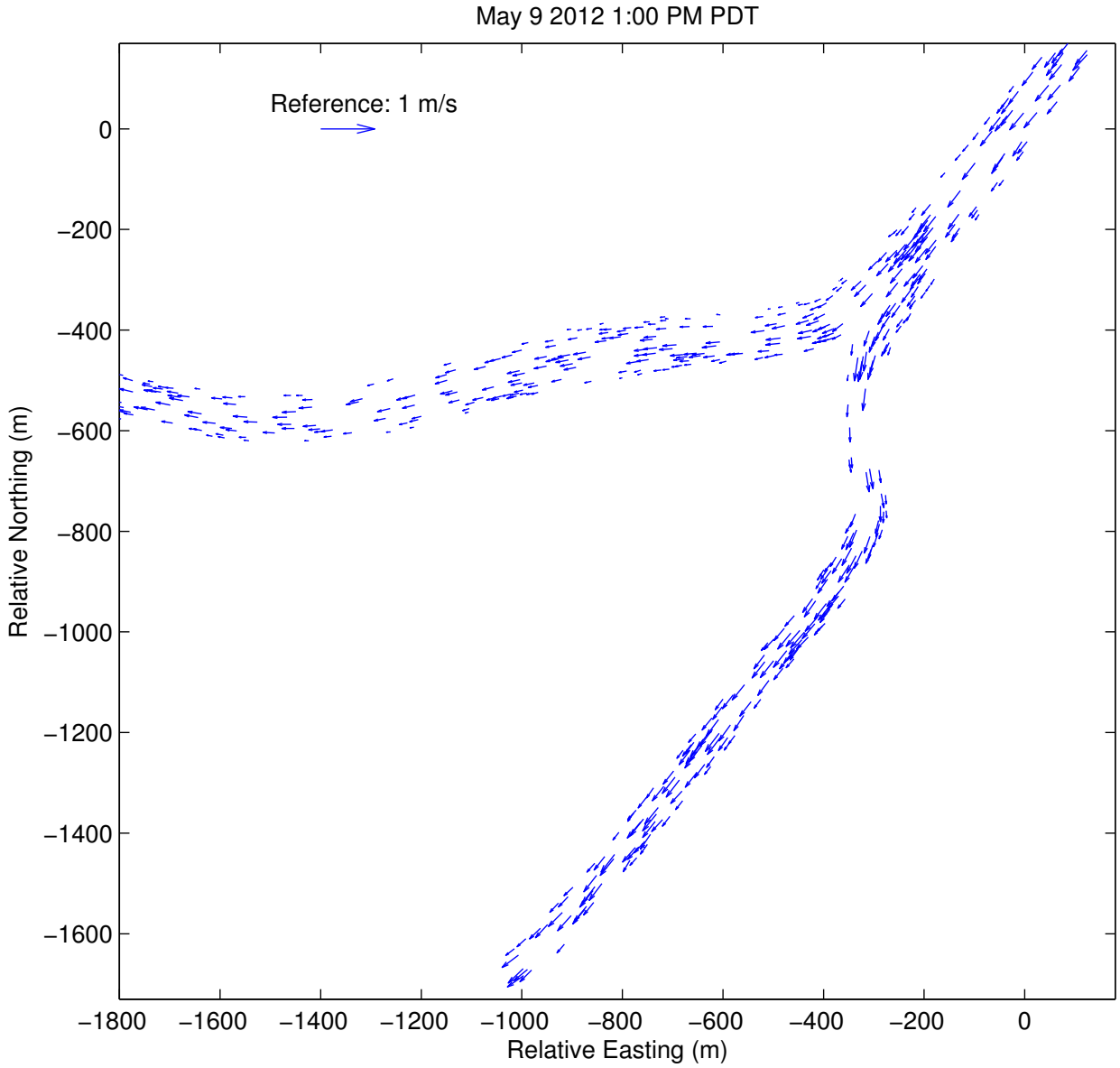

Figure 8.9: Quiver plot of assimilation results for one time step (1:00 PM) during May 9 experiment.

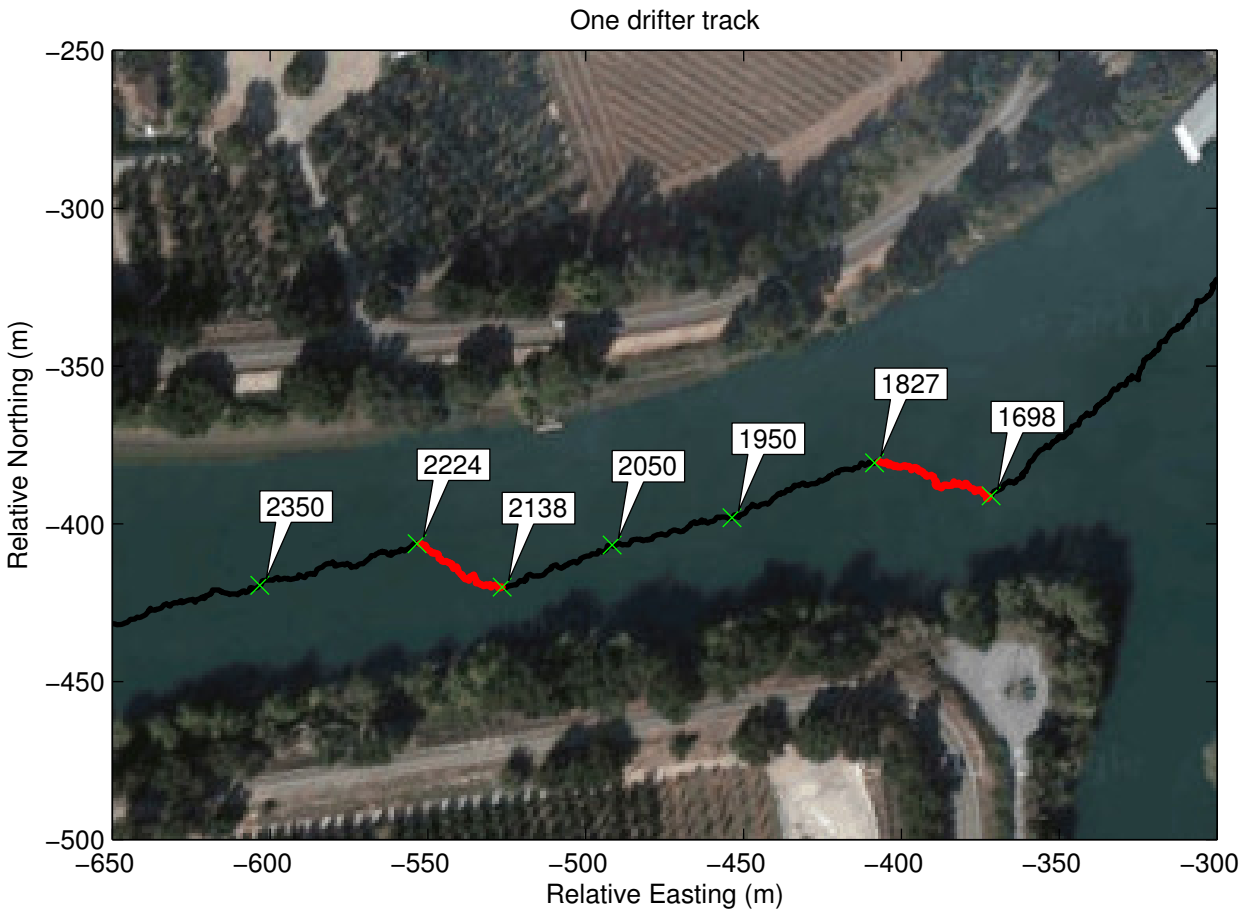

Figure 8.10: Detail view of one active drifter track through experimental region. Red track indicates active propulsion. Numbers indicate mission time in seconds.

portion of the experiment; this indicates that significant portions of the fleet were close enough to the shore that, if they had been active drifters, their propulsion would have been triggered for an obstacle-avoidance move. The time-to-shore value for the active fleet stays mostly above 300 s, for the same reason: any active drifter that comes closer to the shore uses its propulsion to move away.

The dotted lines in Figure 8.13 show the average minimum-time-to-center of the two drifter ensembles. Unlike the minimum-time-to-obstacle time series, there is not a consistent, substantive difference in the aggregate values for the two ensembles. This may be due to the fact that the minimum possible value for the MTTR function is zero; the low variance in the minimum-time-to-center time series may be caused by a "saturation at zero" effect.

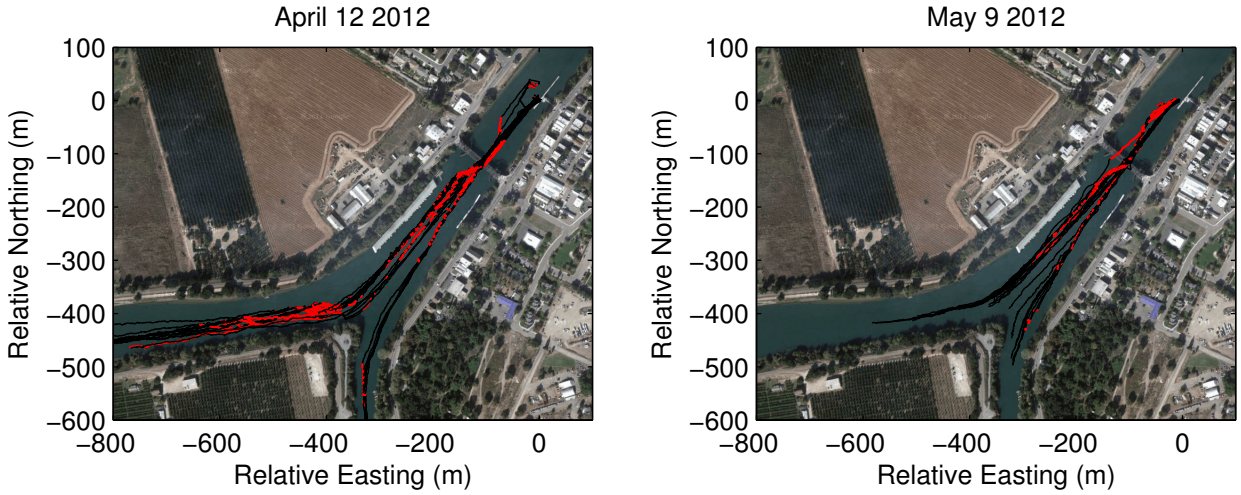

Figure 8.11: Overview of active drifter fleet movement in experimental region on both experimental days. Propelled movement is highlighted in red.

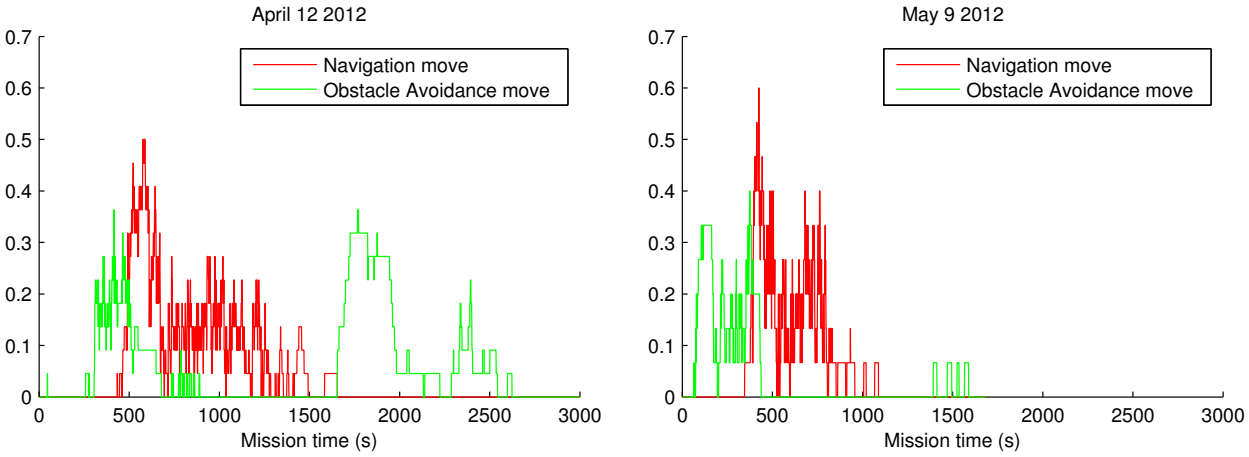

Figure 8.12: Fraction of fleet involved in different types of propulsion during the mission.

8.6 Conclusions

The experiments conducted in Walnut Grove on April 12, 2012 and May 9, 2012 allowed us to demonstrate the following:

1. Small Lagrangian floating sensors are an effective way of gathering water flow information for an Ensemble Kalman Filter driven assimilation process.

2. The parallelizable nature of EnKF computations make them a natural choice for implementation on a computational cluster.

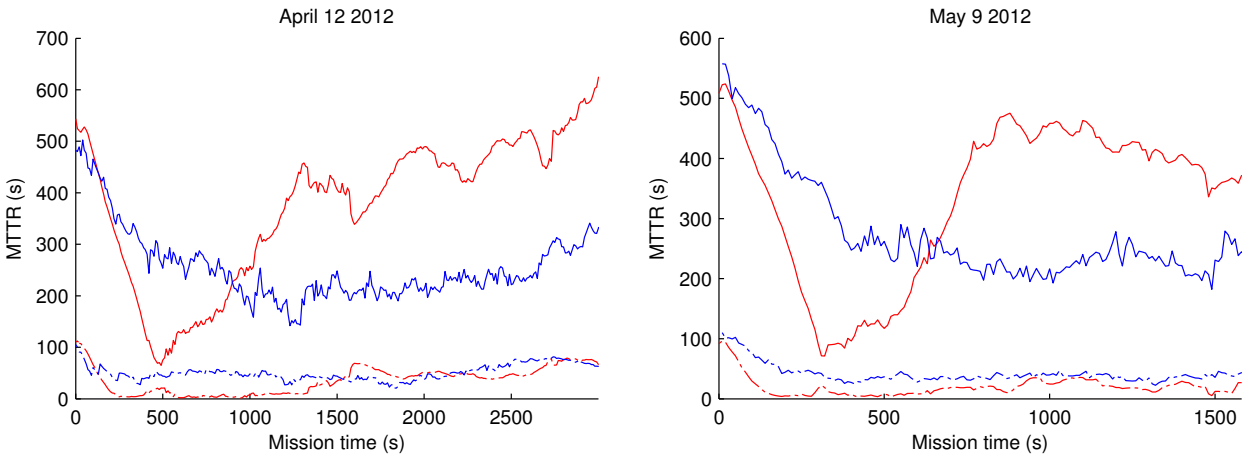

Figure 8.13: Mean MTTR values for active and passive component of fleet. Red lines are active drifters, blue lines are passive drifters. Solid lines are shore MTTR values, dotted lines are center MTTR values.

3. Actuated Lagrangian drifters can accomplish navigational tasks in unstructured environments, and avoid natural obstacles like shorelines; for the propulsion capabilities of the Gen 3 drifter, a duty cycle of 10% is a reasonable first estimate for provisioning movement.

4. The trajectories of an actuated fleet will be significantly different than that of a passive fleet, in particular with respect to proximity to the shoreline and other obstacles. In supervised environments, the passive drifters are a good fit.

Specific disadvantages or drawbacks to the system as implemented on May 9 were:

1. The low reliability of the Gen 3 GSM system drove us to install communication relays over the environment to ensure connectivity. This would not be a feasible solution for larger domains.

2. Despite the small size of the drifters, it would have been unwise to let them drift through the active underwater construction site at the junction of the Georgiana Slough and the Sacramento River. We were less concerned with the possibility of causing harm to the construction operation; our primary concern was the difficulty and danger of retrieving a drifter were it to get caught in the construction equipment. We are very grateful to the Department of Water Resources personnel, who accommodated our experiment with a brief shutdown of operations while we moved through their area.

Future experiments will expand the utility of the Floating Sensor Network system by expanding the spatial and temporal domain of experiments, and working towards real-time processing and assimilation of the incoming data. Solving the communication challenges and

real-time assimilation objective is within our capability. Resolving the issue of interaction with other marine activities is an open problem; for the near future, we will have to mitigate this problem through careful experimental design.

Chapter 9

Conclusions and Future Directions

9.1 State of the Floating Sensor Network System

The feasibility of partially actuated Lagrangian sensing in rivers and estuaries has been successfully demonstrated through the Floating Sensor Network research project. Successive generations of floating sensor development and field experiments have shown that using propulsion in floating devices allows them to satisfactorily avoid the obstacles that make Lagrangian sensing in inland environments difficult.

We refer back to the system decomposition proposed in Chapter 2 to discuss the progress made in the development of these sensor systems, and the challenges ahead for this research.

9.1.1 Lagrangian Units

The design process for the Lagrangian units themselves was comprehensively covered in Chapter 3. The Generation 3 and Android drifters represent the successful design efforts for both active and passive Lagrangian sensors. The Generation 3 device is an actuated device that can successfully handle the obstacle avoidance problem, with a unit cost of approximately $ 2 500. The Android device represents our most successful attempt at lowering the cost of passive Lagrangian sensing, with a unit cost of approximately $ 300. The short-term research path for these devices is two-fold: first, to continue to miniaturize the device, while keeping as much of its present functionality; second, to try to bring the reliability and cost benefits of the Android drifter to the actuated device by replacing the custom integrated electronics with an Android smartphone augmented with microcontrollers for actuation control.

9.1.2 Actuation and Control

Two development paths for the control of active sensor vehicles have been pursued. The single vehicle safety controller, described in Chapter 5, is an extremely successful approach

for the problem of avoiding obstacles in river environments. It is also applicable to navigation challenges like selecting a particular branch of a river network. Extensions to this technique have already been explored, including a method to update the HJBI feedback maps in real-time in case a vehicle encounters a previously unknown obstacle [10]. The HJBI differential game seems like a strong foundation for the single vehicle navigation problem.

One approach to the multi-vehicle control problem was described in Chapter 6. This method included a novel approach to approximating reachability regions in affine flow fields, which is potentially useful to allow online reachability calculations on embedded systems. The applicability of affine flow fields, of course, is a major modeling assumption. One useful direction to consider would be the use of *piecewise affine* flow fields, which could be used as a model for the river flow field.

Overall, the fleet control problem seems to contain significant unexplored territory for future research. The Generation 3 Floating Sensor Network fleet will be a good platform for the development and validation of new approaches to these problems.

9.1.3 Field Team

The field team design problem is largely about how to structure the overall system to reduce the workload on the field team, as well as tools and interface improvements to make the field team's job easier. The development of the obstacle avoidance capability definitely reduces the field team workload, as drifters do not need to be retrieved from riverbanks and other obstacles. While the FSN project did devote significant effort to the interface design problem, it was not part of the work presented in this dissertation. One important conclusion that should be mentioned is that our experience has shown that local communication methods (i.e. the 802.15.4 radios) are far more useful for field work than a centralized GSM channel, due to both reliability issues with the GSM connection as well as latency issues in the chain from the field team to the server to the drifters and back. While the FSN field interface tools could definitely be improved, we note that there has been significant research effort along the same lines from other groups (for example, the NEPTUS system [33]). The FSN project is not near the research frontier in this aspect of the system.

9.1.4 Communication

Wireless sensor networking is an important contemporary research topic. Standardization bodies like the IEEE and the *Internet Engineering Task Force* (IETF) are gradually producing standards for protocols like 802.15.4 networking and 6loWPAN. This is an exciting time for applications-focused researchers in wireless networking. The protocol standards are far from rigid; there are usually many implementation details left unspecified, leading to many opportunities for elaboration or improvements for application-specific needs. The decentralized scheduling scheme proposed in Chapter 7 is one example. It was created to fill a need for networking that can adapt to dynamic connectivity changes, like the conditions

encountered by the Floating Sensor Network fleet. There are many ways in which low-power mesh networking can be improved for the mobile sensor network case. The UC Berkeley OpenWSN project [126] is one example of an ongoing research project seeking opportunities for improvements as the WSN protocols take shape.

9.1.5 Computation

One of the most important aspects of the FSN project is the drive for reliable, efficient data assimilation techniques that can produce estimates from Lagrangian drifter data in real time, while being robust to sensor error, sensor failures, and model uncertainties. The Quadratic Programming technique explored in Chapter 4 was one direction in which the FSN project has made improvements, by reducing a variational approach to a tractable convex optimization problem. Further progress has been made in recursive data assimilation techniques based on the Ensemble Kalman Filter, the Extended Kalman Filter, and other variations. These efforts were not the focus of this dissertation, but continued innovation in this area is key to the future success of Lagrangian sensing in environments driven by Shallow Water Equation type systems.

9.1.6 Visualization

Methods for summarizing, visualizing, and contextualizing the state estimates generated by the Floating Sensor Network project are extremely important for the system to play a useful, relevant role in water managers' decision making. This is an area of the FSN project where there is room for growth. It is very likely that these techniques would need to be developed in close collaboration with the intended end users in order to ensure relevance and applicability.

9.2 Future Directions

We conclude with a brief look at some possible directions for future work on actuated Lagrangian sensing.

9.2.1 Unrecovered Drifters

One way to almost eliminate the field team workload during experiments would be to design and build Lagrangian drifters so that they would not need to be recovered at the end of the experiment. Considering the functional requirements list developed in Chapter 3, this would require radical changes to the economic properties (bringing the cost of the drifters low enough to be single-use) and the liquidation properties (so that the environmental impact of the discarded drifters would be sufficiently small). Some sort of biodegradeable hull could

possibly mitigate the environmental impact; at present, we are not aware of any kind of electronics technology that is so low-impact that it can be responsibly discarded, especially if the experimental domain is particularly sensitive.

9.2.2 Depth-profiling Drifters

Estuarial environments often have interesting stratification and depth-dependent mixing processes as the warm freshwater meets the cold seawater. Drifters with the ability to modify their buoyancy would be able to regulate their depth, to investigate phenomena at different levels of the water column. As an additional benefit, the problem of wind influence and the safety hazards posed to other occupants of the river would almost completely disappear once the device was deeper than about one meter. Depth profiling is a standard technique in oceanography; however, like the floating Lagrangian sensor, the presence of unstructured obstacles in the river environment makes this a difficult technology to adapt. One complicating factor is that RF-based communication is almost certainly unavailable at any significant depth underwater. The key functional requirement where improvement would be needed is reliability. In the floating sensor case, almost all system failures are non-fatal. As long as the device is still buoyant, it can probably be retrieved by the field team for diagnosis and repair. In contrast, if sensor devices sink below the surface of the water for protracted lengths of time, almost any system failure will lead to the loss of the device. Several kinds of redundancy and safety measures would be necessary to bring the system reliability up to the level where subsurface experiments could be attempted with reasonable chances of success.

9.2.3 Data-driven Control

The vehicle control techniques documented in Chapters 5 and 6 are both position-driven techniques; the control depends on the current position of the device, or in the Zermelo-Voronoi case, the position of the device and the position of its neighbors. There are many exciting research opportunities for wireless sensor networks that can alter their configuration based on the measurements they are taking. Tracing out concentration gradients of a particular constituent, or moving so as to reduce the uncertainty in the overall state estimate, or seeking out uncovered areas to provide guaranteed coverage of an area — these are all ideas that have been developed in the literature, that could be brought to practice with a system like the Floating Sensor Network. Closing the loop between the state estimate made from the drifter measurements back to the actuation that drives the drifter positions would be an exciting application of classic control theory ideas to a multi-vehicular framework.

Bibliography

[1] J. Anderson and M. Mierzwa. DSM2 tutorial, an introduction to the Delta Simulation Model II (DSM2). Technical report, State of California, Department of Water Resources, 2002.

[2] K. Ang, G. Chong, and Y. Li. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005.

[3] E. Ateljevich, P. Colella, D. T. Graves, T. J. Ligocki, J. Percelay, P. O. Schwartz, and Q. Shu. CFD modeling in the San Francisco Bay and Delta. In *Proceedings of the Fourth SIAM Conference on Mathematics for Industry*, pages 99–107. San Francisco, California, 2009.

[4] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre. *Viability Theory: New Directions*. Springer-Verlag, Heidelberg, 2nd edition, 2011. doi:10.1007/978-3-642-16684-6.

[5] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991. doi:10.1145/116873.116880.

[6] E. Bakolas and P. Tsiotras. The Zermelo-Voronoi diagram: a dynamic partition problem. *Automatica*, 46(12):2059–2067, 2010. doi:10.1016/j.automatica.2010.09.003.

[7] M. Bardi and I. Cappuzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhäuser, Boston, 1997.

[8] A. M. Bayen, J. Butler, and A. D. Patire. Mobile Millennium: GPS mobile phones as traffic probes, California Networked Traveler — Safe Trip 21 Phase II. Technical Report UCB-ITS-CWP-2011-6, California Center for Innovative Transportation, 2011.

[9] A. M. Bayen, I. M. Mitchell, S. Santhanam, and C. Tomlin. A differential game formulation of alert levels in ETMS data for high altitude traffic. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*. August 2003.

[10] J. Beard. *Autonomous Waterway Navigation Using the Hamilton-Jacobi Equations for Underactuated Vehicles with On-line Obstacle Avoidance*. Master's thesis, University of California, Berkeley, 2012.

[11] J. Beard, K. Weekly, C. Oroza, A. Tinka, and A. M. Bayen. Mobile phone based drifting flow sensors. In *3rd IEEE International Conference on Networked Embedded Systems for Every Application*. 2012. To appear.

[12] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[13] P. Bhatta, E. Fiorelli, F. Lekien, N. Leonard, D. Paley, F. Zhang, R. Bachmayer, R. Davis, D. Fratantoni, and R. Sepulchre. Coordination of an underwater glider fleet for adaptive ocean sampling. In *Proc. International Workshop on Underwater Robotics*. 2005.

[14] D. S. Bitterman and D. V. Hansen. The design of a low cost tropical drifter buoy. In *mds '86: Marine Data Systems International Symposium*, pages 575–581. 1986.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[16] A. E. Bryson and Y. Ho. *Applied Optimal Control*. Blaisdell Publishing Company, Waltham, Maryland, 1969.

[17] M. Buettner, E. Yee, G. V. Anderson, and R. Han. X-mac: A short preamble mac protocol for duty-cycled wireless sensor networks. In *4th international conference on Embedded networked sensor systems (SenSys)*. ACM, Boulder, Colorado, 31 October - 3 November 2006.

[18] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. In *International Conference on Information Processing in Sensor Networks*. ACM, New York, 2007.

[19] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre. Set-valued numerical analysis for optimal control and differential games. *Stochastic and Differential Games: Theory and Numerical Methods*, 4:177–247, 1999.

[20] D. Carle. *Introduction to water in California*. University of California Press, Berkeley, California, 2004.

[21] J. N. Carruthers. Further investigation upon the water movements in the English Channel. Drift-bottle experiments in the summers of 1927, 1928 and 1929, with critical notes on drift-bottle experiments in general. *Journal of the Marine Biological Association of the United Kingdom*, 17(1):241–275, 1930.

[22] J. N. Carruthers. Some oceanography from the past. *The Journal of Navigation*, 16(2):180–188, 1963.

[23] C. Cassandras and S. Lafortune. *Introduction to discrete event systems*, volume 11. Kluwer Academic Publishers, Norwell, Massachusetts, 1999.

[24] A. Chadwick, J. Morfett, and M. Borthwick. *Hydraulics in Civil and Environmental Engineering*. Spon Press, London, UK, 2004.

[25] V. T. Chow. *Open-Channel Hydraulics*. McGraw-Hill, New York, NY, 1959.

[26] D. D. Clark. Overview of the Argos system. In *OCEANS '89. Proceedings*, volume 3, pages 934–939. September 1989. doi:10.1109/OCEANS.1989.586711.

[27] Clean Water Act of 1972: 33 USC § 1344.

[28] J.-M. Coron, B. d'Andréa-Novel, and G. Bastin. A strict Lyapunov function for boundary control of hyperbolic systems of conservation laws. *IEEE Transactions on Automatic Control*, 52(1):2–11, January 2007.

[29] J. Cortés, S. Martínez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005. doi:10.1051/cocv:2005024.

[30] R. Courant. *Differential and Integral Calculus*, volume 2. Interscience Publishers, New York, NY, 1937.

[31] R. E. Davis. Drifter observations of coastal surface currents during CODE: The method and descriptive view. *Journal of Geophysical Research*, 90(C3):4741–4755, 1985. doi: 10.1029/JC090iC03p04741.

[32] C. Detweiller, I. Vasilescu, and D. Rus. An underwater sensor network with dual communications, sensing, and mobility. In *OCEANS 2007-Europe*, pages 1–6. IEEE, 2007. doi:10.1109/OCEANSE.2007.4302445.

[33] P. S. Dias, J. Pinto, R. Gonçalves, and J. B. Sousa. Enabling a dialog — a C2I infrastructure for unmanned vehicles and sensors. In *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, pages 1–6. June 2010. doi: 10.1109/AIS.2010.5547024.

[34] L. Doherty, W. Lindsay, and J. Simon. Channel-specific wireless sensor network path data. In *16th International Conference on Computer Communications and Networks (ICCCN)*, pages 89–94. IEEE, Turtle Bay Resort, Honolulu, Hawaii, August 13–16 2007.

[35] J. Drolet and W. G. Gray. On the well posedness of some wave formulations of the shallow water equations. *Advances in Water Resources*, 11(2):84–91, 1988. ISSN 0309-1708. doi:10.1016/0309-1708(88)90041-3.

[36] Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[37] K. R. Dyer. *Estuaries: A Physical Introduction*. John Wiley & Sons, London, UK, 1973.

[38] EDF. Telemac 2D. version 5.2. Technical report, EDF, 2003.

[39] S. C. Ergen and P. Varaiya. PEDAMACS: Power efficient and delay aware medium access protocol for sensor networks. *IEEE Transactions on Mobile Computing*, 5(7), 2006.

[40] European Telecommuniations Standards Institute. *3GPP TS 23.060 version 10.4.0: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UTMS); General Packet Radio Service (GPRS); Service description; Stage 2"*, 2011.

[41] L. Evans and P. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. Technical report, DTIC Document, 1983.

[42] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99(C5):10143–10162, 1994. doi:10.1029/94JC00572.

[43] G. Evensen. The Ensemble Kalman Filter: theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003. doi:10.1007/s10236-003-0036-9.

[44] S. Finger and J. R. Dixon. A review of research in mechanical engineering design. Part I: Descriptive, prescriptive, and computer-based models of design processes. *Research in engineering design*, 1(1):51–67, 1989.

[45] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, Boston, 3rd edition, 2004.

[46] H. Frankowska. Lower semicontinuous solutions of Hamilton-Jacobi-Bellman equations. In *SIAM Journal of Control and Optimization*, volume 31, pages 257–272. January 1993.

[47] L. Fu, E. Christensen, C. Yamarone Jr, M. Lefebvre, Y. Menard, M. Dorrer, and P. Escudier. TOPEX/POSEIDON mission overview. Technical report, Jet Propulsion Laboratory, 1994.

[48] K. George. A depth-averaged tidal numerical model using non-orthogonal curvilinear co-ordinates. *Ocean Dynamics*, 57(4–5):363–374, October 2007.

[49] Google. Protocol buffers: Google's data interchange format. `https://developers.google.com/protocol-buffers/`.

[50] W. J. Gould. From Swallow floats to Argo–the development of neutrally buoyant floats. *Deep Sea Research Part II: Topical Studies in Oceanography*, 52(3–4):529–543, 2005.

[51] D. Gustafsson. *WirelessHART - Implementation and Evaluation on Wireless Sensors*. Master's thesis, Kungliga Tekniska högskolan, April 2009.

[52] T. Harmon, R. Ambrose, R. Gilbert, J. Fisher, M. Stealey, and W. Kaiser. High-resolution river hydraulic and water quality characterization using rapidly deployable networked infomechanical systems (NIMS RD). *Environmental Engineering Science*, 24(2):151–159, 2007. doi:10.1089/ees.2006.0033.

[53] HART field communication protocol specifications, DDL specifications, 2008.

[54] J. D. Hoffman. *Numerical Methods for Engineers and Scientists*. Marcel Dekker, New York, NY, 2nd edition, 2001.

[55] M. Honnorat, J. Monnier, and F.-X. Le Dimet. Lagrangian data assimilation for river hydraulics simulations. *Computing and Visualization in Science*, 12(5):235–246, 2009. doi:10.1007/s00791-008-0089-x.

[56] V. Hubka and W. E. Eder. *Theory of Technical Systems*. Springer-Verlag, Berlin, 1988.

[57] K. Ide, P. Courtier, M. Ghil, and A. Lorenc. Unified notation for data assimilation: Operational, sequential and variational. *J. Met. Soc. of Japan*, 75(1B):71–79, 1997.

[58] IEEE. *Programs for digital signal processing*. IEEE Press, New York, 1979.

[59] IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 8 September 2006.

[60] IEEE Std 802.15.4e-2012 Standard for Local and metropolitan area networks: Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs): Amendment 1: MAC sublayer, April 2012.

[61] ISA-100.11a-2009: Wireless Systems for Industrial Automation: Process Control and Related Applications, September 11 2009.

[62] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. John Wiley and Sons, New York, NY, 1965.

[63] V. J. Katz. Change of variables in multiple integrals: Euler to Cartan. *Mathematics Magazine*, 55(1):3–11, January 1982. doi:10.2307/2687130.

[64] A. Kwok and S. Martínez. Deployment of drifters in a piecewise-constant flow environment. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 6584–6589. December 2010. doi:10.1109/CDC.2010.5717309.

[65] J.-C. Latombe. *Robot motion planning*. Springer-Verlag, Berlin, 1990.

[66] S. M. LaValle. *Planning algorithms*. Cambridge University Press, Cambridge, UK, 2006.

[67] F.-X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meterological observations: theoretical aspects. *Tellus*, 38A:97–110, 1986. doi: 10.1111/j.1600-0870.1986.tb00459.x.

[68] H. Lindeboom. The coastal zone: an ecosystem under pressure. In J. Field, G. Hempel, and C. Summerhayes, editors, *Oceans 2020: Science, Trends, and the Challenge of Sustainability*, pages 49–84. Island Press, Washington, 2002.

[69] R. Löhner. *Applied CFD techniques*. J. Wiley & Sons, 2nd edition, 2008.

[70] J. Lund, E. Hanak, W. Fleenor, R. Howitt, J. Mount, and P. Moyle. *Envisioning Futures for the Sacramento-San Joaquin Delta*. Public Policy Institute of California, San Francisco, CA, 2007.

[71] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica-Oxford*, 35:349–370, 1999.

[72] M. Mierzwa and B. Suits. Jones tract 2004 levee break DSM2 simulation. Technical report, California Department of Water Resources, October 2005.

[73] I. M. Mitchell. A toolbox of level set methods. *UBC Department of Computer Science Technical Report*, 11, 2007.

[74] I. M. Mitchell, A. M. Bayen, and C. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, 2005.

[75] Motorola. *Motorola G24 Developers' Guide: Module Hardware Description*, 2007.

[76] NERSC's Carver iDataPlex Cluster, 2012.

[77] P. P. Niiler, R. E. Davis, and H. J. White. Water-following characteristics of a mixed layer drifter. *Deep-Sea Research*, Vol. 34, No. 11:1867–1881, 1987.

[78] T. Nishida and K. Sugihara. Approximation of the boat-sail voronoi diagram and its application. In A. Laganá, M. Gavrilova, V. Kumar, Y. Mun, C. Tan, and O. Gervasi, editors, *Computational Science and Its Applications — ICCSA 2004*, volume 3045 of *Lecture Notes in Computer Science*, pages 227–236. Springer Berlin / Heidelberg, 2004. doi:10.1007/978-3-540-24767-8_24.

[79] C. Oestges, B. Montenegro-Vollacieros, and D. Vanhoenacker-Janvier. Radio channel characterization for moderate antenna heights in forest areas. *IEEE Transactions on Vehicular Technology*, 58(8):4031–4035, October 2009.

[80] T. Oki and S. Kanae. Global hydrological cycles and world water resources. *Science*, 313(5790):1068–1072, 2006. doi:10.1126/science.1128845.

[81] S. A. Olsen and J. M. Norris. U.S. Geological Survey streamgaging. Fact Sheet 2005–3131, United States Geological Survey, March 2007.

[82] C. Oroza, A. Tinka, P. K. Wright, and A. M. Bayen. Design of a network of robotic Lagrangian sensors for shallow water environments with case studies for multiple applications. *Journal of Mechanical Engineering Science*, 2012. In review.

[83] S. Osher. A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations. *SIAM Journal on Mathematical Analysis*, 24:1145, 1993.

[84] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Verlag, London, UK, 2003.

[85] G. Pahl and W. Beitz. *Engineering Design: A Systematic Approach*. Springer-Verlag, London, 2nd edition, 1996.

[86] Panasonic. *LC-P0612P datasheet*, June 2000.

[87] Panasonic. *HHR110AAO datasheet*, January 2004.

[88] Panasonic. *NCR18650 datasheet*, February 2010.

[89] Panel on Dietary Reference Intakes for Electrolytes and Water and Standing Committee on the Scientific Evaluation of Dietary Reference Intakes. *Dietary Reference Intakes for Water, Potassium, Sodium, Chloride, and Sulfate*. The National Academies Press, 2005. ISBN 9780309091695.

[90] Parker Hannifin Corporation, Cleveland, Ohio. *Parker O-Ring Handbook*, 2007.

[91] C. L. Parkinson. Aqua: an Earth-Observing Satellite mission to examine water and other climate variables. *Geoscience and Remote Sensing, IEEE Transactions on*, 41(2):173–183, February 2003. doi:10.1109/TGRS.2002.808319.

[92] K. Pister and L. Doherty. TSMP: Time synchronized mesh protocol. In *Parallel and Distributed Computing and Systems (PDCS)*. Orlando, Florida, November 16–18 2008.

[93] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 95–107. ACM Press, Baltimore, Maryland, November 3–5 2004.

[94] S. L. Postel. Entering an era of water scarcity: The challenges ahead. *Ecological Applications*, 10(4):941–948, 2000.

[95] D. W. Pritchard. What is an estuary: Physical viewpoint. In *Estuaries*, pages 3–5. American Association for the Advancement of Science, 1967.

[96] V. Rajendran, K. Obraczka, and J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wireless Networks*, 12(1), 1995.

[97] S. Ramp, R. Davis, N. Leonard, I. Shulman, Y. Chao, A. Robinson, J. Marsden, P. Lermusiaux, D. Fratantoni, J. Paduan, F. Chavez, F. Bahr, S. Liang, W. Leslie, and Z. Li. Preparing to predict: The second autonomous ocean sampling network (AOSN-II) experiment in the Monterey Bay. *Deep Sea Research Part II: Topical Studies in Oceanography*, 56(3–5):68–86, 2009. doi:10.1016/j.dsr2.2008.08.013.

[98] Y. Ru and S. Martínez. Energy-based voronoi partition in constant-flow environments. *IEEE Transactions on Automation Science and Engineering*, 2012. Submitted, November 2011.

[99] P. Saint-Pierre. Approximation of the viability kernel. *Applied Mathematics and Optimization*, 29(2):187–209, 1994.

[100] Sanyo. *UPF673791 datasheet*, March 2011.

[101] J. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Cambridge University Press, Cambridge, UK, 1999.

[102] I. A. Shiklomanov and J. C. Rodda, editors. *World Water Resources at the Beginning of the* 21^{st} *Century*. International Hydrology Series. Cambridge University Press, Cambridge, UK, 2003.

[103] D. R. Smith. *Variational Methods in Optimization*. Prentice-Hall, New Jersey, 1974.

[104] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications Magazine*, 7(5), 2000.

[105] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. WirelessHART: Applying wireless technology in real-time industrial process control. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 377–386. 2008. doi:10.1109/RTAS.2008.15.

[106] Interim population projections for California and its counties 2010–2050. Sacramento, California, May 2012. State of California, Department of Finance.

[107] L. A. Steen and J. J. Arthur Seebach. *Counterexamples in Topology*. Springer-Verlag, New York, 2nd edition, 1978.

[108] R. F. Stengel. *Optimal control and estimation*. Dover, 1994.

[109] T. W. Sturm. *Open Channel Hydraulics*. McGraw-Hill, 2nd edition, 2010.

[110] K. Sugihara. Voronoi diagrams in a river. Technical Report 90–1052, Purdue University, December 1990.

[111] N. P. Suh. *The Principles of Design*. Oxford University Press, New York, New York, 1990.

[112] P. B. Sujit, J. Sousa, and F. L. Pereira. UAV and AUVs coordination for ocean exploration. In *OCEANS 2009-EUROPE*, pages 1–7. IEEE, 2009. doi:10.1109/OCEANSE.2009.5278262.

[113] S. Sundar and Z. Shiller. Optimal obstacle avoidance based on the Hamilton-Jacobi-Bellman equation. *IEEE Transactions on Robotics and Automation*, 13(2):305–310, 1997.

[114] J. C. Swallow. A neutral-buoyancy float for measuring deep currents. *Deep Sea Research (1953)*, 3(1):74–81, 1955. doi:10.1016/0146-6313(55)90037-X.

[115] T. Swift, R. DuVall, and S. Balachandra. Jones tract flood water quality investigations. Technical report, California Department of Water Resources, Sacramento, California, July 2009.

[116] B. Tapley, S. Bettadpur, M. Watkins, and C. Reigber. The gravity recovery and climate experiment: Mission overview and early results. *Geophys. Res. Lett*, 31(9), 2004. doi:10.1029/2004GL019920.

[117] Texas Instruments Inc. 2.4 GHz IEEE 802.15.4 / ZigBee-Ready RF Transceiver (Rev. B), March 20 2007. Data Sheet SWRS041B [available online].

[118] A. Tinka, S. Diemer, L. Madureira, E. R. Marques, J. B. de Sousa, R. Martins, J. Pinto, J. E. da Silva, A. Sousa, P. Saint-Pierre, and A. M. Bayen. Viability-based computation of spatially constrained minimum time trajectories for an autonomous underwater vehicle: implementation and experiments. In *American Control Conference*. St. Louis, Missouri, June 2009.

[119] A. Tinka, M. Rafiee, and A. M. Bayen. Floating sensor networks for river studies. *IEEE Systems Journal*, 2012. doi:10.1109/JSYST.2012.2204914. To appear; accepted Dec. 2011.

[120] A. Tinka, I. Strub, Q. Wu, and A. M. Bayen. Quadratic programming based data assimilation with passive drifting sensors for shallow water flows. *International Journal of Control*, 83(6):1686–1700, August 2010. doi:10.1080/00207179.2010.489621.

[121] Y. Uchiyama. Wetting and drying scheme for POM and its applications to San Francisco Bay. In L. Cheng and K. Yeow, editors, *Hydrodynamics VI: theory and applications: proceedings of the 6th International Conference on Hydrodynamics*, pages 293–299. Perth, Western Australia, 2004.

[122] F. Voegeli, G. Lacroix, and J. Anderson. Development of miniature pingers for tracking atlantic salmon smolts at sea. *Hydrobiologia*, 371–372:35–46, 1998. doi:10.1023/A:1017014903313.

[123] C. Vreugdenhil. *Numerical Methods for Shallow Water Flow*. Kluwer Academic Publishers, Norwell, Massachusetts, 1994.

[124] T. Watteyne, S. Lanzisera, A. Mehta, and K. Pister. Mitigating multipath fading through channel hopping in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*. IEEE, Cape Town, South Africa, May 23–27 2010.

[125] T. Watteyne, A. Mehta, and K. Pister. Reliability through frequency diversity: Why channel hopping makes sense. In *6th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc Sensors and Ubiquitous Networks (PE-WASUN)*. Tenerife, Canary Islands, Spain, October 26–30 2009.

[126] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, and K. Pister. OpenWSN: a standards-based low-power wireless development environment. *Transactions on Emerging Telecommunications Technologies*, 23(5):480–493, 2012. ISSN 2161-3915. doi:10.1002/ett.2558.

[127] D. Webb, P. Simonetti, and C. Jones. SLOCUM: an underwater glider propelled by environmental energy. *Oceanic Engineering, IEEE Journal of*, 26(4):447–452, oct 2001. doi:10.1109/48.972077.

[128] K. Weekly, L. Anderson, A. Tinka, and A. M. Bayen. Autonomous river navigation using the Hamilton-Jacobi framework for underactuated vehicles. In *International Conference on Robotics and Automation*, pages 828–833. IEEE, 2011.

[129] R. Whitehouse. *Dynamics of Estuarine Muds*. Thomas Telford, London, UK, 2000.

[130] J. Wilson, T.C. Advances in autonomous GPS lagrangian buoys. In *Current Measurement, 1995., Proceedings of the IEEE Fifth Working Conference on*, pages 157–162. feb 1995. doi:10.1109/CCM.1995.516167.

[131] J. Wilson, T.C., J. Barth, S. Pierce, P. Kosro, and B. Waldorf. A Lagrangian drifter with inexpensive wide area differential GPS positioning. In *OCEANS '96. MTS/IEEE. Prospects for the 21st Century. Conference Proceedings*, volume 2, pages 851–856. sep 1996. doi:10.1109/OCEANS.1996.568340.

[132] E. Wolanski. *Estuarine Ecohydrology*. Elsevier, Amsterdam, The Netherlands, 2007.

[133] J. Wolff. *Delta Primer*. William Stout Publishers, San Francisco, CA, 2003.

[134] W. Ye, F. Silva, and J. Heidemann. Ultra-low duty cycle MAC with scheduled channel polling. In *4th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pages 321–334. ACM, Boulder, Colorado, November 1–3 2006.

[135] M. M. Zdravkovich, V. P. Brand, G. Mathew, and A. Weston. Flow past short circular cylinders with two free ends. *Journal of Fluid Mechanics*, 203(1):557–575, 1989. doi: 10.1017/S002211208900159X.

[136] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13. ACM, 2003.

[137] O. C. Zienkiewicz and D. V. Phillips. An automatic mesh generation scheme for plane and curved surfaces by 'isoparametric' co-ordinates. *International Journal for Numerical Methods in Engineering*, 3(4):519 – 528, 1971. doi:10.1002/nme.1620030407.

[138] F. Zwicky. *Discovery, Invention, Research*. The Macmillan Company, Toronto, Ontario, 1969. Translated from German.