

AUTONOMOUS SENSING OF A GAUSSIAN SPATIAL PROCESS WITH  
MULTIPLE HETEROGENEOUS AGENTS

by

Michael Brancato

A thesis submitted to the faculty of  
The University of North Carolina at Charlotte  
in partial fulfillment of the requirements  
for the degree of Master of Science in  
Mechanical Engineering

Charlotte

2022

Approved by:

---

Dr. Artur Wolek

---

Dr. Scott Kelly

---

Dr. Stuart Smith



## ABSTRACT

MICHAEL BRANCATO. Autonomous sensing of a Gaussian spatial process with multiple heterogeneous agents. (Under the direction of DR. ARTUR WOLEK)

The performance of a mobile sensor network is measured by its ability to survey regions of interest efficiently and accurately with limited resources. Approaches for sampling trajectory optimization allow for adaptive algorithms that offer significant improvements in mapping error when compared with non-adaptive approaches.

This thesis proposes an algorithm for generating adaptive sampling trajectories for a collaborative multi-agent team with heterogeneous mobility and sensing capabilities. Each agent, modeled as a differential thrust vehicle, contributes to the team's estimate of an unknown attribute in a region of interest by traversing the space while in communication with a centralized planner. The spatial distribution of the attribute is modeled as a stationary, isotropic Gaussian random field. Noisy local measurements of the attribute are synthesized into a global estimate of the underlying field using a Gaussian process regression technique known as kriging. A modified kriging method is proposed to accommodate the potential heterogeneity of measurement errors while improving computation time. A Voronoi-based algorithm is proposed which periodically partitions the sampling space to identify high-value sampling locations. Each agent's path is constructed using waypoints which compose an analogous mechanical system where virtual springs and masses connect sequential waypoints and mass centroids of Voronoi cells. By modeling each waypoint as a point mass within this spring-mass-damper system, an equilibrium position can be identified using an iterative process by which the system constraints are satisfied

through the simulation of a virtual agent through the proposed waypoint set.

Numerical simulations compare the proposed strategy with non-adaptive traversal of a Gaussian random field to validate the effectiveness of the proposed solution. The simulation results show a marked improvement when compared with the non-adaptive sampling methods in scalar fields with sufficient variability in space. The approach is also demonstrated through field experiments conducted on Lake Norman, NC using two custom designed autonomous surface vessel (ASV) mobile sensing platforms to observe bathymetric data. The mechanical, electrical and software design of the ASVs developed for this work is discussed.

## ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Artur Wolek, for his time, encouragement, and advice throughout the process of researching and writing my thesis. I am also deeply indebted to my committee members, Dr. Scott Kelly and Dr. Stuart Smith, for their insights and technical support. I would like to thank my parents who made my time in graduate school possible. Finally, a special thank you to my girlfriend, Brenna Calderara, for her support and optimism, even when I was insufferable.

## TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER 1: INTRODUCTION	1
CHAPTER 2: PRELIMINARIES	5
2.1. Gaussian Processes	5
2.1.1. Covariance Functions	6
2.1.2. Conditioning of Gaussian Processes	7
2.1.3. Simulation of Gaussian Processes	7
2.2. Kriging	10
2.2.1. Heterogeneous Measurement-Error Filtered Kriging	14
2.3. Centroidal Voronoi Tessellation	16
CHAPTER 3: EFFICIENT GAUSSIAN PROCESS REGRESSION WITH HETEROGENEOUS MEASUREMENT NOISE	18
3.1. Filtered Kriging with Heterogeneous Sensors	18
3.2. Adaptive Spatial Truncation	19
3.3. Common Data Neighborhood Discretization	20
3.4. Heterogeneous, Common-Data-Neighborhood Kriging	21

	vii
CHAPTER 4: MULTI-AGENT ADAPTIVE SAMPLING	25
4.1. Sampling Strategy	25
4.1.1. Heterogeneous Agent Dynamics	26
4.1.2. Heterogeneous Agent Sensing	28
4.1.3. Problem Statement	28
4.1.4. Algorithm	31
4.2. Simulation Results	39
4.2.1. Monte Carlo Simulation Setup	39
4.2.2. Monte Carlo Simulation Results	41
4.2.3. Illustrative Example	42
CHAPTER 5: DESIGN OF AUTONOMOUS SURFACE VESSELS	45
5.1. Background	45
5.2. ASV Design Overview	47
5.2.1. Sensor Suite	48
5.2.2. Electrical	53
5.2.3. Hardware, Final Build and Total System Cost	57
5.2.4. Software	60
5.3. Mobile Sensor Network Configuration	66
5.3.1. Centralized Control	66

	viii
5.3.2. Communication Topology	68
5.3.3. Ground Station User Interface	70
5.4. ASV Practical Control Design	72
5.5. Experimentation	74
5.5.1. Wireless Communication Range Testing	74
5.5.2. ASV Control Testing	76
5.6. Design Review	78
CHAPTER 6: EXPERIMENTAL RESULTS	80
6.1. Experimental Environment	80
6.2. Ground Truth Survey	81
6.3. Experimental Methodology and Evaluation Criteria	83
6.4. Mission Description	86
6.4.1. Experimental Trials	87
6.4.2. Discussion of Experimental Results	89
CHAPTER 7: CONCLUSION AND FUTURE WORK	93
REFERENCES	95
APPENDIX A: Experimental Results Supplemental Information	100
APPENDIX B: System Design Diagrams	101
APPENDIX C: Bill of Materials	104



## LIST OF TABLES

TABLE 4.1: Parameters used in Monte Carlo simulation.	41
TABLE 4.2: Comparison of simulation results given all permutations of dynamics and sensing quality.	42
TABLE 5.1: A comparison between commercially available ASVs and the planned UNCC ASV.	47
TABLE 5.2: Communication packet information for 3DM-GX5 systems.	50
TABLE 5.3: Communication packet information for BU-353S4 using NMEA-0183 string format.	50
TABLE 5.4: Communication packet information for GNSS output of 3DM-GX5-45.	52
TABLE 5.5: High-level Bill of Materials for ASV platform.	61
TABLE 5.6: List of packet descriptors for communication with control computer.	65
TABLE 5.7: Actual performance characteristics comparison with desired.	79
TABLE 6.1: Common mission parameters among missions.	84
TABLE 6.2: Mission parameters for experimental trials.	86
TABLE C.1: Detailed ASV Bill of Materials.	105
TABLE C.2: Detailed Ground Station Bill of Materials.	106

## LIST OF FIGURES

FIGURE 2.1: Progressive Gaussian process conditioning without observation noise.	8
FIGURE 2.2: Progressive Gaussian process conditioning with addition of noisy observations.	9
FIGURE 2.3: Comparison of Gaussian process simulation methods.	11
FIGURE 2.4: Two-dimensional field estimation using ordinary kriging.	13
FIGURE 3.1: Proposed adaptive truncation strategy.	21
FIGURE 3.2: A comparison between mapping error and computation for the proposed kriging methods.	24
FIGURE 4.1: State and input diagram for waypoint tracking.	27
FIGURE 4.2: Constrained CVT diagram for demonstration of spring-mass-damper system.	35
FIGURE 4.3: Example simulation scenario featuring a “leader” and “follower” agent.	40
FIGURE 4.4: Monte Carlo simulation results given parameters specified in Table 4.1.	43
FIGURE 4.5: Examples of adaptive patterns.	44
FIGURE 5.1: Examples of commercially available Autonomous Surface Vessels for environmental monitoring.	46
FIGURE 5.2: Rendering of the UNCC ASV platform.	48
FIGURE 5.3: Communication packet layout for recommended minimum specific GNSS data.	51

FIGURE 5.4: Single beam echosounder operation.	53
FIGURE 5.5: High-level connectivity diagram for the ASV.	55
FIGURE 5.6: Voltage divider for battery sensing.	56
FIGURE 5.7: Complete Pelican case with labeled components.	57
FIGURE 5.8: Deconstructed ASV for transport and storage.	58
FIGURE 5.9: ASV propulsion system and framing.	59
FIGURE 5.10: Final ASV build excluding sensor package.	60
FIGURE 5.11: Data communication packet description between autonomy and control computers.	64
FIGURE 5.12: Example network topology for two agent network.	69
FIGURE 5.13: Ground station user interface for monitoring sensor network.	71
FIGURE 5.14: Wireless communication range testing paths.	75
FIGURE 5.15: Signal strength over distance.	76
FIGURE 5.16: Multi-agent testing on Hechenbleikner Lake.	77
FIGURE 5.17: Maximum speed testing with remote control.	77
FIGURE 5.18: PID controller testing results.	78
FIGURE 5.19: Waypoint control testing on Hechenbleikner Lake	79
FIGURE 6.1: Waypoint control testing and bathymetric mapping near the Lake Norman Community Sailing Center.	81
FIGURE 6.2: Lake Norman experimental semivariogram.	82

FIGURE 6.3: Estimate of bathymetry given echosounder measurements using kriging. The origin corresponds to the southwest corner of the white box in Fig. 6.1.	83
FIGURE 6.4: Experimental results from the first trial.	87
FIGURE 6.5: Experimental results from the second trial.	88
FIGURE 6.6: Experimental results from the third trial.	89
FIGURE 6.7: Experimental trial comparison.	90
FIGURE A.1: Control paths for experimental comparison.	100
FIGURE A.2: Topographical map of Lake Norman [51].	100
FIGURE B.1: Data flow diagram for ASV onboard software.	101
FIGURE B.2: Data flow diagram for the ground station.	102
FIGURE B.3: Detailed ASV wiring diagram.	103

## CHAPTER 1: INTRODUCTION

Mobile sensor networks provide an automated, cost-effective, and scalable means for spatial data collection in applications such as precision agriculture [1, 2] and environmental monitoring [3]. Path planning algorithms maximize efficiency and information gain during data collection tasks by optimizing sampling paths while considering the dynamics and sensing capabilities of the mobile robots and a model of the spatial process of interest. In coverage path planning [4–8] the aim is for a multi-robot system to efficiently cover an area of interest with uniform sampling (e.g., using lawnmower-type paths). Alternatively, adaptive sampling (AS) algorithms [9] (also referred to as informative path planning [10]) enable an online estimate of a spatio-temporal process to guide the collection of subsequent measurements to maximize information gain. In the adaptive sampling framework, robots adjust their sampling paths during deployment to allocate more sampling effort to regions of greater variability and/or uncertainty.

Adaptive sampling algorithms often use Gaussian processes (GPs) to model continuous spatio-temporal attributes of interest. A GP is a random process characterized by a mean and a covariance function that describes the smoothness of the underlying attributes. GP regression (known as *kriging* in the geosciences [11]) enables optimal estimation of spatio-temporal attributes in unsampled locations as a weighted linear combination of existing measurements. However, the time-complexity of standard

GP regression is  $O(N^3)$  [12, ch.8] where  $N$  is the number of samples; thus, it is only practical to perform a regression with a relatively small number of samples for robotic systems with limited onboard computation.

In its basic form, observation of a spatial field using a sensor network is achieved through the dispersal of sensors throughout the region of interest. For static sensor networks, this dispersal is a significant field of study. The sensor placement problem has been addressed through Mixed-Integer Linear Programming (MILP) problems over discrete point sets [13],[14] and minimization of sensor coverage overlap [15]. This problem has been extended to the case where sensors have limited movement, allowing them to reach a terminal resting location unique from their starting point. The Minimax method is an algorithm developed for the filling of coverage gaps using these mobile sensors [16]. Alternative methods rely on virtual forces and potential fields to draw or repel sensors to optimal sensing locations [17]. However, these methods are offline and rely on geometric interactions between sensor locations rather than information relating to the sensed field.

Mobile sensor networks are of particular interest given their ability to adaptively configure themselves to an evolving understanding of the environment which they are observing [18],[19] and their coverage is advantageous to that of static or mostly-static sensor networks [20]. These networks consist of individual agents capable of short-range communication and limited computational power. Each agent can move independently towards a collective goal which is typically related to the improvement of the estimation of a scalar field. Recent work has focused on the path planning problems associated with the motivation of agents through an unknown space [21–

25]. In [21], an RRT\* algorithm, termed MDMI-RRT\*, is used to coordinate a decentralized mobile network of Autonomous Surface Vessels (ASVs) towards the estimation of a scalar field. However, the MDMI-RRT\* algorithm does not consider agent dynamics or mission constraints. Singh et. al. [23] develop an informative path planning algorithm which utilizes mutual information as a metric for coverage of the field. Time optimality and sub-approximation guarantees are derived for the proposed algorithm, but sampling locations must be known *a priori* and additional field discretization is required. In [22] and [25], Gaussian process regression was used to estimate the scalar field and paths are generated to minimize the estimate variance over the field.

Coverage path planning (CPP) [4] is the task of observing all points within an area or volume of interest. In the past, CPP problems were treated much like the static sensor placement problem. The alteration being that a single mobile sensor would treat those static sensor locations as waypoints rather than actual sensors residing there. As its name suggests, this research typically assumes that sufficient time and energy are available to the agent in order to achieve coverage of a given field. However, recent work has considered energy constraints for CPP problems using heuristics to allow for limited battery life of the agent [5],[6]. However, both of these works assume that an easily accessible “charging station” is available to allow the agent to complete the coverage in horizons. The energy constrained CPP problem is augmented using the virtual potential field approach to sensor placement in [7]. A single agent traverses a set of waypoints which themselves take the place of static sensors in the potential field while using a distance constraint to restrict the

overall spacing between consecutive waypoints. However, this work only considers a single agent and constructs the potential field using geometric constraints rather than information about a scalar field, meaning it is inherently an offline algorithm.

This thesis seeks to address the current shortcomings in the consideration of heterogeneous agent dynamics and sensing in the study of adaptive sampling using mobile sensing networks. By assuming the scalar field is modeled by a Gaussian process, an informed estimate of the field based upon progressive noise-corrupted measurements is combined with the potential field method of coverage path planning to produce a multi-agent adaptive sampling algorithm. This thesis will consider the individual dynamics of each agent as well as differences in sensing quality to generate informative paths over unknown scalar fields.

The contributions of this thesis are (1) an efficient Gaussian process regression framework to fuse measurements from multiple agents with heterogeneous measurement noise using an adaptive truncation with common data neighborhoods, (2) an adaptive sampling approach that considers heterogeneous vehicle dynamics and sensing to allocate sampling trajectories, and (3) demonstration and comparison of the approach through Monte Carlo simulation and field experiments involving two autonomous surface vessels (ASVs) mapping the bathymetry of a freshwater lake.



## CHAPTER 2: PRELIMINARIES

The following chapter describes mathematical preliminaries which form the foundation upon which the research is constructed. A brief introduction regarding the statistical tools and geometric constructs guides the reader to a position in which the content of the work can be readily understood. Additional resources are provided as needed. Throughout this thesis, a lower case, bold variable, e.g.,  $\mathbf{x}$ , represents a vector while an upper case, bold variable, e.g.,  $\mathbf{X}$ , represents a matrix.

### 2.1 Gaussian Processes

Gaussian processes have been regularly used for the simulation and modeling of environmental phenomena in the sciences [26–28]. With few assumptions about the actual characteristic being modeled, this statistical tool provides an estimate and associated uncertainties that are valuable in the exploration of unknown fields. These benefits and a proven record in field estimation within the geosciences field have made this tool attractive for robotic application given its ability to predict spatial characteristics in real time [29].

A random process is an infinite collection of joint random variables [30]. If the variables associated with the process have a joint Gaussian distribution, this is referred to as a Gaussian process (GP). A Gaussian process can be denoted as

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \quad (2.1)$$

where the collection of random variables comprising  $f(\mathbf{x})$  for  $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$  have a mean function  $\mu(\mathbf{x})$  and covariance function  $k(\mathbf{x}, \mathbf{x}')$  where

$$\begin{aligned}\mu(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))],\end{aligned}\tag{2.2}$$

where  $\mathbb{E}[\cdot]$  is the expected value of a random variable.

A useful subset of Gaussian processes are those which are second-order stationary with zero mean [12]. This classification restricts the mean of the process to be zero for all points,  $\mu(\mathbf{x}) = 0$ , and the covariance function to be invariant under translation,

$$k(\mathbf{x}, \mathbf{x} + \mathbf{h}) = \mathbb{E}[f(\mathbf{x}) f(\mathbf{x} + \mathbf{h})],\tag{2.3}$$

where  $\mathbf{h}$  is an offset from the location vector, known as lag. Given the invariance under translation, the covariance function becomes a function of  $\mathbf{h}$  only

$$k(\mathbf{h}) \triangleq \mathbb{E}[f(\mathbf{x}) f(\mathbf{x} + \mathbf{h})].\tag{2.4}$$

### 2.1.1 Covariance Functions

Covariance functions define a sense of smoothness upon the GP. The amount of variation as well as the frequency of variation are defined by relating points in the sample space with one another. The covariance function, or *kernel*, used within this thesis is the isotropic Gaussian covariance function

$$k(\mathbf{h}) = \sigma_0^2 e^{-3(\|\mathbf{h}\|\omega^{-1})^2}, \quad (2.5)$$

where  $\sigma_0^2$  is the variance of each composite random variable and  $\omega$  is the length-scale of the GP. Note that the factor of three in the exponent ensures that the value of the covariance function is 0.05 when  $\|\mathbf{h}\| = \omega$ , a standard feature in kriging for geosciences [31].

### 2.1.2 Conditioning of Gaussian Processes

A sampling of a GP is a deterministic function of space or time. The likelihood of selecting a particular sample function from the GP is determined by the mean and covariance functions. As the number of observations within the training set increases, the process can be conditioned to reduce the subset of possible sample functions. Figure 2.1 visualizes the progressive addition of information and its effect on the confidence bounds and allowable sample functions.

Additionally, uncertainty in observation quality can be incorporated into the conditioning of the GP. This allows for consideration of sensor measurement noise variance and its effect as shown in Fig. 2.2.

### 2.1.3 Simulation of Gaussian Processes

Gaussian processes can be simulated via two main methods: the linear combination of the square-root of the covariance matrix and a random vector or circulant embedding [32, ch.12.2]. The following is a short introduction into the technical details of the former as it relates to the simulation of two-dimensional GPs used within this thesis.

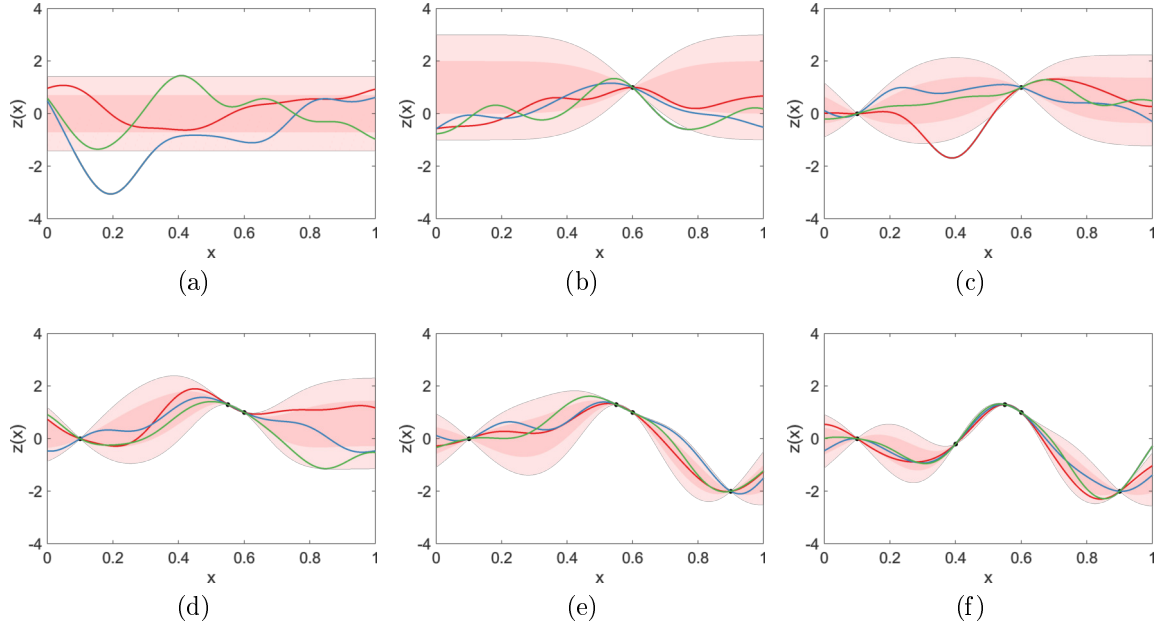


Figure 2.1: Gaussian process conditioning with addition of noise-free observations. A set of three sample functions are generated at random from the one-dimensional zero-mean Gaussian process with  $\sigma_0^2 = 0.5, \omega = 0.3$ . The inner red band represents  $\pm\sigma$  with the outer representing  $\pm 2\sigma$ . Panel (a) is an unconditioned GP. Panels (b)-(f) shown progressive addition of observations (block dots) and the corresponding effect on sample functions and GP variance.

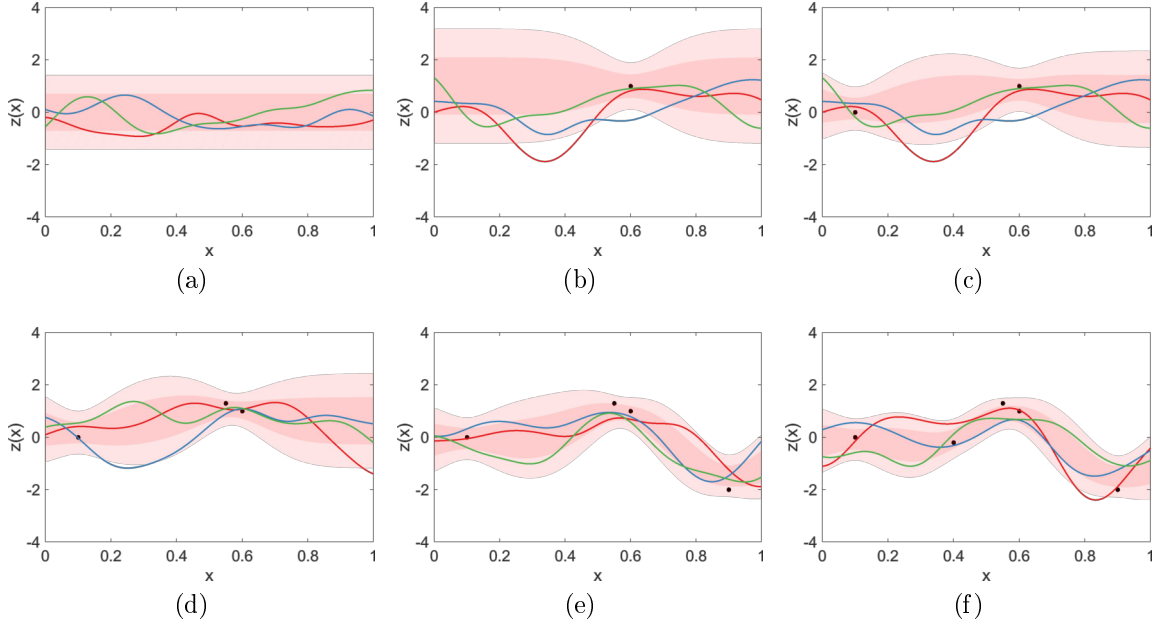


Figure 2.2: Progressive Gaussian process conditioning with addition of noisy observations.

Let  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$ , where  $\mathbf{p} \in \mathcal{Q} \subset \mathbb{R}^2$  be the set of user-defined spatial points at which the GP is evaluated. A mean vector,  $\boldsymbol{\mu} = (\mu(\mathbf{p}_1), \dots, \mu(\mathbf{p}_N))^T$ , and covariance matrix with elements  $\Sigma_{i,j} = k(\mathbf{p}_i, \mathbf{p}_j)$ ,  $i, j = 1, 2, \dots, N$  [32, ch.12.2] are defined for the selected evaluation points. Using Choleksy decomposition, the matrix square root of the covariance matrix is found,  $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}^T$ . Let  $\boldsymbol{\nu}$  be the random vector with  $\nu_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$  for  $i = \{1, 2, \dots, N\}$ . Finally, the simulated GP is found by

$$\mathbf{X} = \boldsymbol{\mu} + \mathbf{A}\boldsymbol{\nu}. \quad (2.6)$$

While Choleksy decomposition is the main method used for simulation of GPs within this thesis, it is not the only available method. The matrix square root,  $\mathbf{A}$  can be derived through spectral decomposition such that  $\mathbf{A} = \Re(\mathbf{V}\sqrt{\mathbf{D}})$ , where

$\mathbf{D}$  is the diagonalized matrix of eigenvalues,  $\sqrt{\mathbf{C}}$  is the element-wise square root of matrix  $\mathbf{C}$ ,  $\mathbf{V}$  is the matrix of eigenvectors, and  $\Re(\mathbf{P})$  is the real component of each element of matrix  $\mathbf{P}$ .

The method of circulant embedding is an efficient method of simulating stationary Gaussian processes by embedding the covariance matrix into a block circulant matrix, with each block being circulant [32, ch. 12.2.2]. The matrix square root,  $\mathbf{A}$ , is then calculated using the fast Fourier transform and simulation of the field happens in accordance with (2.6).

Figure 2.3 shows the comparison between various simulation methods where the histogram displays the occurrence over 100 trials. The method of circulant embedding is shown to be significantly faster than the method using Cholesky decomposition. However, it is apparent that Cholesky decomposition method is more capable of reliably reproducing the desired mean and standard deviation for the simulation field.

## 2.2 Kriging

Gaussian processes can be used in conjunction with training data sets in order to make predictions about points of interest in the sample space in which the GP is defined. A type of GP prediction developed for use in geostatistics is known as *kriging* [12]. Kriging estimation comes in numerous variations; however, this discussion will be limited to *ordinary* kriging as it is the sole method used in this thesis. Ordinary kriging assumes a constant, unknown mean and models the residuals of the observation from the mean as a GP, rather than the observations themselves [33]. Ordinary kriging is known as a Best, Linear Unbiased Estimator (BLUE) because its formulation seeks to minimize mean square error using a weighted combination

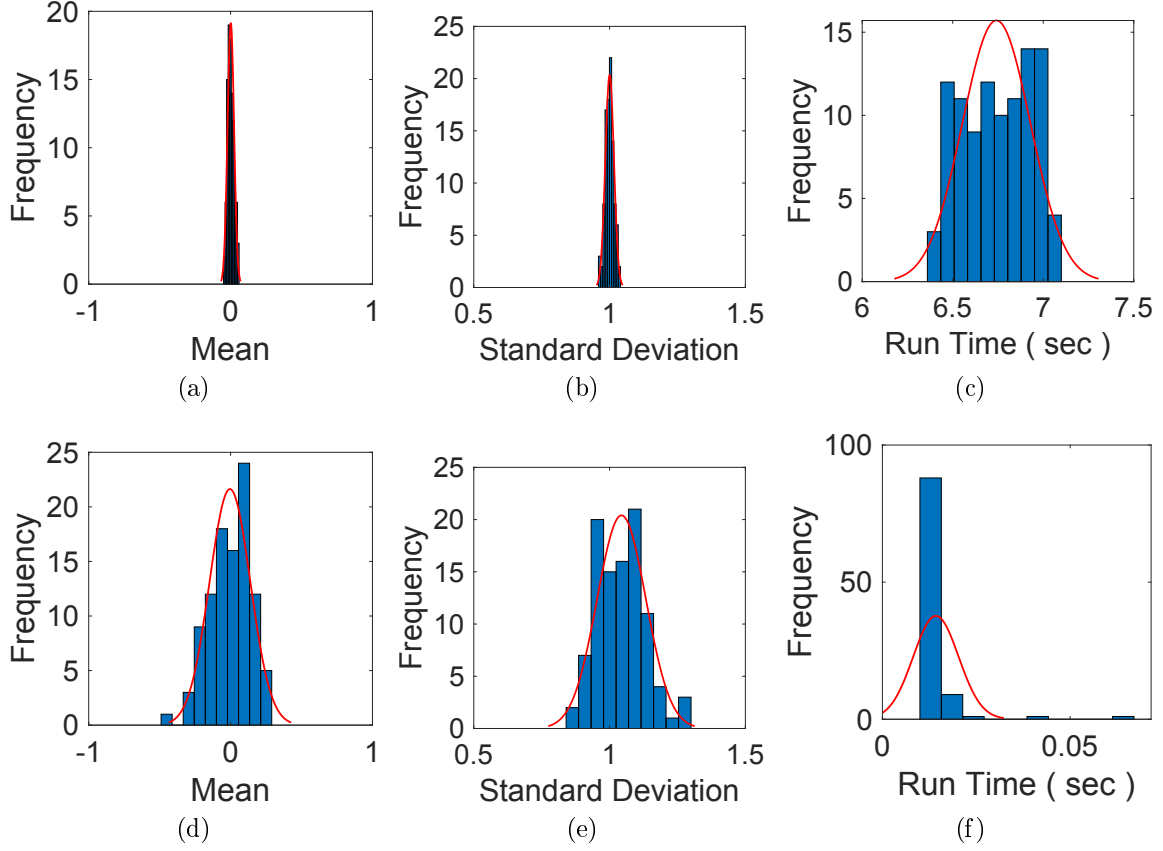


Figure 2.3: Comparison of Gaussian process simulation methods. All simulated GPs are zero-mean, two-dimensional, and use the Gaussian covariance function (2.5) with  $\sigma_0^2 = 1$ ,  $\omega = 0.1$ . Subplots (2.3a)–(2.3c) are histograms of means, variance and run times respectively using Cholesky decomposition for simulation. Subplots (2.3d)–(2.3f) mirror the above for circulant embedding simulation. All statistics were gathered over 100 trails using a  $51 \times 51$  grid of evaluation points with a fit Gaussian distribution depicted as a red line. While circulant embedding is shown to be much faster, Cholesky decomposition is shown to produce more reliable target ensemble statistics.

of observations to represent estimated points.

Let  $\mathbf{Z} = [z_1, \dots, z_N]^\top$  be partial realizations of the random function at spatial locations  $\mathbf{X} = \{\mathbf{x}_i \mid i = 1, 2, \dots, N\}$ . A detailed version of the derivation of the ordinary kriging estimator through the minimization of expected estimation error via Lagrangian multipliers is detailed explicitly in [31, ch.4]; therefore, only the results will be presented.

The matrix equation for determination of optimal weights for ordinary kriging,  $\boldsymbol{\lambda}_{OK}$  and Lagrange multiplier,  $\mu_{OK}$  [34], for a desired point of estimation,  $\mathbf{x}_0$ , using ordinary kriging is

$$\begin{bmatrix} \boldsymbol{\lambda}_{OK} \\ \mu_{OK} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{OK}(\mathbf{X}) \mathbf{Z} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{v}(\mathbf{X}, \mathbf{x}_0) \\ 0 \end{bmatrix}, \quad (2.7)$$

where the elements of the covariance vector are  $\mathbf{v}_i = k(\mathbf{x}_i, \mathbf{x}_0)$ ,  $\mathbf{1}$  is the  $N$  element column vector of all ones, and the modified covariance matrix is

$$\boldsymbol{\Sigma}_{OK} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_1) & \dots & k(\mathbf{x}_N, \mathbf{x}_1) & 1 \\ k(\mathbf{x}_1, \mathbf{x}_2) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_N, \mathbf{x}_2) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k(\mathbf{x}_1, \mathbf{x}_N) & k(\mathbf{x}_2, \mathbf{x}_N) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{bmatrix}. \quad (2.8)$$

By solving (2.7) for the optimal weights associated with each observation related to the point of estimation, the estimate and estimate error variance are



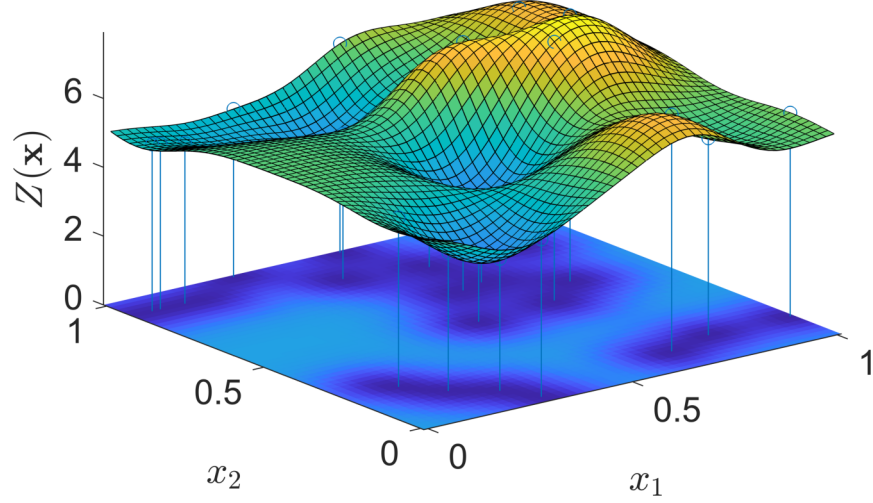


Figure 2.4: Two-dimensional field estimation using ordinary kriging. The surface representing  $Z(\mathbf{x})$  is characterized by semivariogram (2.11) with  $\boldsymbol{\theta} = [\zeta, \omega, \sigma_0^2]^\top = [0, 0.3, 1]^\top$  and  $\mu = 5$ . A total of 20 observations, represented by projections from the surface to the  $\mathbf{x}$ -plane, comprise the training set. The coloring on the  $\mathbf{x}$ -plane indicates estimation variance; the darker the color, the lower the relative variance. At locations where measurements were taken, the variance equals zero.

$$\hat{Z}_{OK}(\mathbf{x}_0) = \mathbf{Z}^\top \boldsymbol{\lambda}_{OK} \quad (2.9)$$

$$\sigma_{OK}^2(\mathbf{x}_0) = k(\mathbf{0}) - \mathbf{v}(\mathbf{X}, \mathbf{x}_0)^\top \boldsymbol{\lambda}_{OK}. \quad (2.10)$$

Repeating this point estimation for various spatial locations creates a field estimate of underlying GP with user defined resolution. Figure 2.4 shows the result of kriging over a set of uniformly spaced sampling points.

### 2.2.0.1 Kriging with Heterogeneous Measurement Error Variance with an Unknown Covariance Function

When sampling real-world scalar fields, the exact covariance function can only be estimated. For this reason, users of kriging tend to substitute the covariance function with the semivariogram,  $\gamma$ , a measure of dissimilarity of measurements given spatial distance [33, ch.2.6]. For GPs which are second-order stationary, the covariance and semivariogram are related by  $\gamma(\mathbf{h}) = k(\mathbf{0}) - k(\mathbf{h})$  [31]. The semivariogram can be easily estimated through a process of fitting a curve to a scatter-plot of the average variation in measurements over a given distance [31]. The resulting fit curve is termed the experimental semivariogram,  $\hat{\gamma}$ . The semivariogram model used in this thesis follows from (2.5),

$$\hat{\gamma}(\|\mathbf{h}\|; \boldsymbol{\theta}) = \zeta (1 - \delta_{\|\mathbf{h}\|}) + \sigma_0^2 \left(1 - e^{-3(\|\mathbf{h}\|\omega^{-1})^2}\right), \quad (2.11)$$

where  $\delta_{\|\mathbf{h}\|} = 1$  when  $\|\mathbf{h}\| = 0$ , and  $\delta_{\|\mathbf{h}\|} = 0$  otherwise, and the vector of hyperparameters is  $\boldsymbol{\theta} = [\zeta, \omega, \sigma_0^2]^\top$  comprising the “nugget”,  $\zeta$ , which accounts for micro-scale variation and sensor measurement error, the length-scale [35],  $\omega$ , and the large-scale variance of the field,  $\sigma_0^2$ .

#### 2.2.1 Heterogeneous Measurement-Error Filtered Kriging

Heterogeneous measurement-error filtered kriging (*HFK*) extends ordinary kriging to the case of site-specific measurement error with known variance [34]. Let

$$\tilde{Y}(\mathbf{x}_i) = Z(\mathbf{x}_i) + \epsilon(\mathbf{x}_i), \quad (2.12)$$

represent the noise-corrupted measurement of the attribute  $Z$  at location  $\mathbf{x}_i$  where  $\epsilon(\mathbf{x}_i)$  is the realization of a Gaussian random variable zero-mean and site-specific measurement noise variance associated with the observation,  $\sigma_\eta^2(\mathbf{x}_i)$ . The vector of noise-corrupted observations shall be  $\tilde{\mathbf{Y}}(\mathbf{X})$ . This site-specificity will be leveraged to account for heterogeneity among agents by associating measurement locations with the agent which performed the observation. The measurement noise variance from each agent is then applied to their corresponding measurement locations which can then be handled abstractly as site-specific noise. Christensen [34] proposed to account for noisy measurements (2.12) using the  $HFK$  estimator  $\hat{Z}_{HFK}(\mathbf{x}_0) = \boldsymbol{\lambda}_{HFK}^\top \tilde{\mathbf{Y}}(\mathbf{X})$  with variance  $\hat{\sigma}_{HFK}^2(\mathbf{x}_0) = \tilde{\boldsymbol{\gamma}}(\mathbf{X}, \mathbf{x}_0)^\top \boldsymbol{\lambda}_{HFK}$  where the optimal kriging weights,  $\boldsymbol{\lambda}_{HFK}$ , and mean,  $\mu_{HFK}$ , are found from

$$\begin{bmatrix} \boldsymbol{\lambda}_{HFK} \\ \mu_{HFK} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\Gamma}}(\mathbf{X}) & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\boldsymbol{\gamma}}(\mathbf{X}, \mathbf{x}_0) \\ 1 \end{bmatrix}. \quad (2.13)$$

In (2.13),  $\tilde{\boldsymbol{\Gamma}}(\mathbf{X})$  and  $\tilde{\boldsymbol{\gamma}}(\mathbf{X}, \mathbf{x}_0)$  are augmented semivariogram matrices and vectors, respectively. The matrix  $\tilde{\boldsymbol{\Gamma}}(\mathbf{X})$  is defined by augmenting  $\boldsymbol{\Gamma}_{ij}$  with the average of the site specific variance along off-diagonal elements [34]

$$\tilde{\boldsymbol{\Gamma}}(\mathbf{x}_i, \mathbf{x}_j) = \hat{\gamma}_Z(\mathbf{h}_{ij}) + (1 - \delta_{\|\mathbf{h}_{ij}\|}) \frac{\sigma_\eta^2(\mathbf{x}_i) + \sigma_\eta^2(\mathbf{x}_j)}{2}. \quad (2.14)$$

Additionally,  $\tilde{\boldsymbol{\gamma}}(\mathbf{X}, \mathbf{x}_0)$  is defined by augmenting the  $i$ th element of the semivariogram

vector with half the site specific variance of the observation [34]

$$\tilde{\gamma}(\mathbf{x}_i, \mathbf{x}_0) = \hat{\gamma}_Z(\mathbf{h}_{i0}) + \frac{\sigma_\eta^2(\mathbf{x}_i)}{2}. \quad (2.15)$$

In (2.14) and (2.15), the estimated semivariogram  $\hat{\gamma}_Z$  is obtained from the noise-corrupted estimated semivariogram  $\hat{\gamma}_{\tilde{Y}}$  by subtracting the average of the measurement-error variance across all  $M$  sampling locations from the nugget estimated using the noisy measurements,  $\zeta_{\tilde{Y}}$ :

$$\hat{\gamma}_Z(\mathbf{x}_i, \mathbf{x}_j; \zeta_Z, \omega, \sigma_0^2) = \hat{\gamma}_{\tilde{Y}}\left(\mathbf{x}_i, \mathbf{x}_j; \zeta_{\tilde{Y}} - \frac{1}{M} \sum_{l=1}^M \sigma_\eta^2(\mathbf{x}_l), \omega, \sigma_0^2\right). \quad (2.16)$$

With the addition of site-specific measurement error, the noisy Gaussian process  $\tilde{Y}$  is no longer second-order stationary since the noise term's dependence on location violates the assumption that the kernel is invariant under translation (2.4).

### 2.3 Centroidal Voronoi Tessellation

This work uses centroidal Voronoi tessellations (CVTs) to identify high-value sampling locations. Recall that a standard Voronoi tessellation (VT) partitions a planar region,  $\mathcal{Q}$ , into a set containing  $k$  non-intersecting polygonal cells,  $\{V_i\}_{i=1}^k$ , whose union equals  $\mathcal{Q}$ . Given a set of generating points  $\{\mathbf{g}_i\}_{i=1}^k \in \mathcal{Q}^k$ , the Voronoi cell  $V_i$  is the set of points within  $\mathcal{Q}$  that are closer to the cell's generating point,  $\mathbf{g}_i$ , than any other generating point [36, ch.5]:

$$V_i = \{\mathbf{x} \in \mathcal{Q} \mid \|\mathbf{x} - \mathbf{g}_i\| \leq \|\mathbf{x} - \mathbf{g}_j\|\} \quad (2.17)$$

for all  $j = \{1, 2, \dots, k\}$  with  $j \neq i$ . When a density function is defined over the domain  $\rho : \mathcal{Q} \rightarrow \mathbb{R}$ , each cell has a mass

$$M_{V_i} = \int_{\mathbf{x} \in V_i} \rho(\mathbf{x}) d\mathbf{x}, \quad (2.18)$$

and centroid

$$\mathbf{c}_{V_i} = \frac{1}{M_{V_i}} \int_{\mathbf{x} \in V_i} \mathbf{x} \rho(\mathbf{x}) d\mathbf{x}. \quad (2.19)$$

CVTs [37] are variants of VTs in which the generating points of the VT are co-located with the associated cell's centroid. In this work, the CVT algorithm will be used to adaptively identify high-priority waypoints to sample. The equations (2.18)–(2.19) will be discretized and the integrals are replaced with summations over a uniform grid. The CVT is computed using a variant of Lloyd's method [38].

## CHAPTER 3: EFFICIENT GAUSSIAN PROCESS REGRESSION WITH HETEROGENEOUS MEASUREMENT NOISE

In this chapter, a modified kriging approach is proposed to account for agents with heterogeneous measurement noise variance and to improve computational efficiency.

### 3.1 Filtered Kriging with Heterogeneous Sensors

Consider a mobile sensor network of  $N$  agents that measure a spatial attribute  $Z$  with heterogeneous measurement variance  $\sigma_{\eta,i}^2$  for  $i = 1, \dots, N$ . This approach models a team of mobile robots with agents that have different sensing modalities or types/quality of sensors. In this thesis, an approach is proposed to specialize heterogeneous measurement-error filtered kriging (HFK, see Sec. 2.2.1) to this case by replacing the site-specific measurement variance with an agent-specific one. Suppose that each agent collects measurements at the same sampling rate and begins to sample at the same time. The total number of measurements from each agent is equal and all measurements taken from all agents are used to populate the set  $\mathbf{Z}$ . An indicator function,  $\beta(\mathbf{x})$ , is defined that returns the measurement variance  $\sigma_{\eta}^2$  associated with the agent that took the observation at  $\mathbf{x}$ , assuming no two samples are co-located. Thus, (2.14) becomes  $\tilde{\mathbf{\Gamma}}(\mathbf{x}_i, \mathbf{x}_j) = \hat{\gamma}_Z(\mathbf{h}_{ij}) + 0.5(1 - \delta_{\|\mathbf{h}_{ij}\|})[\beta(\mathbf{x}_i) + \beta(\mathbf{x}_j)]$ . Similarly, (2.15) is modified as  $\tilde{\gamma}(\mathbf{x}_i, \mathbf{x}_0) = \hat{\gamma}_Z(\mathbf{h}_{i0}) + 0.5(\beta(\mathbf{x}_i))$ , while (2.16) becomes

$$\hat{\gamma}_Z(\mathbf{x}_i, \mathbf{x}_j) = \hat{\gamma}_{\tilde{Y}}\left(\mathbf{x}_i, \mathbf{x}_j; \zeta_{\tilde{Y}} - \frac{1}{N} \sum_{i=1}^N \sigma_{\eta,i}^2, \omega, \sigma_0^2\right). \quad (3.1)$$

### 3.2 Adaptive Spatial Truncation

The time complexity of kriging is dominated by the inversion of the semivariogram matrix in (2.7). To improve efficiency, spatial truncation can be used to reduce the size of this matrix by considering only nearby measurements with strong correlations to an estimation point and rejecting measurements that have little influence [31],[39]. Typical truncation methods rely on geometric selector regions (squares, circles, etc.) centered about an estimation point to capture relevant measurements. This strategy is effective when there is a sufficient number of measurements around the estimation point [39]; however, it performs poorly in sparsely sampled regions (e.g., at the start of a mission or near the boundary of the field).

To address these challenges, an adaptive method of observation truncation is proposed wherein a standard rectangular geometric selector is used if it contains a threshold  $M_{\min}$  number of measurements, and otherwise a nearest-neighbor selector is used guarantee a minimum number of measurements (i.e., by considering measurements outside the geometric selector, if needed). The geometric selector is denoted  $G_R(\mathbf{x}, \mathbf{X}; w_G, h_G) = \{\mathbf{x}_i \in \mathbf{X} \mid \mathbf{x}_i \in R(\mathbf{x}; w_G, h_G)\}$  for all  $i = 1, 2, \dots, M$ , where  $R(\mathbf{x}; w, h) \subset \mathbb{R}^2$  denotes a rectangular area centered on  $\mathbf{x} = (s_i, s_j) \in \mathbb{R}^2$  with width  $w$  and height  $h$ . The nearest-neighbor selector is denoted  $G_{\text{NN}}(\mathbf{x}, \mathbf{X}; M_{\min}) \subset \mathbf{X}$  and it selects the subset of at most  $M_{\min}$  measurement locations that are nearest-

neighbors to  $\mathbf{x}$  [40]. The adaptive selector is then

$$G(\mathbf{x}, \mathbf{X}; w_G, h_G, M_{\min}) = \begin{cases} G_R(\mathbf{x}, \mathbf{X}; w_G, h_G) & \text{if } |G_R(\mathbf{x})| > M_{\min} \\ G_{\text{NN}}(\mathbf{x}, \mathbf{X}; M_{\min}) & \text{if } |G_R(\mathbf{x})| \leq M_{\min} \end{cases} \quad (3.2)$$

where  $|G_R(\mathbf{x})|$  is the number of measurements in the geometric selector.

### 3.3 Common Data Neighborhood Discretization

Spatial truncation reduces the size of matrices required for inversion during the estimation process. However, it still requires inverting unique semivariogram matrices for every point to be estimated (since the geometric selector moves with the estimation point). Given many points of interest, the computation time of inverting many truncated matrices may exceed inverting the original (full measurement set) semivariogram. To reduce the number of required matrix inversions for kriging we adopt the common data neighborhood (CDN) approach proposed in [41]. A group of estimation points are assigned a common semivariogram matrix based on nearby measurements (i.e., a fixed adaptive selector is used for multiple nearby estimation points). This approach is implemented as follows.

Define a uniform grid of points with  $m$  columns and  $n$  rows  $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{m \cdot n}\}$ , where  $\mathbf{p}_i \in \mathcal{Q}$  for all  $i = \{1, 2, \dots, m \cdot n\}$ . The set of common data neighborhoods,  $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_{N_D}\}$ , is defined as a collection of  $N_D$  disjoint rectangular regions whose union covers the entire space,  $\mathcal{Q}$ . Each data neighborhood  $\mathcal{D}_i \triangleq R(\mathbf{s}_i; w_{\mathcal{D}}, h_{\mathcal{D}})$  is centered at a point  $\mathbf{s}_i \in \mathbb{R}^2$  and is defined by a width  $w_{\mathcal{D}}$  and height  $h_{\mathcal{D}}$ , and  $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$  for all  $i \neq j$  with  $i, j = 1, 2, \dots, N_D$ . Let  $G_i \triangleq G(\mathbf{s}_i, \mathbf{X}; w_G, h_G, M_{\min})$



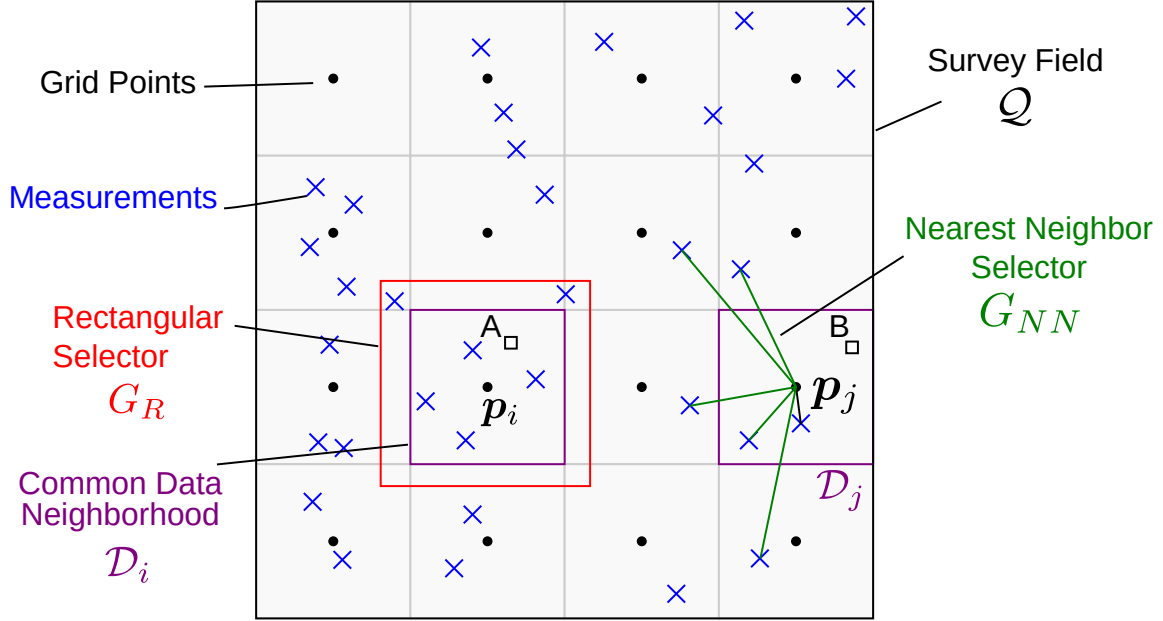


Figure 3.1: Example of the proposed adaptive truncation strategy with  $M_{\min} = 5$ . Black circles are grid points, and blue ‘x’s are measurement locations. Two CDNs  $\mathcal{D}_i$  and  $\mathcal{D}_j$ , centered on grid points  $\mathbf{p}_i$  and  $\mathbf{p}_j$ , respectively, are highlighted in purple. When estimating a point  $A$  in  $\mathcal{D}_i$  the corresponding rectangular selector contains more than  $M_{\min}$  measurements and these measurements are used to define  $\tilde{\mathbf{T}}$  in (2.13). When estimating a point  $B$  in  $\mathcal{D}_j$  the nearest-neighbor selector is used to define  $\tilde{\mathbf{T}}$  instead since insufficient measurements are located within the corresponding rectangular selector.

be the adaptive selector (3.2) for data neighborhood  $\mathcal{D}_i$ , with  $w_G \geq w_{\mathcal{D}}$  and  $h_G \geq h_{\mathcal{D}}$ .

### 3.4 Heterogeneous, Common-Data-Neighborhood Kriging

Our estimation approach combines *HFK* specialized to heterogeneous agents, adaptive selectors, and CDNs. Suppose that an estimation point lies with the  $i$ th CDN,  $\mathbf{x}_0 \in \mathcal{D}_i$ . For all such estimation points, the *HFK* estimator (2.13) is used with a subset of observations  $\mathbf{X}$  given by the adaptive selector (3.2). Importantly, this same truncated data set is used for *all* estimation points in  $\mathcal{D}_i$  and hence the inversion in (2.13) need only occur once for all such points.

This work proposes to modify [41] to utilize the adaptive spatial truncation (§3.2) and to incorporate heterogeneous measurement-error filtered kriging (§2.2.1). The kriging weights and mean for the point  $\mathbf{x}_0 \in \mathcal{D}_i$  are

$$\begin{bmatrix} \boldsymbol{\lambda}_{HC} \\ \mu_{HC} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{\Gamma}}(\mathbf{G}_i) & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\boldsymbol{\gamma}}(\mathbf{G}_i, \mathbf{s}_i) \\ 1 \end{bmatrix}, \quad (3.3)$$

and the estimate and variance are denoted with a subscript  $HC$ ,

$$\hat{Z}_{HC}(\mathbf{x}_0) = \boldsymbol{\lambda}_{HC}^\top \tilde{\mathbf{Y}}(\mathbf{X}) \quad (3.4)$$

and

$$\hat{\sigma}_{HC}^2(\mathbf{x}_0) = \tilde{\boldsymbol{\gamma}}(\mathbf{X}, \mathbf{x}_0)^\top \boldsymbol{\lambda}_{HC}. \quad (3.5)$$

The proposed approach is amenable to parallelization and permits a trade-off between computational efficiency and accuracy by adjusting the minimum number of considered observations,  $M_{\min}$ , as well as the dimensions of the common data neighborhoods,  $w_{\mathcal{D}}, h_{\mathcal{D}}$  and the dimensions of the search neighborhoods,  $w_{\mathbf{G}}, h_{\mathbf{G}}$ . These parameters are typically chosen as integer multiples of the respective directional length-scale. Given that the fields used for this work are isotropic, the length-scale is independent of direction meaning  $w_{\mathcal{D}} = h_{\mathcal{D}}$  and  $h_{\mathbf{G}} = w_{\mathbf{G}}$ .

The proposed heterogeneous, common-data-neighborhood kriging ( $HC$ ) estimator was compared through numerical simulations to a standard  $HFK$  estimation [34] (without adaptive selectors or CDNs) and to standard ordinary kriging. The sim-

ulations consisted of generating a GP within a normalized domain  $\mathcal{Q} = [0, 1]^2$  and using hyper-parameters  $\boldsymbol{\theta} = [\zeta, \omega, \sigma_0^2]^\top = [0, 0.3, 1]^\top$ . Measurement locations were randomly selected and the observations were polluted with noise according to (2.12). For half of the measurements,  $\sigma_{\eta,1}^2 = 0.1$ , for the other half,  $\sigma_{\eta,2}^2 = 0.5$  to represent data collection by heterogeneous agents with different quality sensors. The noisy observations were processed by the HC estimator with  $h_{\mathcal{D}} = w_{\mathcal{D}} = 0.5\omega$ ,  $h_{\mathcal{G}} = w_{\mathcal{G}} = \omega$  and  $M_{\min} = 20$ . The same measurements were also used to estimate the field with the HFK estimator and the ordinary kriging estimate with what can be considered a naive approach for nugget in this scenario,  $\zeta = (0.1 + 0.5) / 2$ . Results were evaluated by comparing the average deviation of estimate from the true field on a point-to-point basis which is referred to as mapping error (ME),

$$\text{ME} = \sum_{i=1}^{|\mathcal{P}|} \left| \hat{Z}(\mathbf{x}_i) - Z(\mathbf{x}_i) \right| / |\mathcal{P}| . \quad (3.6)$$

The estimate was computed over one hundred unique realizations of the GP. For each realization, each method (HFK, HC, ordinary kriging) was used with 500, 1000, and 2000 measurements and the mean ME (3.6) and computation time were recorded. The results from this numerical study are shown in Fig. 3.2. A computation time savings of approximately 10x is achieved when comparing HC to HFK and ordinary kriging.

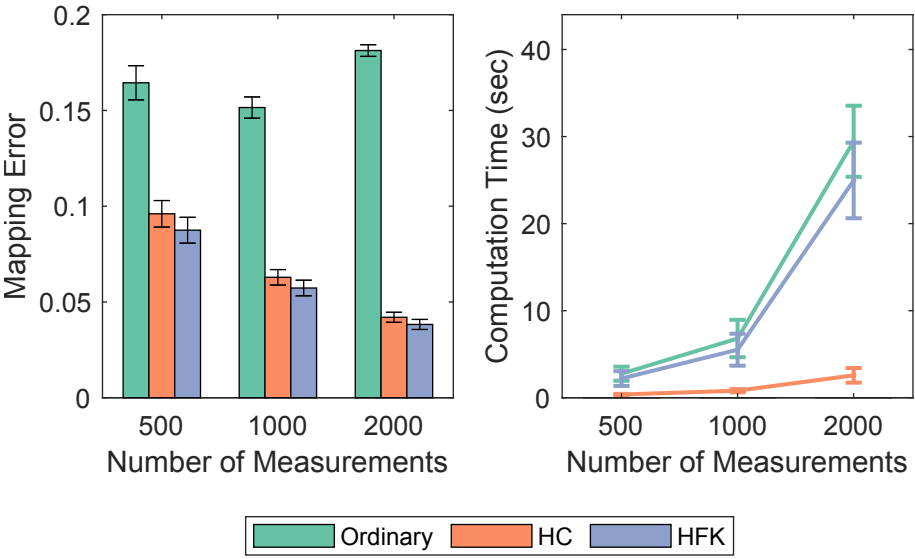


Figure 3.2: A comparison between mapping error and computation time for the proposed kriging methods when presented with noisy measurements over 300 trials. The error bands indicate plus or minus one standard deviation.

It is also shown that implementation of HFK in the event of heterogeneous measurement noise variance is superior to the naive approach of adjusting the nugget when using ordinary kriging. Finally, the accuracy reduction experienced when using HC compared to HFK is minimal and the computational efficiency gained is deemed to outweigh this slight loss in accuracy. It should be noted that the expectation of improved accuracy given increased number of measurements did not hold for standard ordinary kriging due to numerical instabilities in inverting large matrices, an effect which is pronounced in situations where measurement variation is not considered.

## CHAPTER 4: MULTI-AGENT ADAPTIVE SAMPLING

Building upon the previously described estimation methodology, this chapter utilizes the predictive power of kriging to make informed decisions about future sampling sites. The method by which multiple agents are routed within a sampling space is rooted upon the Voronoi tessellation strategy as well as fundamental principles of mechanical systems. Simulation results are provided for the validation of the proposed strategy.

### 4.1 Sampling Strategy

The method by which growing knowledge of the underlying scalar field is leveraged for path planning is founded upon a modification to the calculation of the field's centroidal Voronoi tessellation in which waypoints act as generating points. While there are numerous ways to calculate the unique centroidal Voronoi tessellation for a given set of generating points and boundary conditions, a simple method involves imposing a virtual force field which acts upon the set of points [42]. Typically, this virtual force connects a generating point only to the centroid of the corresponding Voronoi cell. A method relatable to the construction of a mechanical spring-mass-damper system in which sequential points are connected by springs and dampers to form a while connecting each waypoint via a spring and damper to to the geometric centroid of its corresponding Voronoi cell was demonstrated to be effective by Jensen-Nau [7]. Jensen-Nau utilized this method to constrain total mission distance

for a single agent. This thesis expands the aforementioned work by implementing kriging for the definition of a cost field to allow the definition of mass centroids on Voronoi cells, multi-agent support, consideration of agent dynamics and sensing characteristics, and enforcing a mission constraint of time rather than distance.

#### 4.1.1 Heterogeneous Agent Dynamics

Consider a set of  $N$  agents with heterogeneous dynamics. To illustrate the proposed approach we consider differential thrust unmanned boats, but the approach can be generalized to other vehicle models. The unmanned boats have the dynamics:

$$\begin{aligned}\ddot{x}_i &= \frac{1}{m_i} \{[u_r + u_l] \cos(\theta_i) - b_i \dot{x}_i\} \\ \ddot{y}_i &= \frac{1}{m_i} \{[u_r + u_l] \sin(\theta_i) - b_i \dot{y}_i\} \\ \ddot{\theta}_i &= \frac{L_i}{2I_i} (u_r - u_l)\end{aligned}\tag{4.1}$$

where  $(x, y) \in \mathbb{R}^2$  is the planar position relative to a fixed inertial reference frame with origin  $O$  and ortho-normal basic vectors  $\{\hat{\mathbf{i}}_1, \hat{\mathbf{i}}_2, \hat{\mathbf{i}}_3\}$ ,  $\theta_i \in [0, 2\pi)$  is the heading (see Fig. 4.1),  $m_i$  is mass,  $I_i$  is the rotational inertia about  $\hat{\mathbf{i}}_3$ ,  $L_i$  is the distance between thrusters, and  $b_i$  is a damping coefficient of the  $i$ th agent. The left and right thruster forces,  $u_l$  and  $u_r$ , respectively, are bounded:  $u_l, u_r \in [0, u_{\max}]$  where  $u_{\max}$  is a maximum thrust. The  $i$ th agent follows a reference path consisting of a set of waypoints  $\mathbf{W}_i = \{\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, \dots, \mathbf{w}_{i,N_w}\}$  where  $\mathbf{w}_{i,j} \in \mathcal{Q}$  for all  $j = \{1, 2, \dots, N_w\}$  and  $N_w$  is the number of waypoints in the path. For a given waypoint,  $\mathbf{w}_{i,j} = (x_w, y_w) \in \mathbf{W}_i$ , the homing guidance law  $\theta_d = \text{atan2}(y_w - y_i, x_w - x_i)$  determines the desired

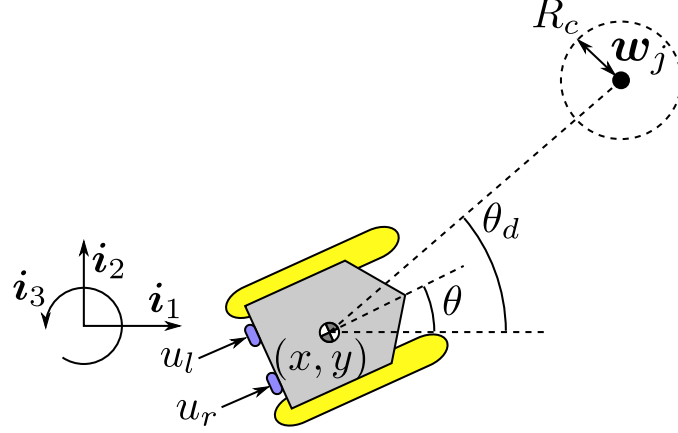


Figure 4.1: State and input diagram for waypoint tracking. The  $i$ th agent moves with heading  $\theta$  from  $i_1$  under control inputs,  $u_l$  and  $u_r$ . Tracking waypoint  $w_{i,j}$  commands the agent to point in the direction  $\theta_d$  until the agent's position  $(x_i, y_i)$  enters the capture radius of the waypoint.

heading angle, and the thrusters are commanded according to

$$\begin{aligned} u_l &= \text{sat}(\delta_v - \delta_\theta, 0, u_{\max}) \\ u_r &= \text{sat}(\delta_v + \delta_\theta, 0, u_{\max}), \end{aligned} \quad (4.2)$$

where  $\delta_v$  and  $\delta_\theta$  are the outputs of PID controllers tracking a desired speed  $v_d$  and a desired heading  $\theta_d$ , respectively, and  $\text{sat}(\kappa; a_l, a_h)$  is the saturation function that bounds an argument  $\kappa$  between a lower bound  $a_l$  and upper bound  $a_h$  such that  $a_l \leq \kappa \leq a_h$ . Once the position of the  $i$ th agent is sufficiently close to the current waypoint,  $\|[x_i, y_i]^\top - w_{i,j}\| \leq R_c$  where  $R_c \in \mathbb{R}$  is a capture radius, the next waypoint in the sequence becomes active. The closed-loop dynamics of the  $i$ th agent can be summarized as

$$\dot{s}_i = f_i(s_i, u_i(s_i); W_i), \quad (4.3)$$

where  $\mathbf{s}_i = [x_i, y_i, \theta_i, \dot{x}_i, \dot{y}_i, \dot{\theta}_i]^\top \in \mathbb{R}^5 \times [0, 2\pi)$  is the state of the system (4.1) rewritten in first-order form and  $u_i(\mathbf{s}_i)$  is the guidance and control law (4.2).

#### 4.1.2 Heterogeneous Agent Sensing

The collection of  $N$  agents may also vary in their sensing quality. The objective quality of an agent's sensing capability is inversely related to the measurement noise variance associated with its observation equipment. The measurement noise variance for agent  $i$ ,  $\sigma_{i,\eta}^2$ , characterizes the expected difference between the true and measured values of the observed scalar field.

During simulation, individual agents are assigned a measurement noise variance and all points observed by agent  $i$  are associated with  $\sigma_{i,\eta}^2$  for the purposes of performing the kriging estimate with site-specific noise (3.3). For consistency with experimental trials, it was assumed that bathymetric information comprises the scalar field during simulations. However, the methods proposed within this thesis are applicable to any scalar field which satisfies the definition of second-order stationarity, is described by a directionally independent semivariogram, and does not vary in time. Additionally, while the sampling region was assumed to be square, non-convex shapes may be considered in future works.

#### 4.1.3 Problem Statement

The goal of the adaptive sampling algorithm is to design waypoint paths for each agent that minimize the mapping error (3.6) while exploiting the heterogeneous dynamics and sensing capabilities of the team. Given a waypoint list  $\mathbf{W}_i$  for agent  $i$ , initial conditions  $\mathbf{s}_i(t_0)$ , a sensing time interval  $T_s$ , the closed-loop dynam-



ics (4.3), and a maximum mission time,  $T_m$ , the map  $\phi_i(\mathbf{W}_i; \mathbf{s}_i(t_0), t_0, T_m, T_s) = [\mathbf{q}_{i,1}, \mathbf{q}_{i,2}, \dots, \mathbf{q}_{i,M}] \in \mathbb{R}^{2 \times M}$  returns the  $i$ th agent's sampling locations arranged as columns of a matrix in chronological order where  $\mathbf{q}_{i,j} \in \mathcal{Q}$  for all  $j = 1, 2, \dots, M$ . Let the matrix  $\mathbf{X} \in \mathbb{R}^{2N \times M}$  denote the set of sampling locations across all agents at the end of the mission

$$\mathbf{X} = [\phi_1(\mathbf{W}_1)^\top, \phi_2(\mathbf{W}_2)^\top, \dots, \phi_N(\mathbf{W}_N)^\top]^\top, \quad (4.4)$$

where  $\phi_i(\mathbf{W}_i)$  is the function which converts the  $i$ th agent's trajectory given a way-point set into a set of sampling locations equally-spaced in time.

The corresponding observations made at each location in  $\mathbf{X}$  are denoted  $\tilde{\mathbf{Y}} \in \mathbb{R}^{2N \times M}$ . Since the realization of the field  $Z(\mathbf{x})$  is not known to the agents, the mapping error (3.6) cannot be computed by the agents, instead the following cost function is used for path planning

$$\mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}, \alpha) = (\alpha + \sigma_{HC}^2(\mathbf{X}; \mathbf{p})) \cdot \left| \hat{Z}_{HC}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}) \right|, \quad (4.5)$$

where  $\mathbf{p} \in \mathbb{R}^2$  is a point at which to evaluate the cost function. The first term in the product (4.5) is a quantization of the uncertainty remaining at potential sampling locations,  $\sigma_{HC}^2(\mathbf{X}; \mathbf{p})$ , added to a modifier,  $\alpha$ , whereas  $\hat{Z}_{HC}$  is the estimate of the zero-mean field which increases cost in areas that diverge greatly from the field mean (i.e., peaks or valleys). The modifying term,  $\alpha$ , prevents the points cost from reaching zero in regions that have already been sampled. This incentivizes returning to previously visited locations if time permits. In regions that have already been

sampled, the term  $(\alpha + \sigma_{HC}^2(\mathbf{X}; \mathbf{p}))$  will approach a minimum value, dependent upon  $\alpha$ , thereby reducing the cost contribution of that point compared with unsampled locations. As  $\alpha$  is increased, the incentive to return to previously sensed areas also increases.

The optimization is then:

$$\arg \min_{\mathbf{X}(\mathbf{W})} \sum_{\mathbf{p} \in \mathcal{P}} \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}, \alpha) \quad (4.6a)$$

$$\text{subject to } \dot{\mathbf{s}}_i = \mathbf{f}_i(\mathbf{s}_i, \mathbf{u}_i(\mathbf{s}_i); \mathbf{W}_i), \quad (4.6b)$$

$$\tilde{Y}(\mathbf{x}_j) = Z(\mathbf{x}_j) + \epsilon(\mathbf{x}_j) \text{ for all } \mathbf{x}_j \in \mathbf{X}(\mathbf{W}), \quad (4.6c)$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_i(\mathbf{s}_i) \leq \mathbf{u}_{\max}, \quad (4.6d)$$

$$T(\mathbf{w}_{i,N_w}) \leq T_m \text{ for all } i = 1, \dots, N, \quad (4.6e)$$

where  $T(\mathbf{w}_{i,j}) \triangleq \psi(\mathbf{w}_{i,j}; \mathbf{W}_i, \mathbf{s}(t_0), t_0)$  is a function that returns the elapsed time for agent  $i$  to reach its  $j$ th waypoint given its list of waypoints and the initial state of the agent at time  $t_0$ . The evolution of system states for agent  $i$  through time is constrained by the closed-loop dynamics in (4.6b) while the allowable control vector is restricted in (4.6d). Heterogeneity in dynamics is encapsulated through the equations and motion and constraints of inputs. The variation between agents' sensing capability is detailed in (4.6c). The constraint on mission time detailed in (4.6e) ensures that the estimated capture time of the final mission waypoint is less than or equal to the desired total mission time.

#### 4.1.4 Algorithm

Direct minimization of the cost (4.6) for real-time, adaptive sampling is computationally intensive. Therefore, a sub-optimal algorithm is proposed which extends the CVT path planning approach in [7] to consider agents with heterogeneous sensing, heterogeneous dynamics, and mission time constraints. The process by which the cost function and associated constraints detailed in (4.6) are achieved comprises of repeated calls to the modified centroidal Voronoi path generation algorithm, **mCVP**G [A.1], from an algorithm [A.2] for adjustment of **mCVP**G parameters to achieve target mission time,  $T_m$ . This optimization process in [A.2] is online with replanning occurring at fixed intervals throughout the mission.

Algorithm 2 begins with the initialization of  $N \cdot N_w$  unique points within the sample space  $\mathcal{Q}$  [A2.2], where  $N_w$  is the total number of waypoints assigned to each agent. Note that this is typically done using a waypoint pattern to give initial structure to the path; however, any pattern or random initialization strategy should be suitable but untested in this work. This waypoint set is then discretized into  $N_c$  planning cycles comprising  $C = \lceil N_w/N_c \rceil$  waypoints [A2.3] with  $\lceil \cdot \rceil$  denoting the ceiling operator. The entire mission comprises the iteration over each of these cycles as information is progressively collected, with  $g$  being the current cycle index. At the beginning of each planning cycle, the set of measurement coordinates,  $\mathbf{X}$ , and noisy observations,  $\tilde{\mathbf{Y}}$ , is utilized to calculate the cost field at points in  $\mathcal{P}$  [A2.5]. The cost field constructed using point cost (4.5) is arranged as a matrix

$$\mathfrak{J} = \begin{bmatrix} \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{1,1}, \alpha) & \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{1,2}, \alpha) & \dots & \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{1,n}, \alpha) \\ \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{2,1}, \alpha) & \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{2,2}, \alpha) & \dots & \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{2,n}, \alpha) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{m,1}, \alpha) & \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{m,2}, \alpha) & \dots & \mathcal{J}(\mathbf{X}, \tilde{\mathbf{Y}}; \mathbf{p}_{m,n}, \alpha) \end{bmatrix}, \quad (4.7)$$

where  $\mathbf{p}_{i,j}$  is the grid point associated with the  $i$ th row and the  $j$ th column. For the first iteration, when no observations have been performed, the cost at all grid points is initialized to a non-zero constant for convenience of calculating the initial cell centroids. The cost field is then passed along to the **mCVP**G algorithm for the selection of waypoints which seek to minimize the cost function (4.6) without consideration of constraints.

Using the  $N \cdot N_w$  within the sample space as generating points, **mCVP**G begins by calculating the Voronoi tessellation [A1.3]. The cost field,  $\mathfrak{J}$  defines a density function over the Voronoi cells that is used to calculate the mass centroids of each cell using (2.19) [A1.4]. A potential field is then created by modeling each agents' waypoints as an independent spring-mass-damper system to pull waypoints in directions that will minimize the cost. Waypoints are masses that are connected by springs to their corresponding Voronoi cell mass centroid and to adjacent waypoints in the path. Dampers are placed alongside each spring to model energy dissipation as the system comes to a rest at an equilibrium position. The values of the spring constants, dampers, and relaxed spring length corresponding to an agent constitute the set  $\gamma_i = \{k_{p,i}, k_{c,i}, b_i, d_i\}$  for  $i = 1, \dots, N$ . The set of configuration parameters

is initialized at the beginning of each planning cycle with user defined initialization variables in `initSDs` [A2.6].

---

**Algorithm 1** Multi-agent centroidal Voronoi path generation (mCVPg)

---

**Require:**  $\{\mathbf{W}_i\}_{i=1}^N$  // generating points  
**Require:**  $N$  // number of agents  
**Require:**  $\{\gamma_i\}_{i=1}^N$  // spring, damp. constants  
**Require:**  $g$  // current cycle  
**Require:**  $C$  // number of waypoints per cycle  
**Require:**  $\mathfrak{J}_k$  // cost surface  
**Require:**  $\eta$  // stopping criteria  
**Require:**  $I$  // maximum iterations  
1:  $b \leftarrow 0$  // iteration counter  
2: **while**  $b < I$  **do**  
3:    $V \leftarrow \text{voronoiTessellation}(\{\mathbf{W}_i\}_{i=1}^N)$   
4:    $\mathbf{c}_V \leftarrow \text{voronoiMassCentroid}(V, \mathfrak{J}_k)$   
5:   **for**  $\alpha = 1, 2, \dots, N$  **do**  
6:      $\mathbf{f}_\alpha \leftarrow \text{calcForce}(\mathbf{W}_\alpha, \gamma_\alpha, \mathbf{c}_V)$   
7:      $(\mathbf{W}_\alpha, \dot{\mathbf{W}}_\alpha) \leftarrow \text{movePts}(\mathbf{W}_\alpha, \mathbf{f}_\alpha, g, C)$   
8:   **end for**  
9:   **if**  $\|\dot{\mathbf{w}}_{i,j}\| \leq \eta$  **for all**  $i = 1, \dots, N$  **and**  $j = 1, \dots, N_w$  **then**  
10:     **break** // waypoints converged  
11:   **end if**  
12:    $b \leftarrow b + 1$   
13: **end while**  
14: **return**  $\{\mathbf{W}_i\}_{i=1}^N$  // updated generating points

---

Figure 4.2 visualizes this spring-mass-damper system for a series of three waypoints. Each waypoint,  $\mathbf{w}_{i,j}$ , excluding the first and last elements, has three linear spring and damper connections: one to its previous neighbor  $\mathbf{w}_{i,j-1}$ , one to its next neighbor ( $\mathbf{w}_{i,j+1}$ ), and one to its corresponding mass centroid ( $\mathbf{c}_{V_{i,j}}$ ). The spring force exerted upon the  $j$ th waypoint by its adjacent waypoints is:

$$\mathbf{f}_{i,j}^{(p)} = k_{p,i} [(\|\mathbf{r}_{i,j-1}\| - d_i) \hat{\mathbf{r}}_{i,j-1} + (\|\mathbf{r}_{i,j+1}\| - d_i) \hat{\mathbf{r}}_{i,j+1}], \quad (4.8)$$

where  $\mathbf{r}_{i,j\pm 1} = \mathbf{w}_{i,j} - \mathbf{w}_{i,j\pm 1}$  is the vector distance between waypoint  $j$  and the next waypoint,  $(j+1)$ , or the previous waypoint,  $(j-1)$ . The normalized distance vector is  $\hat{\mathbf{r}}_{i,p} = \mathbf{r}_{i,p}/\|\mathbf{r}_{i,p}\|$ ,  $d_i$  is the spring's unstretched length, and  $k_{p,i}$  is a spring constant. The relaxed spring length is updated prior to each planning cycle to be  $d_i = 0.5\bar{v}_i(T_m - T(\mathbf{w}_{i,gC}))/ (N_w - gC)$ , where  $v_{d_i}$  is the  $i$ th agent's target speed. This relaxed spring length heuristic divides the expected distance agent  $i$  will travel given the remaining time,  $T_m - T(\mathbf{w}_{i,gC})$ , and equally spacing the remaining waypoints,  $N_w - gC$ . The force exerted upon the  $j$ th waypoint by its mass centroid is  $\mathbf{f}_{i,j}^{(c)} = k_{c_i}(\mathbf{c}_{V_{i,j}} - \mathbf{w}_{i,j})$  where  $k_{c_i}$  is a nominal constant for the centroidal attracting spring force. Additionally, the linear damping force is  $\mathbf{f}_{i,j}^{(b)} = -b_i\dot{\mathbf{w}}_{i,j}$  where  $\mathbf{w}_{i,j}$  is the velocity of the waypoint. The total force [A1.6] acting upon point  $\mathbf{w}_{i,j}$  is

$$\mathbf{f}_{i,j} = \mathbf{f}_{i,j}^{(p)} + \mathbf{f}_{i,j}^{(c)} + \mathbf{f}_{i,j}^{(b)} \quad (4.9)$$

The equation of motion for waypoint  $\mathbf{w}_{i,j}$  with mass  $m$  is then  $\ddot{\mathbf{w}}_{i,j} = \mathbf{f}_{i,j}/m$  and all waypoints are assumed to have equal mass  $m$ . The function `movePts` performs Euler integration to determine the new position of each waypoint given the associated total force (4.9). The evolution of this online algorithm means that waypoints associated with previous planning cycles have been attained and no longer need updating. That being said, the nature of Voronoi tessellation and its integration within the `mCVP` algorithm demands that the previously attained waypoints remain consid-

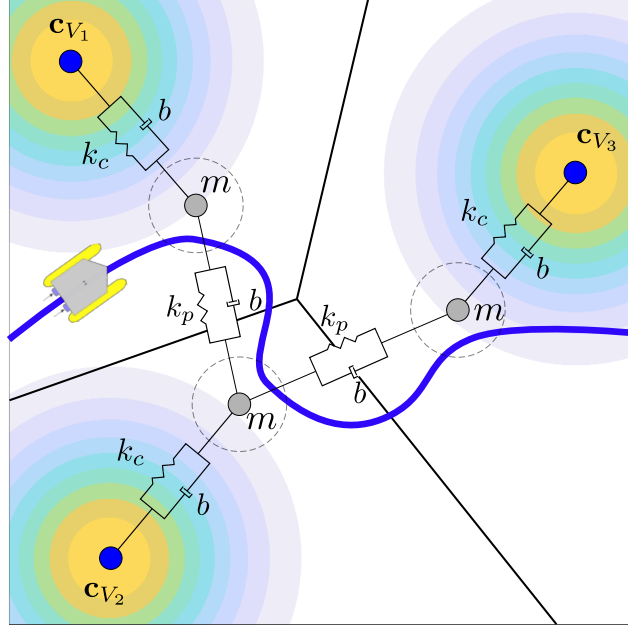


Figure 4.2: A constrained CVT of three waypoints visualized as gray circles with mass  $m$  and associated capture radius. Voronoi cells are defined by (2.17). Mass centroids are depicted as blue circles and are closely co-located with their Voronoi cell's region of high density corresponding to warmer colors in the contour plot. Pairs of adjacent waypoints are connected by a spring,  $k_p$  and damper,  $b$ , system. Waypoints are connected to their associated mass centroid with unique spring constant,  $k_c$ , and damping,  $b$ . A dynamics-compliant trajectory waypoints set is shown in blue.

ered. Therefore, if the waypoint is associated with a future cycle, i.e., the waypoint index is greater than  $gC$ , the position is updated and the velocity is recorded. Otherwise, the waypoint is frozen and the associated velocity is zero [A1.7]. The system is numerically integrated until the speed of each waypoint is below a desired threshold,  $\eta$  [A1.9]. In lieu of explicit guarantees of system convergence in given time, a maximum number of iterations is imposed [A1.2].

Upon return from the mCVPG algorithm, each agent is assigned a set of  $N_w$  waypoints, having already visited  $gC$  of them. Each agent is simulated through its

remaining waypoints from its current state [A2.11] and the estimated time to traverse all remaining waypoints is reported. This simulation time,  $t_i$ , is compared with the remaining mission time [A2.13]. If the difference exceeds the user defined mission time tolerance,  $T_\Delta$ , the spring-mass-damper system parameters are updated. The stiffness of springs connecting waypoints to the Voronoi cell mass centroids when the simulated time is less than the desired time and decreasing their stiffness when the simulated time is greater than the desired time. In practice, a buffer,  $\mathbf{E}$ , contains the mission time error from previous iterations for each agent [A2.12]. Let  $\max(\mathbf{E}_i)$  be the maximum mission time error for the  $i$ th agent. The spring stiffness is updated for the next optimization iteration as follows:

$$k_{c,i} \leftarrow \begin{cases} k_{c,i} \left( \frac{\epsilon_i}{\max(\mathbf{E}_i)} + 1 \right) & \text{if } t_i + T_\Delta < T_m \\ k_{c,i} \left( \frac{\epsilon_i}{\max(\mathbf{E}_i)} + 1 \right)^{-1} & \text{if } t_i - T_\Delta > T_m \\ k_{c,i} & \text{if } |t_i - T_m| \leq T_\Delta. \end{cases} \quad (4.10)$$

The chosen method of parameter update is a modified version of optimization shown to result in convergence of this particular type of spring-mass-damper system in [7]. The damping coefficients for the  $i$ th agent are  $b_i = 0.5\sqrt{\max(k_{p,i}, k_{c,i})}$  [A2.14], similar to Jensen-Nau et. al. [7]. The optimization loop [A2.8 – 21] runs until the simulated mission time for agent  $i = 1, 2, \dots, N$ , is less than  $T_\Delta$  away from the desired mission time, calculating a new constrained CVT solution with updated parameters during each optimization step. A maximum number of iterations is enforced [A2.8] to the optimization loop for each cycle to ensure a result is returned in finite time.



The previously described portion of the optimization algorithm handles the simulation of agents' motion for the selection of waypoints to satisfy mission constraints and optimize for information observation of the unknown field. Once the agents are deployed, the proposed waypoints are allocated to each agent and the process of measurement collection begins [A2.22]. Agents traverse the waypoint defined paths and append measurement information into the  $\mathbf{X}$  and  $\tilde{\mathbf{Y}}$  matrices. Due to discrepancies between simulated and actual mission time during experiments, the time elapsed since the previous planning cycle is reported, and the remaining mission time is updated [A2.23]. The mission control algorithm continues until all planning cycles are complete.

---

**Algorithm 2** Satisfying mission time constraint using mCVPG with multiple agents

---

**Require:**  $T_m \geq 0$  // mission time  
**Require:**  $T_\Delta \geq 0$  // mission time tolerance  
**Require:**  $N$  // number of agents  
**Require:**  $N_w$  // total num. waypoints in mission  
**Require:**  $N_c$  // number of planning cycles  
**Require:**  $\gamma_0$  // init. spring, damp. constants  
**Require:**  $\mathcal{Q} \subset \mathbb{R}^2$  // operating domain  
**Require:**  $\mathcal{P}$  // evaluation grid points set  
**Require:**  $I$  // maximum iterations  
**Require:**  $I_s$  // maximum iterations for mCVPG  
**Require:**  $\{\bar{v}_i\}_{i=1}^N$  // agents' target speed  
1:  $\mathbf{X} \leftarrow, \mathbf{Y} \leftarrow$  // initialize measurement information to null  
2:  $\mathbf{W} \leftarrow \text{initializeWaypoints}(N, N_w, \mathcal{Q})$   
3:  $C \leftarrow \lceil N_w / N_c \rceil$  // num. wpts per cycle  
4: **for**  $g = \{1, 2, \dots, N_c\}$  **do**  
5:    $\mathfrak{J} \leftarrow \text{evaluateCost}(\mathbf{X}, \mathbf{Y}, \mathcal{P})$  // calculate cost field  
6:    $\{\gamma_i\}_{i=1}^N \leftarrow \text{initSDs}(\gamma_0)$   
7:    $b \leftarrow 0$  // iteration counter  
8:   **while**  $b < I$  **do**  
9:      $\mathbf{W} \leftarrow \text{mCVPG}(\mathbf{W}; N, \{\gamma_i\}_{i=1}^N, g, C, \mathfrak{J}, 10^{-4}, I_s, \{d_i\}_{i=1}^N)$   
10:     **for**  $\alpha = \{1, 2, \dots, N\}$  **do**  
11:        $t_\alpha \leftarrow \text{simulateAgent}(\mathbf{W}_\alpha; \mathbf{s}_\alpha(T(\mathbf{w}_{\alpha, gC})), g, C, \bar{v}_\alpha)$   
12:        $\epsilon_\alpha \leftarrow T_m - t_\alpha$  // sim. time error  
13:       **if**  $|\epsilon_\alpha| > T_\Delta$  **then**  
14:          $\gamma_\alpha \leftarrow \text{updateSDs}(\epsilon_\alpha, T_m, \gamma_\alpha)$   
15:       **end if**  
16:     **end for**  
17:     **if**  $|\epsilon_i| \leq T_\Delta$  for all  $i = 1, \dots, N$  **then**  
18:       **break**  
19:     **end if**  
20:      $b \leftarrow b + 1$   
21:   **end while**  
22:    $(\mathbf{X}, \mathbf{Y}, t_m) \leftarrow \text{collectData}(\mathbf{W}, N, g, C, \mathbf{X}, \mathbf{Y})$  // actual mission  
23:    $T_m \leftarrow T_m - t_m$  // remaining mission time  
24: **end for**

---

## 4.2 Simulation Results

In this section, the proposed approach is applied to numerically simulated GPs generated via Cholesky's decomposition of the covariance matrix (2.6). Each GP realization is constructed using  $\boldsymbol{\theta} = [\zeta, \omega, \sigma_0^2]^\top = [0, \omega_t, 1]^\top$  as the hyper-parameters where  $\omega_t$  is a selected entry from Table 4.1. The *HC* parameters are  $w_{\mathcal{D}} = h_{\mathcal{D}} = 0.5\omega$ ,  $w_{\mathcal{G}} = h_{\mathcal{G}} = 1.5\omega$  and  $M_{\min} = 10$ . In each simulated experiment, two pairs of agents observe the same GP; one pair attempts to adaptively minimize the proposed cost function (4.6) while the other pair travels along a predefined lawnmower path. The effectiveness of the proposed algorithm is determined by comparing the total mapping error to that of the naive, lawnmower survey given identical initial conditions.

### 4.2.1 Monte Carlo Simulation Setup

Each simulation scenario is constructed based upon one of the control agents. Designated the “leader”, this agent is assigned a target speed and number of swaths around which to construct a lawnmower path to survey the field. The total mission time is dependent on the leader's target speed and number of swaths. The other agent, or “follower,” is assigned a consistent three swaths within the field while its speed is dependent on the leader. Figure 4.3 depicts a leader with four assigned swaths, spreading the survey region throughout  $\mathcal{Q}$ . The waypoints are defined in terms of swath width,  $c_w$ . The swath width,  $C_w$ , is the width of the space,  $\mathcal{Q}_w$ , divided by the total number of swaths,  $N_l$ :  $c_w = \mathcal{Q}_w / (N_l + 1)$ . The points are inset from the border of the space by  $c_w/2$  and the order of the waypoints is such that the left-most agent travels to the right and the right-most agent travels to the

left, both initially moving upward. The leader is simulated through its assigned

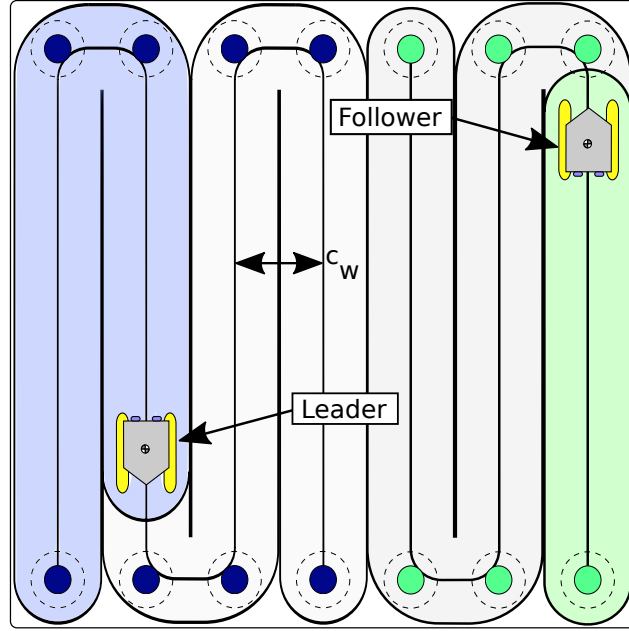


Figure 4.3: Example simulation scenario featuring a “leader” and “follower” agent. The leader is assigned four swaths, the follower is assigned three. Paths are positioned within the field so that the sensor coverage area is evenly distributed. The colored shaded regions depict the portion of the field surveyed by each agent, while the gray is to be surveyed in the future. The follower agent trails the leader as its target speed is lesser to ensure common completion times.

waypoints, the follower agent’s target speed is then adjusted to ensure that each agent achieves all assigned waypoints in common time. These parameters are then transferred directly to the adaptive agents to ensure that each simulation contains identical initial conditions and agent dynamics. Covering all of the desired variations depicted in Table 4.1 with 30 iterations with varying GP simulations resulted in 3600 simulations.

The simulations conducted with the leader/follower arrangement in a lawnmower pattern constitute a “control” group that is then compared with the proposed adap-

Table 4.1: Parameters used in Monte Carlo simulation. Simulations were conducted on all permutations of the listed variables used a square scalar field with normalized lengths. Spatial variables are therefore unit-less and speed variables represent the percentage of the major dimension the agent can travel per second. Gaussian process length-scales are also interpreted as percentages of the field dimension.

Parameter	Symbol	Value
Number of agents	$N$	2
Agent speed	$v$	$(0.01, 0.02, 0.04) s^{-1}$
Measurement noise	$\sigma_\eta^2$	$\{0.0, 0.05, 0.15\} m^2$
GP length-scale	$\omega$	$\{0.1, 0.2, 0.3, 0.5\}$
Number of swaths for leader agent	$N_l$	$\{6, 7, 8, 9, 10\}$
Number of planning horizons	$N_c$	10
Agent mass	$m$	1 kg
Agent moment of inertia	$I$	$0.1 \text{ kg}\cdot\text{m}^2$
Maximum thrust	$u_{\max}$	$10 \text{ kg}\cdot\text{f}$





tive sampling algorithm. The simulations with the adaptive sampling approach use identical agent dynamics and initial configurations to the control group. The adaptive group is initialized with  $N_w$  along the same lawnmower pattern given to the control ground agents.

#### 4.2.2 Monte Carlo Simulation Results

Results from the simulations are shown in Figure 4.4. Subfigure 4.4a compares agents matching speed over varying GPs with noisy and noise-free measurements. In the case of noise-free measurements, the proposed adaptive algorithm exceeds the performance of the lawnmower pattern in fields where the length-scale is small relative to the field. As the length-scale approaches 50% of the field, adaptive algorithm and lawnmower patterns converge to similar mapping errors given that there is very little variation to observe within the field. When comparing the noisy measurements, the same trend exists, but is much less pronounced. The smoothing effect of mea-

surement error filtered kriging adds difficulty to the estimation process which makes proper path selection less reliable. Subfigure 4.4b compares the effect of heterogeneous dynamics on the performance of the adaptive sampling algorithm over a GP with a length-scale of 0.2. The abscissa shows the total number of swaths across both agents. Given that the follower agent always travels three swaths, the leader therefore varies from three to seven swaths. Additionally, the leader varies its speed between  $0.01\text{s}^{-1}$ ,  $0.02\text{s}^{-1}$ , and  $0.04\text{s}^{-1}$  while the follower alters its speed to ensure common mission times. This plot shows the insensitivity of the algorithm to variations in speed given the close trend of the adaptive results with a similarly close trend among the control as a sanity check. Additionally, the convergence of the adaptive and control curves show the diminishing returns from the adaptive algorithm as mission time increases.

Table 4.2: Comparison of simulation results given all permutations of dynamics and sensing quality. The result being the fraction of final field cost of the adaptive sampling mission to the cost of the control sampling mission. The red line being parity, a ratio of 1:1. Simulations performed over a scalar field with  $\sigma_0^2 = 1$  and  $\omega = 0.1$ . Ninety simulations split evenly among the discrete mission classifications.

Cases	Dynamics		Sensing		Results
	Equal	Hetero.	Equal	Hetero.	
1	X		X		 1.16
2		X	X		 1.41
3	X			X	 1.05
4		X		X	 1.08

### 4.2.3 Illustrative Example

An example of resulting trajectories in multi-agent sampling is depicted in Figure 4.5. Subfigures 4.5a and 4.5b are the planned trajectories and cost field after

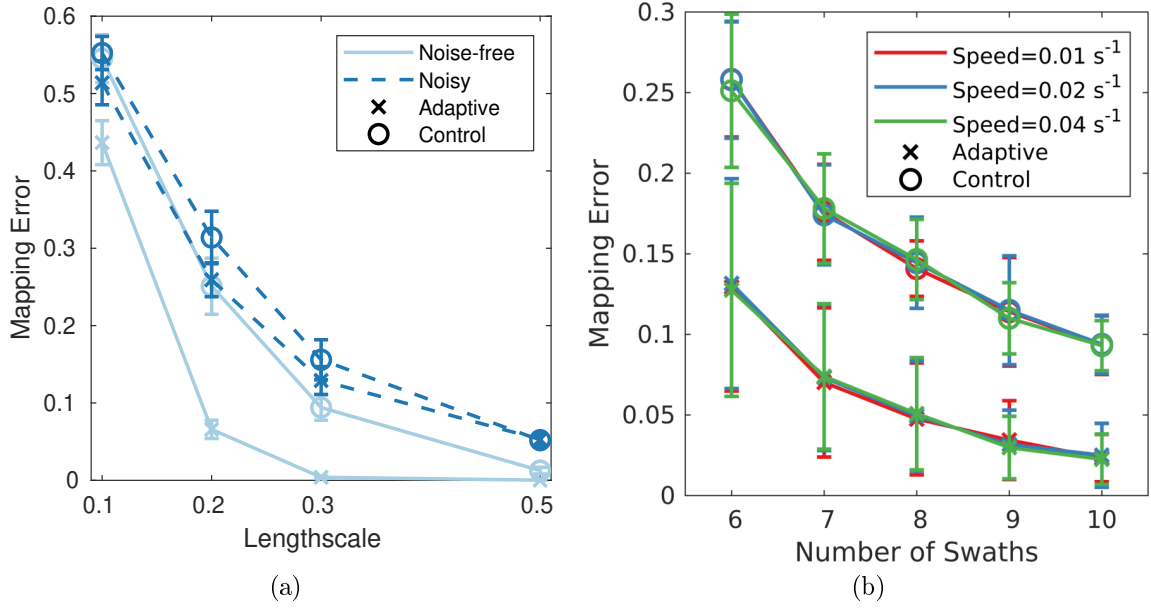


Figure 4.4: Monte Carlo simulation results given parameters specified in Table 4.1. Panel (a) compares final mean mapping error over all simulated length-scales when both agents’ speeds are  $0.01\text{s}^{-1}$  and given no measurement noise variance (“Noise-free measurements”) and  $\sigma_{1,\eta}^2 = 0.15, \sigma_{2,\eta}^2 = 0.05$  for measurement error case (“Noisy measurements”). Panel (b) compares the effects of agent speed on ME in a field with a length-scale of 0.2. The follower agent consistently travels three swaths while the leader varies from three to seven swaths while traveling at varying speeds.

the second planning cycle and the final planning cycle respectively when each agent makes noise-free measurements, henceforth referred to as “mission 1”. Subfigures 4.5d and 4.5c are the planned trajectories and cost field at the same time-steps as the aforementioned, henceforth “mission 2”; however, the agent associated with the red line has a measurement error variance  $\sigma_{1,\eta}^2 = 0.15$  while the white line agent has an error variance of  $\sigma_{2,\eta}^2 = 0.05$ . The scalar field over which both scenarios are operating is shown in Figure 4.5f and the lawnmower sampling pattern used as a control for comparison is shown in Figure 4.5c. Mission 1 clearly shows how agents with common

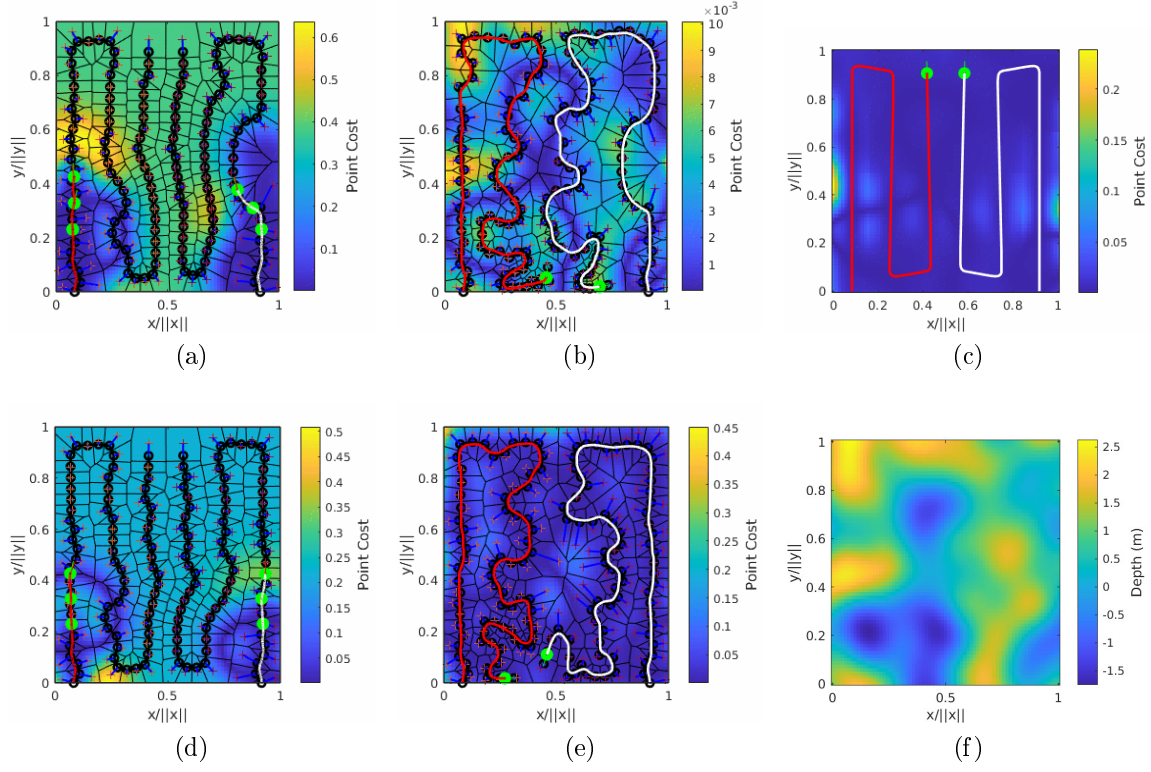


Figure 4.5: Examples of adaptive patterns given  $v = 0.01s^{-1}$ ,  $N_l = 3$ , and  $\omega = 0.2$ . From left to right, top to bottom: panel (a) shows the cost field at the end of the second planning horizon given no measurement noise for either agent. Panel (b) shows the final paths taken by the same agents from (a) and the resulting cost field while (c) is the path and cost field of the control agents. Panel (d) is the same snap shot as (a) given measurement noise  $\sigma_{1,\eta}^2 = 0.15$  for the red agent and  $\sigma_{2,\eta}^2 = 0.05$  for the white agent. (e) is the final path and cost field related to (d). (f) is the actual field over which all of these mission occurred.

measurement noise variance will cover roughly common field area while attempting to follow stretches of peaks and valleys as they occur within the field. Mission 2 shows how the white agent, having better sensing capabilities, seeks to support the red agent which suffers from excessive measurement noise variance.



## CHAPTER 5: DESIGN OF AUTONOMOUS SURFACE VESSELS

The following chapter details the design process associated with the construction of a pair of Autonomous Surface Vessels (ASV) used for testing and experimental validation. The selection of components, software design, and validation of performance criteria in order to evaluate system capability for required testing, is described.

### 5.1 Background

There exist numerous commercially available autonomous surface vessels for the purposes of environmental monitoring. These products are robust robots capable of carrying modular sensor packages that allow users to perform short, automated missions for the collection of data. Figure 5.1a shows several examples of such ASVs: OceanX’s ME120 [43], 5.1b is Evologics’ Sonobot [44], and 5.1c is Maritime Robotics’ Otter [45]. All these robots are outfitted natively with sonar capabilities for bathymetric mapping as well as the ability to extend their sensing suite as needed.

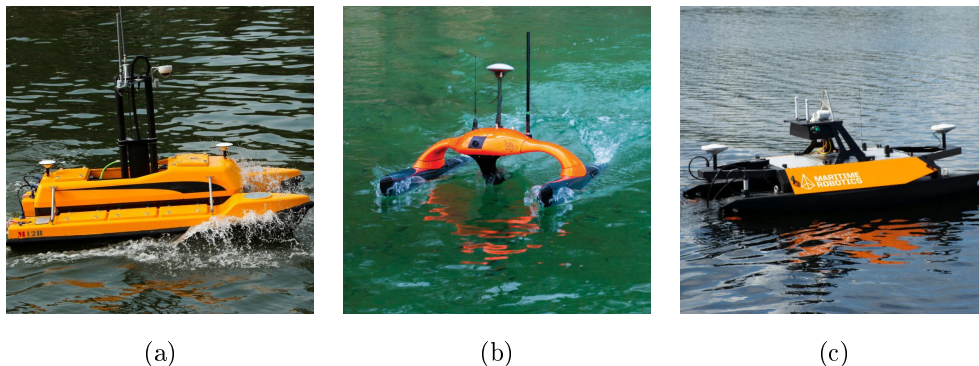


Figure 5.1: Examples of commercially available Autonomous Surface Vessels for environmental monitoring.

The platforms shown in 5.1 come with proprietary software packages to configure their mission parameters. This closed ecosystem, along with their prohibitive cost, make commercially available ASVs unattractive to research labs whose needs differ from the needs of government or industry. The Autonomous Robotics and Systems Laboratory at UNCC decided to develop a platform which was low-cost and open source. These features allow for a fleet of ASVs to be developed for the cost of a single commercial product while providing a simple framework for autonomy and controls research.

The goals for the system are as follows: (1) a per vehicle cost of  $< \$5000$  with a minimum of two vehicles, (2) an open-source control framework written mostly in Matlab, (3) low weight for ease of deployment, and (4) a modular sensor suite. The following sections will describe how each of these goals were satisfied during the design of the UNCC ASV. Section 5.2.3 describes the design process and component selection for the chassis of the platform. Section 5.2.2 details the layout of electri-

cal components and describes expected performance. Section 5.2.1 highlights the selection of sensors necessary for autonomous operation as well as the modularity of environmental sensor packages available to the system. Section 5.2.4 overviews the open source control framework developed using Matlab. The concluding section presents experimental results and derived performance metrics.

## 5.2 ASV Design Overview

The design process began with a review of the existing systems on the market to determine reasonable performance goals and understand design principles which are employed in professional systems. Key characteristics of the systems shown in Figure 5.1 are broken out in Table 5.1. A review of the performance capabilities of these systems led to the definition of metrics deemed to be reasonable for a low-cost system.

Table 5.1: A comparison between commercially available ASVs and the planned UNCC ASV.

	Length (m)	Width (m)	Max Speed (m/s)	Battery Life (h)	Comm. Range (km)	Weight (kg)
ME120	2.5	1.4	5	6	5	150
Otter	2	1.08	1.54	9	-	65
Sonobot 5	1.3	0.92	5	9	1.5	27
UNCC	0.9	0.7	1–2	1–2	1	<18

Additionally, significant computational power was desired for the UNCC ASV to allow for onboard adaptive path planning. For this reason, significant diminution of

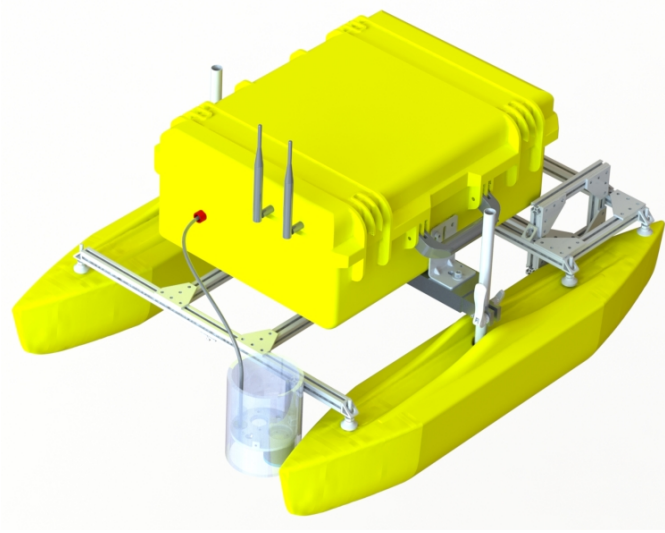


Figure 5.2: Rendering of the UNCC ASV platform.

mission time and top speed were deemed allowable to compensate for the priority of compute power. Figure 5.2 is a rendering of the proposed design for the platform.

### 5.2.1 Sensor Suite

The sensor suite consists of devices for measuring the configuration variables of the autonomous system, such as heading, relative location, speed, etc., as well as sensors for environmental monitoring. Sensors were selected prior to design and construction of the platform to ensure that the system was designed to optimize the available sensors and to avoid a more difficult sensor integration with an existing system.

#### 5.2.1.1 Attitude and Heading Reference System (AHRS)

For effective autonomous control, the robot should understand its orientation with respect to a global reference frame. For this, an Attitude and Heading Reference Sys-

tem (AHRS) combines multiple sensor inputs to construct an estimate of the sensors three-dimensional orientation. AHRSs are typically constructed using a gyroscope for the sensing of angular rotation rate, an accelerometer for the sensing of linear acceleration, and a magnetometer for sensing of local magnetic fields to act as a compass to align with the Earth's magnetic field. A typical AHRS will employ a sensor fusion algorithm onboard that is designed to utilize all the sensors to generate an estimate which is more accurate than a measurement derived from a single sensor alone.

For the ASVs in this project, the LORD Microstrain 3DM-GX5 series of AHRS was selected. This line of sensors provides advanced sensor fusion algorithms which utilize extended Kalman filtering to provide an accurate and stable estimate of orientation. The system has been designed to operate with two different sensors from the 3DM-GX5 series: the 3DM-GX5-25 and the 3DM-GX5-45. The -45 includes a Global Navigation Satellite System (GNSS) receiver while the -25 does not. Section 5.2.1.2 discusses the configuration implications of the different sensors.

Communication between the ASV main computer and the AHRS unit uses the LORD Data Communications Protocol, a proprietary packetized interface which allows for a command and response format. LORD provides an Application Programming Interface (API) for easy integration of the AHRS targeting C++, Python and .NET [46]. Since the supplied API could not be utilized directly with Matlab on Linux systems a custom implementation of the LORD Data Communications Protocol was developed in C++ to convert the communications from a packetized hexadecimal format known as MicroStrain Inertial Packet, or MIP, to a NMEA-like string

Table 5.2: Communication packet information for 3DM-GX5 systems.

Descriptor	MIP ID	Provided Information
\$EULER_EF	(0x82,0x05)	Roll, Pitch, Yaw
\$EULER_RATE	(0x82,0x0e)	Roll Rate, Pitch Rate, Yaw Rate
\$EF_STATUS	(0x82,0x10)	Filter State

format for easy debugging. Table 5.2 shows the most utilized commands for reading orientation, angular rate, and the status of the EKF which provides information on the expected reliability of the output of the filter.

#### 5.2.1.2 Global Positioning System (GPS)

The ASV is nominally configured with the 3DM-GX-25 AHRS and utilizes a BU-353S4 GPS receiver for positional information. This receiver provides positional information at a rate of 1 Hz in degrees latitude and longitude. The BU-353S4 utilizes the NMEA-0183 communication protocol for broadcasting location information over USB [47]. Table 5.3 shows a subset of NMEA codes sent by the GPS receiver which are used for by the ASV system for location information.

Packet Type	UTC <sup>[1]</sup> Time	Latitude	Longitude	Status	Satellite Info	SOG <sup>[2]</sup>
\$GPGGA	X	X	X	X		
\$GPGSA				X	X	
\$GPRMC	X	X	X		X	X

<sup>1</sup> Universal Time Coordinated

<sup>2</sup> Speed over Ground

Table 5.3: Communication packet information for BU-353S4 using NMEA-0183 string format. The \$GPGGA and \$GPGSA packets supply information on the satellite status

which can be used to determine if a sufficient number of satellites are communicating with the GPS receiver to achieve accurate positional information. The \$GPRMC consists of the minimum recommended GPS data provided by the receiver. This packet type provides information on location and estimated speed. Figure 5.3 shows the format of a \$GPRMC NMEA-0183 packet. The simple comma-delimited format of the protocol makes it human-readable and simple for debugging purposes.

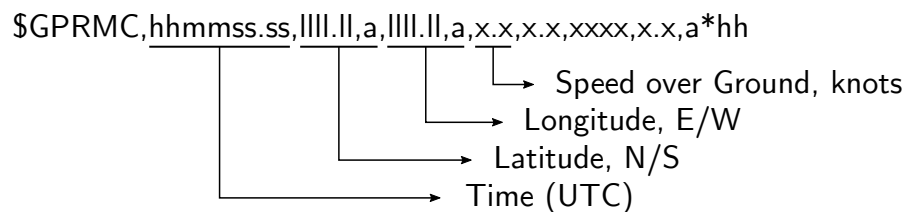


Figure 5.3: Communication packet layout for recommended minimum specific GNSS data.

Alternatively, the ASV can be configured to use the 3DM-GX5-45 inertial navigation system (INS). This INS unit integrates GPS and therefore the BU-353-S4 becomes redundant. The 3DM-GX5-45 can provide positional information at a rate of 4 Hz. As mentioned previously, the communication method used by the 3DM-GX5-45 is more opaque than the NMEA string format. The software designed to parse the sensor communications transcribes the output into a NMEA-like string for GNSS information much like the format used for orientation information. Table 5.4 shows the packet information returned by the 3DM-GX5-45 for parsing by the main program in Matlab.

Table 5.4: Communication packet information for GNSS output of 3DM-GX5-45.

Descriptor	MIP ID	Provided Information
\$GPS	(0x81,0x03)	Latitude, Longitude, Horizontal Accuracy, Vertical Accuracy
\$GNSS_STATUS	(0x82,0x0e)	Fix Type, Number of Satellites
\$NED	(0x82,0x05)	Velocity (North, East, Down)

Both GPS receivers provide location information in degrees latitude and longitude. However, due to the curvature of the Earth, the distance between two GPS coordinates requires consideration of absolute location, making these calculations unwieldy if performed numerous times. To simplify the calculation of distance between coordinates, latitude and longitude are converted to the Universal Transverse Mercator (UTM) coordinate system. The UTM system divides the Earth into 60 north-south zones. Within each zone, coordinates are defined by meters north and east from the equator [48]. This makes distances calculations as simple as finding the Euclidean distance between two UTM coordinate pairs. The system assumes that the ASV will be operating solely in UTM zone 17N.

#### 5.2.1.3 Depth Sensor

The main sensor package for the ASV is the BlueRobotics Ping Sonar. This single-beam echosounder emits a 30 degree beam capable of measuring distances up to 164 feet underwater. The Ping Sonar uses a piezoelectric transducer to produce a 115 kHz frequency pulse which travels from the device, to the floor of the water body and



returns; this action is depicted in Figure 5.4. As the signal returns to the transducer, the diaphragm is disturbed, producing an electrical signal. Computing onboard the Ping Sonar deduces the distance which the wave traveled given the time from send to receive and the speed of sound in the medium. The manufacturer recommended speed of sound for freshwater is 1500 m/s, which is used for this project.

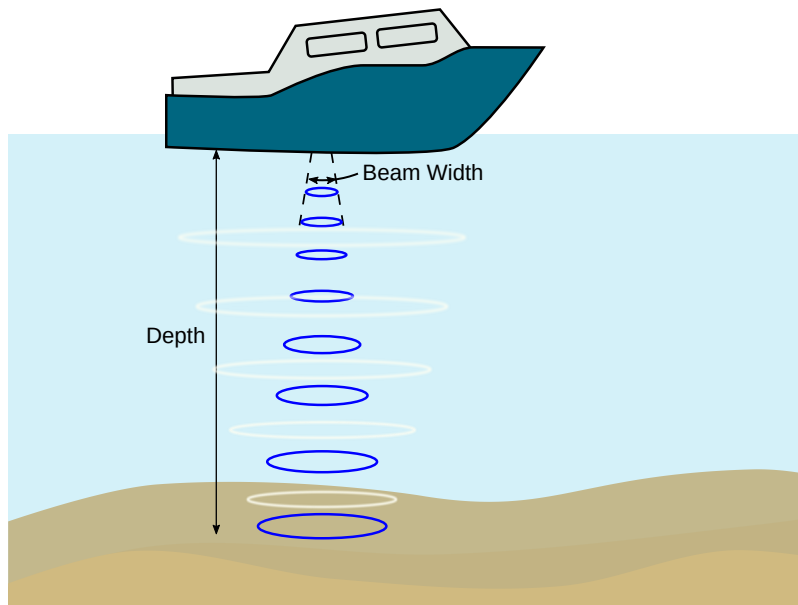


Figure 5.4: Single beam echosounder operation.

BlueRobotics provides APIs for various languages to integrate with the Ping Sonar. However, no packages were provided for Matlab integration, so a custom driver was developed for use with this project.

### 5.2.2 Electrical

Housed within a water-tight Pelican case are all the sensitive electrical components required for operation of the ASV. The electrical system consists of three sub-systems:

(1) batteries and power, (2) computation and sensors, and (3) actuator controllers. The system is powered by two 14.8 V 6000 mAh 50C LiPo batteries. The first battery is converted to a stable 12 V through a DC-DC converter for use with the computation and sensors sub-systems. The second LiPo battery directly powers the motor controllers to avoid current overdraw causing brown-out conditions in the vehicle's computer.

Figure 5.5 is a high-level connectivity diagram between discrete components of electrical sub-systems (2) and (3). The power sub-system was excluded for simplicity; a more detailed version can be found in Appendix B.3. The autonomy computer is an Intel NUC10i7FNH PC with 10th generation i7 processor and 32 GB of DDR4 RAM. This computer acts as the central hub for all sensors and directly commands a separate control computer responsible for lower level hardware interfacing. Wireless communication between agents and the ground station occurs via onboard Wi-Fi as well as an RFD900x radio modem. The RFD900x is a 900 MHz transceiver capable of broadcasting with 1 W of power and receiving transmissions from multiple kilometers away. An additional receiver is included for manual wireless control. The Spektrum AR8010T receiver communicates with a Spektrum DX8e remote control in a one-to-one network operating at 2.4 GHz for a more modest range.

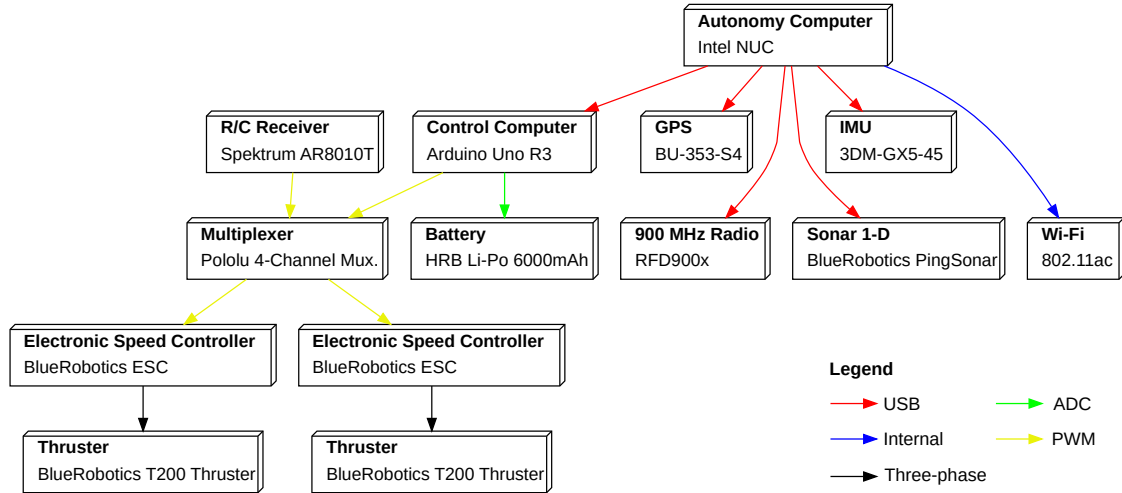


Figure 5.5: High-level connectivity diagram for the ASV. Arrows indicate ownership through direction and method of communication using color.

The control computer is an Arduino Uno R3 microcontroller running at 16 MHz clock frequency. The Arduino is responsible for executing the instructions received from the autonomy computer and interfacing with the vehicle's thrusters and radio receiver. The vehicle is equipped with two BlueRobotics T200 thrusters. The control computer interfaces with a pair of BlueRobotics basic electronic speed controllers (ESCs) that connect to each of the three coil packs in the three-phase brushless motor driving each thruster. By varying the duty cycle of the PWM signal, each thruster can be independently controlled from full reverse thrust to full forward thruster and any thrust in between. BlueRobotics provides experimental data regarding expected thrust given battery voltage and PWM duty cycle [49]. To allow for manual control interrupt, a Pololu 4-Channel RC server multiplexer is placed between the Arduino and the ESCs. The Spektrum AR8010T wireless receiver allows for remote manual control up to 1 km in distance. A switch on the Spektrum DX8e transmitter selects

the signal passed through the multiplexer to the ESCs for remote or autonomous control. In the event of disconnection from the transmitter, the AR8010T is programmed to command zero thruster while it awaits re-connection.

Battery voltage sensing is also handled through the control computer. Information about the battery voltage levels can be requested by the autonomy computer for further processing. Given that the LiPo battery voltage ranges from approximately 13.5 V to 16.5 V, the Arduino is not capable of directly measuring the battery using its internal analog-to-digital converter. An intermediary circuit is used to divide the voltage so that it is in a readable range for the Arduino. The circuit shown in Fig. 5.6 consists of an op-amp configured as a voltage follower with a voltage divider with gain of  $1/3$ . A diode ensures protection to the Arduino and drops the voltage by approximately 1.5 V. An exact linear equation for the operation of the circuit was determined through bench-top testing.

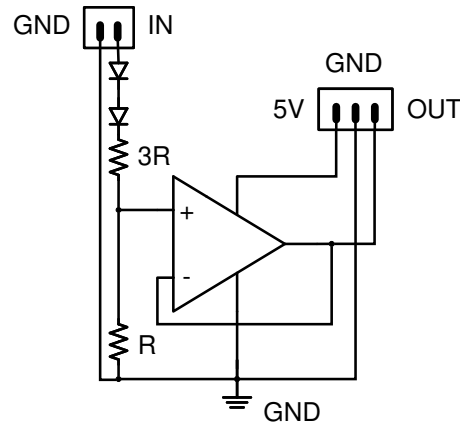


Figure 5.6: Voltage divider for battery sensing.

The electrical components are secured to a rectangular 80/20 extruded aluminum frame with the Pelican case. The communication equipment is secured to the lid of the case using adhesives. Two external 900 MHz half-wave dipole whip antennae connect to the RFD900x to allow for improved connectivity for the radio. Figure 5.7 shows the internal wiring of the Pelican case including locations for LiPo batteries.

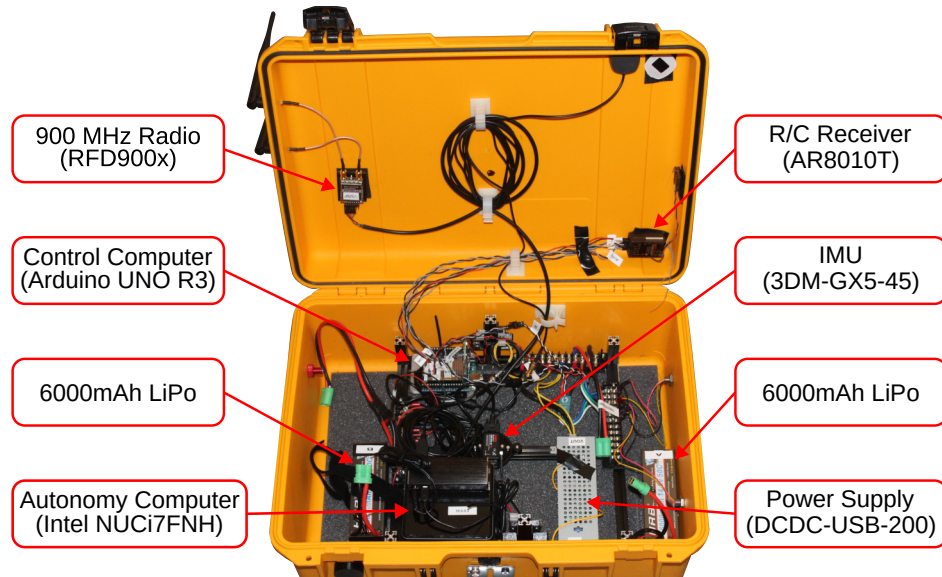


Figure 5.7: Complete Pelican case with labeled components.

### 5.2.3 Hardware, Final Build and Total System Cost

To reduce costs, the chassis of the platform was constructed using off-the-shelf components with minimal modification required. This approach allows multiple ASVs to be constructed quickly and conveniently as needed. The system was also designed to be modular, allowing a user to disassemble the entire system into multiple sub-assemblies for ease of transportation and storage. The system also provides easy

connection points for additional sensors.



Figure 5.8: Deconstructed ASV for transport and storage.

The foundation of the platform is the dual pontoon catamaran design. The set of pontoons are canoe stabilizers purchased from Spring Creek Manufacturing and provide approximately 55 lbs. of buoyancy when fully submerged. The canoe stabilizer kit came with an adjustable crossbeam which was cut to length to achieve the desired width. Cam locks are used to secure each pontoon in position and orientation. However, this system did not provide sufficient restraint of the pontoons, allowing them to be rotated if sufficient force was applied. To avoid this, holes were drilled through the crossbeam and the pontoon post through which a quick release pin was placed to ensure minimal rotation of the pontoon during operation.

Atop the crossbeam is seated the Pelican case housing all the sensitive electronics. On either side of the case, 90-degree brackets are secured using 1/4 in bolts and insulated rivet nuts to ensure the housing remains watertight. These brackets are then secured to the pontoon crossbeam using 5/16 in nuts and bolts which can be



Figure 5.9: ASV propulsion system and framing.

easily removed to separate the Pelican case from the chassis of the ASV. A frame constructed of 20 mm anodized aluminum 80/20 rails rests on top of the pontoons and is secured directly to the pontoon crossbeam. This frame allows for simple connection of various sensors as well as mounting of the propulsion system.

The propulsion system for the ASVs comprises two BlueRobotics T200 thrusters shown in Figure 5.9. These thrusters are mounted on a separate 80/20 frame which is connected to the main chassis framing. This allows the thrusters to be raised and lowered as necessary to ensure that the thruster is submerged during operation. Cobalt series bulkhead connectors from Blue Trail Engineering penetrate the rear of the Pelican case to provide quick connect and disconnect for each thruster.

Figure 5.8 shows the ASV deconstructed into its basic components. The Pelican case is readily removed and can be carried by the handle with all components firmly affixed within. The pontoons are separable from the frame by removing the quick release pin from each and loosening the cam locks. The propulsion system is secured

to the main frame by four 6 mm screws while each thruster can be removed from the propulsion system framing by loosening two Phillips head screws. The frame can be removed from the crossbeam by extracting two screws and folded in half for storage. The final build of the ASV platform is shown in Figure 5.10 and a high-level breakdown of unit cost is found in Table 5.5.



Figure 5.10: Final ASV build excluding sensor package.

This cost breakdown assumes the 3DM-GX5-25 and BU-353S4 sensors are utilized. Given the ability to replace both sensors with the 3DM-GX5-45, the electronics and sensors category becomes \$4374.65 with a total unit cost of \$6157.07.

#### 5.2.4 Software

The ASVs are capable of autonomous waypoint following based upon missions created in a Matlab interface. The focus of the software design was simplicity, ease of debugging, and extendibility. The core tenets of the software design as follows:



Table 5.5: High-level Bill of Materials for ASV platform.

Component	Cost
Chassis	\$873.11
Propulsion	\$504.00
Batteries and Power Supply	\$405.31
Electronics and Sensors*	\$2974.65
	\$4757.07

\* Configuration with BU-353S4 and 3DM-GX5-25

- Open-source software architecture that can be easily modified.
- Simple and detailed enough for future students to be able to quickly understand the core functionality and conduct autonomy and controls research experiments with minimal changes to the core code.
- Minimal dependencies outside of Matlab virtual environment, allowing portability between platforms. Portability should depend only upon the sensor package available to a particular device.
- Sufficiently transparent operations for on-site debugging of core processes and communication.

The ASV's internal computer runs Ubuntu 20.04 with Matlab 2021b installed. To provide the Matlab program easy access to all USB devices, a system was configured to allow hot-plugging (i.e., the ability to add or remove a device from a program without interruption). This prevents any interruptions in the USB device communication from disrupting the operation of the entire program. This is important when working with Matlab programs as they only expose a single thread to the user, mean-

ing all errors are in line with the main programming. By utilizing the package *socat*, a series of pseudo-terminals allow a valid serial stream to be exposed to Matlab at all times, independent of availability of the USB device. These pseudo-terminals are bidirectional communication channels officiated by the kernel which allow processes to speak directly [50, ch. 17.9]. With simple error handling, the programs can detect whether the device is functioning and decide whether or not to interact. In this way, fatal disconnection errors are avoided. A detailed visualization of how the *socat* pseudo-terminals connect the device to the Matlab program is shown in Figure B.1.

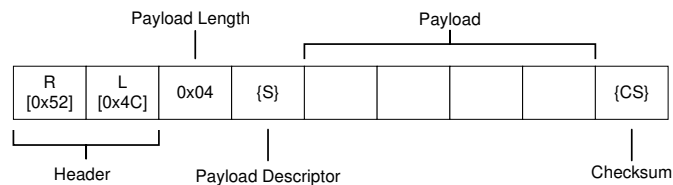
Within the Matlab virtual environment, a handler object is created which reacts to incoming information in an asynchronous manner. When data arrives through the pseudo terminal, an interrupt is triggered, and the message is logged into a First-In-First-Out (FIFO) data structure which the real time program can access to see if any messages from that device are available. Each handler object is responsible for the initialization, parsing of incoming messages, structuring of outgoing messages and clean-up for each external device which is expected to be connected and utilized. In this way, the addition (or exemption) of sensors can be made with minimal alteration of the main control loop. Also, the containerized objects can be allowed to fail gracefully with the strategic placement of *try...catch* statements to let Matlab handle the clean-up of a malfunctioning module.

When Matlab processes information, it stores all variables in memory. Specific calls must be made to ensure that the data stored in memory is packaged and saved onto the hard disk. Storing as much information as possible is critical for research systems and care should be taken to avoid errors which cause logging issues. There-

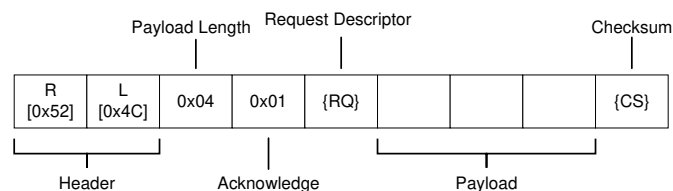
fore, an *SQLite* database was configured to quickly receive information to be stored throughout the operation of the program. Databases are robust and efficient ways to hold large stores of information on the hard disk. With rapid read/write rates, the database is also used to recall information from the current run, rather than storing large, potentially expanding arrays of data in memory. The database *SQLite* was chosen due to its single-file design and the fact that it does not rely on a continuously running instance for I/O.

Given that the autonomy computer lacks any General Purpose Input/Output (GPIO), the control computer allows access to the external digital and analog world. A rapid data communication protocol was developed to allow the Autonomy Computer to make requests of the Control Computer when data was required. The Control Computer then responses with an acknowledgment along with indications that the proper message was received and information relating to the result of the request.

The protocol designed for this function is similar to the protocols used by the 3DM AHRS and the BlueRobotics Ping Sonar. All information is sent in hexadecimal format with identifiers which distinguish each individual packet. The anatomy of the packets are displayed in Figure 5.11. Every packet begins with two bytes: R (hexadecimal 0x52) and L (hexadecimal 0x4c). By searching for these two starting bytes, the receiver can clearly see when a new packet has started making packet error detection much easier. Next, a payload length is supplied. For the time being, this value is always four because this provides sufficient information for the operations currently required by the ASV. However, the system is designed to be easily updated



(a)



(b)

Figure 5.11: Data communication packet description between autonomy and control computers. (a) command packet, (b) response packet

when necessary.

After the payload length byte, a payload descriptor is placed. For packets acknowledging and replying to request, this value is 0x01. Command packets can consist of numerous descriptors, some of which are outlined in Table 5.6. Command packets have a space of four bytes to provide information about the requested operation. For example, to write an analog value to a given pin, the requester must provide the pin number (1 byte) and the analog value to set (2 bytes, similar to `analogWrite` in the Arduino environment). When responding to a request, the packet must include the command descriptor to ensure that the proper command is receiving a reply.

Table 5.6: List of packet descriptors for communication with control computer.

Descriptor	Description	Inputs	Outputs	Offset (Bytes)	Length (Bytes)
0x2	Read Analog Pin	Pin #	Request	0	1
			Pin #	1	1
			Value	2	2
0x4	Write Analog Value	Pin #, Value	Request	0	1
			Pin #	1	1
			Value	2	2
0x6	Get Status of Spektrum Receiver		Status	0	1
0x7	Set Motor RPM	Motor ID, PWM (Duty)	Request	0	1
			Motor ID	1	1
			Value	2	2

The presented data communication protocol allows for the control computer to become an extension of the autonomy computer. There are no major computing functions present on the Arduino, all actions require command over the serial port prior to execution. This has the limitation that, in the event of failure of the autonomy computer, the control computer cannot autonomously steer the ASV. This feature would make the system more robust to individual component failure. This would, however, require significant configuration of how the sensors post their in-

formation as the autonomy computer is the sole authority receiving the information required for autonomous control.

### 5.3 Mobile Sensor Network Configuration

Each ASV is considered a mobile sensor when it is equipped with an environmental monitoring device (e.g., a Ping Sonar). The pairing of this ASV with a ground station which sends commands and receives sensor data remotely is denoted a network. As additional ASVs are deployed, this network grows, and the burden of routing and error checking grows proportionally.

#### 5.3.1 Centralized Control

A single ground station is required within a given sensor network. This ground station handles the delegation of mission parameters and provides the user a view into the operational status of each ASV within the network as well as the network itself. The ground station consists of a Dell Vostro-7500 laptop, a Wi-Fi router and a RFD900x configured in multi-point mode. Similar to the ASV control software, the ground station ports USB devices through pseudo terminals prior to reaching the Matlab environment as shown in Figure B.2.

Missions are defined in an XML format which is loaded by the ground station. The commands within the XML format are translated into a NMEA-like communication protocol developed for messages between any agent in the network. To allow for robust communication between an undefined number of agents within the network, the following is necessary:

- Packet identifier

- Target descriptor
- Source descriptor
- Common command table
- Unique termination character

The protocol developed to meet these requirements is termed *ASVLang*. Much like the previously discussed NMEA-like protocols, it is a plain-text, comma-delimited string beginning with a command prefixed by a dollar sign. The command is followed by the command number, an integer which signifies the number of communications between source and target. Next, the name of the target followed by the name of the source. If necessary, a series of arguments can follow. The entire string is terminated by the carriage return character (hexadecimal 0x0d). The following is an example of an *ASVLang* command which adds a waypoint to the target’s waypoint list:

```
$AWAYPOINT,18,ALPHA,ONSHORE,5,508103.00000,3921432.00000,2.000\n
```

Specifically, this command is the 18th communication packet between the ASV “ALPHA” and the ground station (“ONSHORE”). It places a waypoint at 508103 meters east, 3921432 meters north in UTM zone 17S with capture radius of 2 meters as the 5th element in the agent’s waypoint list. The ASV is then responsible for moving through its waypoint list in ascending order. The *ASVLang* supports commands for the following:

- Acknowledge communication
- Start mission
- Pause mission
- Stop mission
- Set heading PID gains
- Add waypoint
- Remove waypoint
- Report vehicle state
- Report single point depth information
- Signal waypoint attainment
- Report error
- List waypoint in queue
- Mission state (Running, Paused)

The protocol is simple to extend to include additional commands as necessary and the number of arguments is not limited by the protocol, although bandwidth limitations may be restrictive given that it is a plain-text communication protocol.

The ground station implements an event system which gives the user extensive configurability for mission design. Received messages are checked for event triggers. When an event is triggered, the ground station can perform a pre-defined action such as broadcasting the next set of waypoints for an agent or run a specific function. This event system forms the foundation of the sensor network, allowing the user to limit the number of waypoints transmitted at a given time to reduce network traffic or to begin the computations required for the next adaptive sampling horizon.

### 5.3.2 Communication Topology

Remote communication is achieved through Wi-Fi as well as 900 MHz radio. Messages are broadcast over both protocols as a redundancy. The *ASVLang* protocol



is designed to handle packet repetition and ignore duplicate packets. Figure 5.12 depicts the broadcasting nature of the messaging system; all messages are received by all nodes within the network. Each node is then able to determine if they are the target of a specific message. The current network configuration consists of three nodes: Alpha, Bravo, and Onshore. Each node knows their name and the name of other nodes within the network to allow for message targeting.

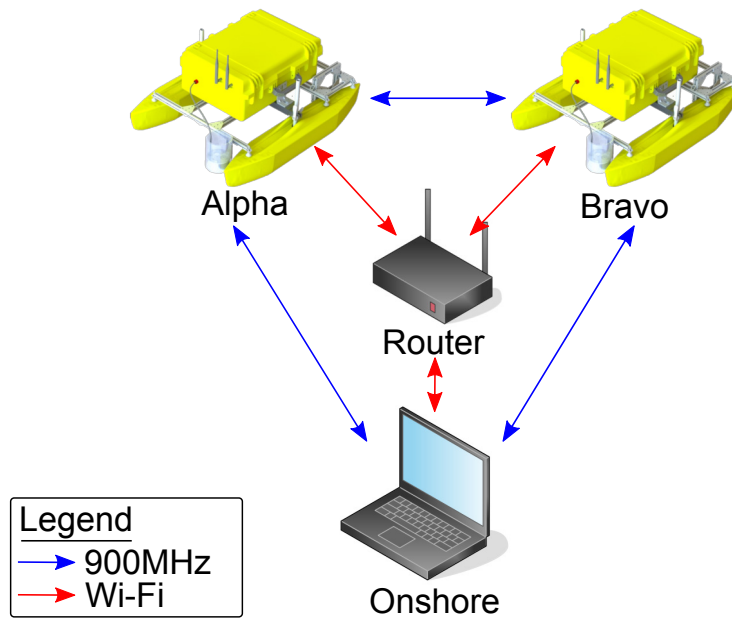


Figure 5.12: Example network topology for two agent network.

For messages over Wi-Fi, the User Datagram Protocol (UDP) was selected. UDP is a connection-less protocol, in contrast to the more standard Transmission Control Protocol (TCP) which makes a connection between a server and client. TCP provides error checking and packet re-transmission which is not available through UDP. However, UDP allows for all agents on the same subnet to receive a broadcast message. The additional benefits afforded by TCP are then implemented in a more

basic form through the *ASVLang* protocol.

The stock RFD900x is configured for point-to-point mode, meaning a pair of transceivers can only communicate with one another. A multi-point firmware supplied by RFDesign must be loaded onto each RFD900x within the network. By loading this firmware on all RFD900x's within the network, messages sent from any one modem are received on all models in the network. The only requirement is that a single master node be initialized to control the structure of the network. In the case of the UNCC ASV fleet, that master node is the RFD900x connect to the ground station.

### 5.3.3 Ground Station User Interface

A user interface was designed to display communication between all the agents and the ground station. The GUI was designed and built within Matlab's App Designer interface to comply with the goal of minimization dependencies outside of the Matlab environment. Figure 5.13 displays the main screen of the aforementioned user interface.

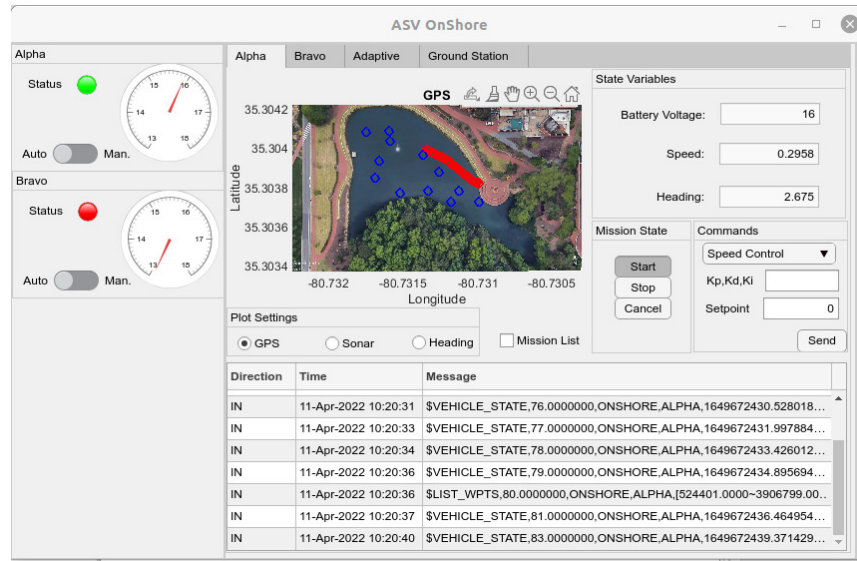


Figure 5.13: Ground station user interface for monitoring sensor network.

The GUI is split into five panels: (1) left status panel, (2) Alpha status panel, (3) Bravo status panel, (4) Adaptive status panel, and (5) ground station specific data panel. The left status panel is always visible and provides basic information about the deployed agents such as its communication status, battery level, and whether it is in manual or automatic mode. The status panels specific to the agents, Alpha and Bravo, are identically designed and provide information specific to the corresponding agent. As shown in Figure 5.13, the current location and waypoints can be viewed in real time as well as the readings from the sonar sensor or the heading of the agent over time. Also, in the table at the bottom of the panel, all communication between the agent and the ground station is displayed to assist with debugging any issues that may arise during operation. Controls are provided to start, stop and cancel the mission for a given agent. Finally, current battery level, instantaneous speed, and heading are displayed with numeric values in the upper right.

The panel for monitoring adaptive sampling calculation progress is designed to provide a generic interface to view the status of an external computation server. This server, running a separate instance of Matlab, computes the specific adaptive sampling mission as required. Data is passed between the GUI and the processing server via a combination of UDP commands, SQL database tables, and Matlab's proprietary save file format. This allows for a flexible system which can be easily extended to any type of adaptive mission. The ground station panel provides simple information related the operation of the GUI itself including internal error messages and the status of connected services such as the RFD900x and the external adaptive sampling processing server.

The ASV *OnShoreGUI* provides a simple interface for monitoring all the complex interactions required for monitoring and controlling a network of remote agents. All information gathered by the ground station is logged into a local SQL database to allow for real time querying of messages for debugging purposes or post-mission evaluation of critical systems.

#### 5.4 ASV Practical Control Design

As described in §4, each ASV is modeled with the closed loop dynamics (4.1). The aforementioned sensors provide the requisite state information for control. Heading information is gathered at a rate of 10 Hz from the IMU. A simple calibration is performed prior to operation in which the agent is pointed in the eastward direction, as determined by a cell phone compass application, and the appropriate taring command is executed. This process ensures that east represents  $0^\circ$  with positive rotation being counter-clockwise, identical to the assumptions within the simula-

tion environment. Additional calibration methods are supplied by the supplier such as hard- and soft-iron calibrations. These account for disturbances to the magnetometer due to magnetic objects and nearby metals respectively. These calibration procedures were found to be insufficient given the dynamic magnetic field generated by the nearby high-speed switching electronics (i.e. power supply, ESC, etc.) which produce varying magnetic effects over time.

Speed information is ascertained through the discrete differentiation of distance over time. Time is supplied by the operating system in POSIX time, the number of seconds elapsed since 12:00:00 UTC on January 1, 1970. Distance is determined by taking the Euclidean norm of the difference in UTM coordinates at the current and previous observations. Given noise associated with GPS, a simple boxcar filter is applied to the estimated velocity value to smooth the estimate over time.

While the waypoint tracking methodology is exactly transplanted onto the on-board controller from the simulation agent, alterations were made to the command equations (4.2) for practical concerns. A feed-forward term,  $\delta_f$ , was added to maintain agent motion to aid with speed calculations using differential spatial location over time. Additionally, an additional saturation is included to ensure that, in the event that the controller requires saturation to maintain speed, the turning radius is not affected. This situation may occur when disturbances cause the ASV to require greater than expected thrust to maintain speed. The resultant modified command equations are

$$\begin{aligned}
u_l &= \text{sat} \{ \text{sat} (\delta_f + \delta_v, u_{\min}, \tilde{u}_{\max}) - \delta_\theta, 0, u_{\max} \} \\
u_r &= \text{sat} \{ \text{sat} (\delta_f + \delta_v, u_{\min}, \tilde{u}_{\max}) + \delta_\theta, 0, u_{\max} \},
\end{aligned} \tag{5.1}$$

where  $u_{\min} < \tilde{u}_{\max} \leq u_{\max}$ . Testing was typically conducted with  $u_{\max} = 1.2$  kg·f,  $u_{\min} = 0.2$  kg·f,  $\tilde{u}_{\max} = 0.9$  kg·f, and  $\delta_f = 0.6$  kg·f. These force limitations were primarily chosen to obtain desired battery life while attaining reasonable speed; however, in the event of strong winds or other disturbances, these values would need to be updated.

## 5.5 Experimentation

This section describes the experiments performed to ensure capability of the design ASV system. The testing provides baseline performance characteristics and ensures the system's ability to meet research needs.

### 5.5.1 Wireless Communication Range Testing

Prior to experimentation with the ASV platform, tests were made to identify the functional range of the wireless communication systems to ensure safety of operation in large-scale testing. With the support of undergraduate researchers, John Driver and Connor Davidson, signal strength testing was conducted for Wi-Fi, RFD900x and the Spektrum R/C transceiver system (Spektrum DX8e transmitter and Spektrum AT8010T receiver). Tests were performed on UNCC's main campus during July 2021. Figure 5.14 shows the paths taken during each individual range test. An operator was placed at the origin point of all paths with an RFD900x transceiver,

a Spektrum DX8e transmitter and a Linksys Velop Wi-Fi router. Another operator carried a RFD900x transceiver, an AR8010T receiver and a Dell Vostro-7500 laptop connected to a BU-353S4 GPS for location tracking. As the mobile operator moved away from the origin point, Wi-Fi signal strength was logged automatically while the operator stationed at the origin manually logged the signal strength of the Spektrum system and the RFD900x. All devices were positioned at roughly ground level during testing.

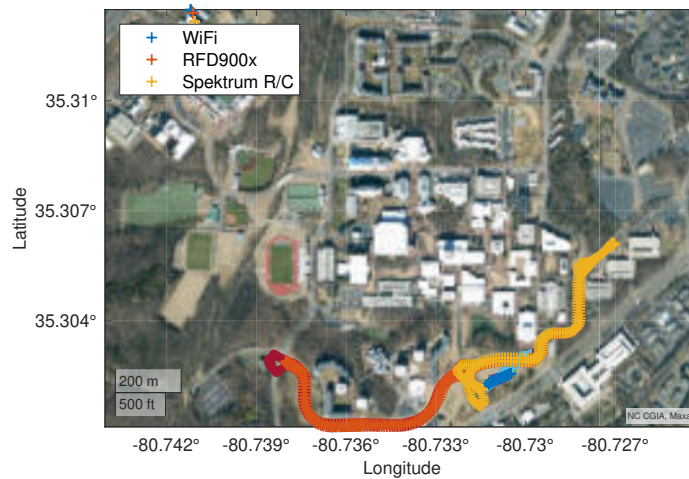


Figure 5.14: Wireless communication range testing paths.

The results of the wireless range testing are shown in Figure 5.15. From the resulting data, it can be concluded that manual control is reliable within 200 m when both transmitter and receiver are at ground level. Serial communication to the onboard autonomy computer can be assumed reliable up to 500 m, with redundancy through Wi-Fi connectivity up to 190 m.

As mentioned previously, these results represent the work case scenario for multiple

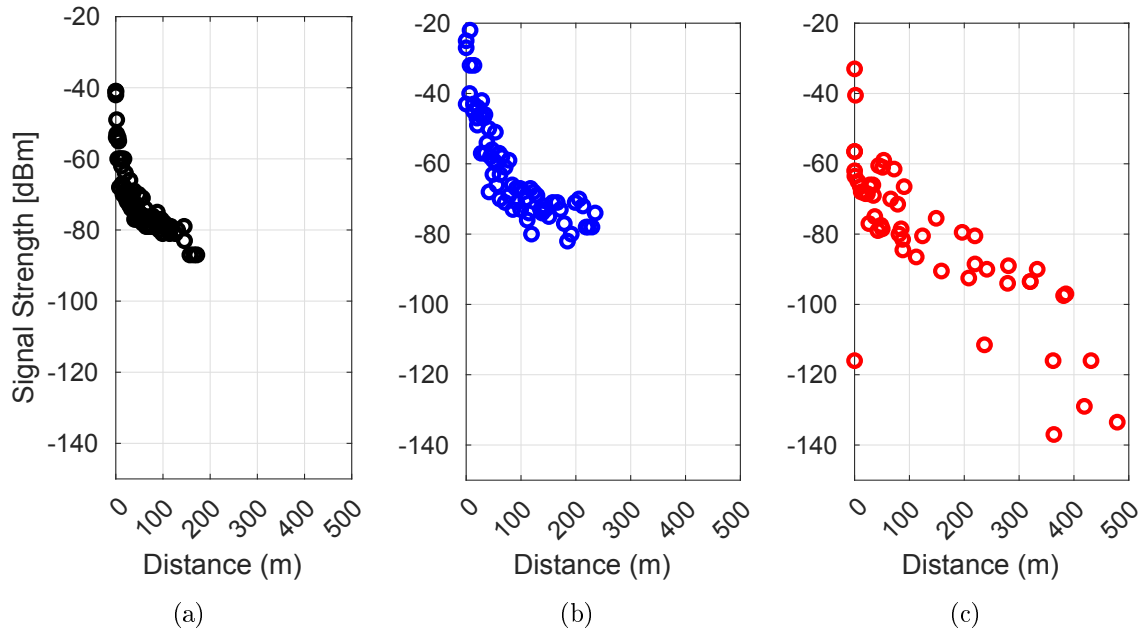


Figure 5.15: Signal strength over distance for (a) Wi-Fi, (b) R/C transmitter, and (c) RFD900x radio

reason: (1) transmitter and receiver are both held at roughly ground level, (2) close proximity to large buildings causes interference in wireless communication, and (3) communication between RFD900x's can be extended using multiple nodes and built-in multi-hop network configuration.

### 5.5.2 ASV Control Testing

Preliminary testing of the ASV platform began at Hechenbleikner Lake on the UNCC campus near Atkins Library. These tests were used to validate buoyancy, stability, remote control, maximum speed, heading controller functionality, and simple waypoint following. Finally, testing was conducted to ensure simultaneous multi-agent communication and controllability from a single ground station. Figure 5.16



shows the pair of ASVs deployed on Hechenbleikner Lake during testing to ensure controllability with multiple simultaneous agents.



Figure 5.16: Multi-agent testing on Hechenbleikner Lake.

Full speed testing results, shown in Figure 5.17, indicate that the current maximum attainable speed for the ASV platform is approximately 1.15 m/s. Although the upper goal of 2 m/s was not achieved, the system can meet the lower goal of 1 m/s.

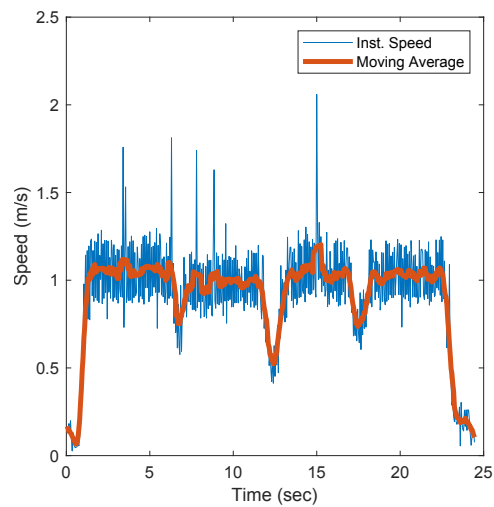


Figure 5.17: Maximum speed testing with remote control.

Results of the process for tuning of the gains for the PID controller used for heading

tracking are shown in Figure 5.18.

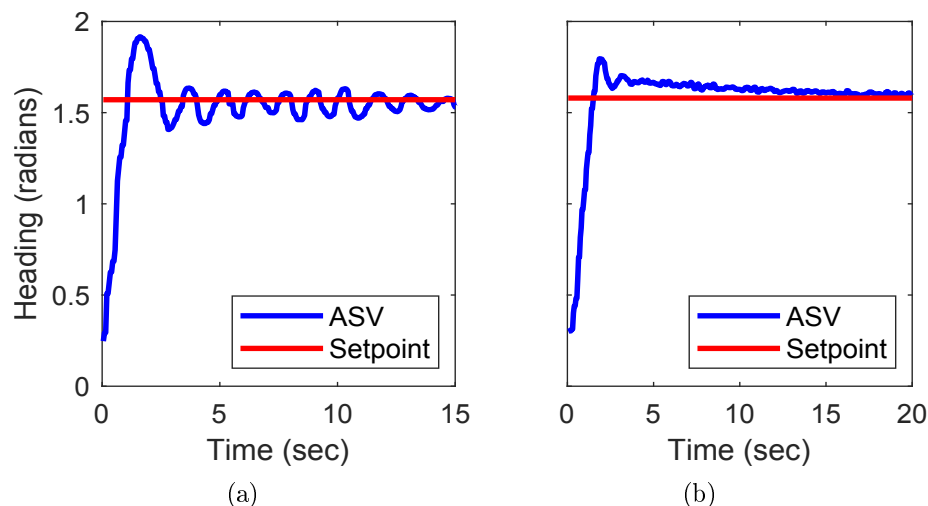


Figure 5.18: PID controller testing results (a) with minimal external disturbance and (b) with perpendicular 5-10 mph wind.

Once tuning of the heading controller was acceptable to continue, waypoint testing was performed on the same lake.

## 5.6 Design Review

A pair of Autonomous Surface Vessels were developed for use with current and ongoing research efforts in the Autonomous Robots and Systems Laboratory. These ASVs were developed to be low-cost, easy to deploy, and simple for research development. Experimentation was performed to validate performance and the results of which presented. Table 5.7 displays the desired characteristics of the surface vessels as well as the measured performance of the ASVs.

Although many of the desired criteria were achieved, there is clearly room for improvement. Future work should target improvements in top speed as well as sustained

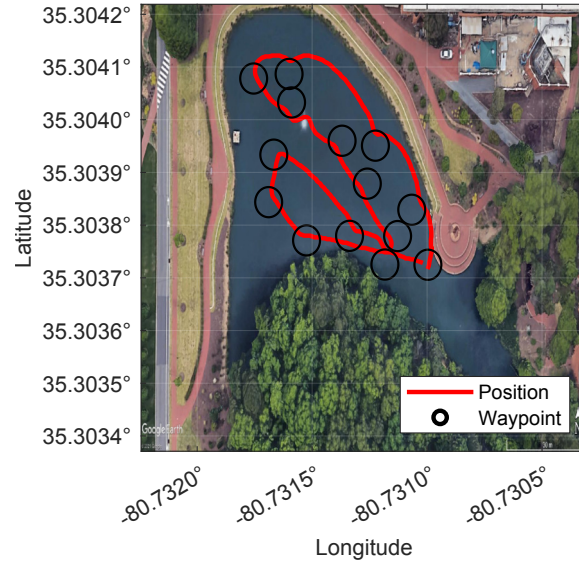


Figure 5.19: Waypoint control testing on Hechenbleikner Lake

Table 5.7: Actual performance characteristics comparison with desired.

	Length (m)	Width (m)	Max Speed (m/s)	Battery Life (h)	Comm. Range (km)	Weight (kg)
Target	0.9	0.7	1-2	1-2	1	<18
Actual	0.9	0.7	1.15	1.5	0.5	16

cruising speed. The initial calibration of the heading reference system can also be improved to ensure accurate tracking during missions. Also, additional testing should be performed to determine exact range of the RFD900x radios and modifications should be made for improvement as necessary. Finally, additional sensor packages should be integrated within the platform to allow for more advanced algorithmic research and experimentation.

## CHAPTER 6: EXPERIMENTAL RESULTS

Experiments were conducted to demonstrate the feasibility of the adaptive sampling algorithm developed in §4 to run onboard the ASVs described in §5. This chapter describes the process by which the experimentation was conducted, and the parameters used to achieve the provided results followed by an interpretation of the experimental outcomes.

### 6.1 Experimental Environment

Given the expected large length-scale for the variation of bathymetry, a large area was required to demonstrate the adaptive multi-agent survey. Additionally, this area had to be free from static impediments as the ASVs are not outfitted with object detection and avoidance. Finally, large bodies of water are typically public spaces, meaning that recreational boat traffic would need to be avoided as best as possible during testing.

Considering all of these challenges, an area near the Lake Norman Community Sailing Center on Lake Norman, NC was identified as a suitable testing location. Thanks to the altruism of the sailing center, testing was conducted directly from the pier for convenient loading and unloading of the vessels.

## 6.2 Ground Truth Survey

Prior to adaptive sampling testing, an exhaustive survey was conducted to generate a map of the bathymetry in the selected area with as high confidence as possible. A lawnmower pattern comprising of eight swaths spanning an area of 139 m wide by 189 m long was constructed and a single ASV agent performed the planned survey. Figure 6.1 displays the path taken by the ASV while tracking the waypoints constructing the lawnmower pattern. The experiment was conducted on March 28th, 2022 when there were winds of 5 mph perpendicular to the swaths with gusts up to 12 mph causing the vehicle's path to bend between the sparsely placed waypoints. Throughout these experiments, it was determined that a safe mission time could extend up to 1.5 hr at a cruising speed of approximately 0.5 m/s.

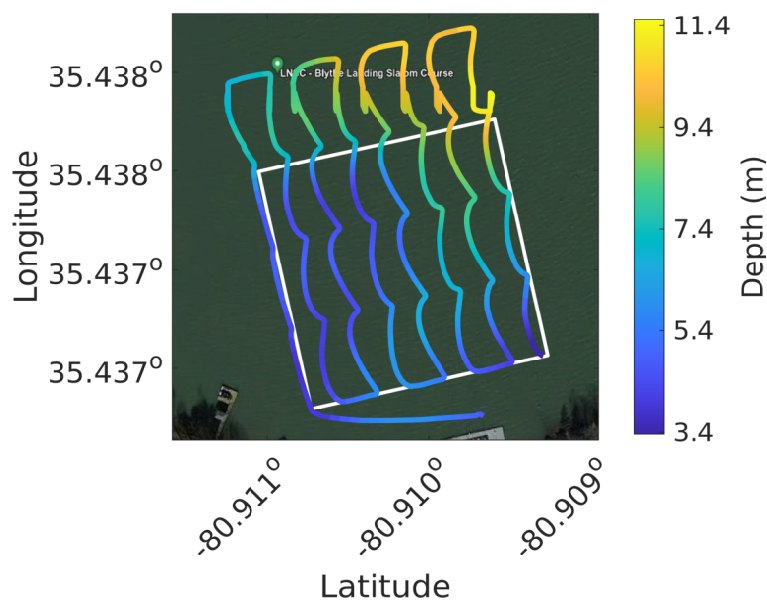


Figure 6.1: Waypoint control testing and bathymetric mapping near the Lake Norman Community Sailing Center.

Throughout the survey, the ASV was equipped with the BlueRobotics Ping Sonar for bathymetric mapping. Using the approximately 5000 data points collected during the survey, a semivariogram was fit assuming isotropy. Figure 6.2 shows the experimental semivariogram data points, the curve fit to this data, and the estimated range and sill. Using the estimated semivariogram function, a kriging estimate was performed over a uniform grid covering the entire survey region. The resulting field estimate is displayed in Figure 6.3. The estimate compares favorably to the publicly available bathymetric maps of the corresponding region [51].

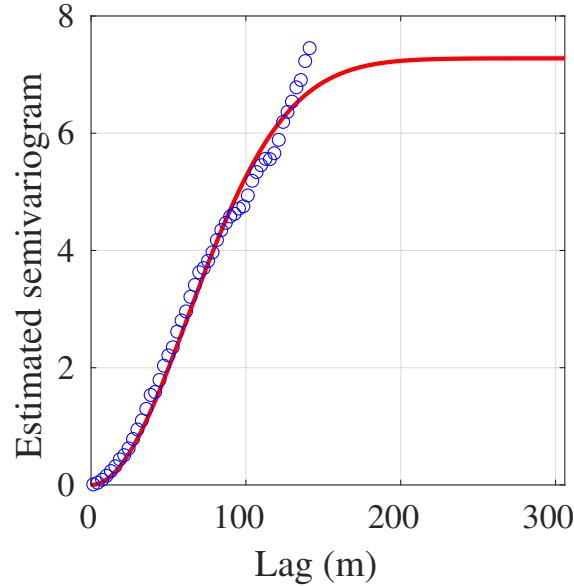


Figure 6.2: Experimental semivariogram fit to data collected during ground truth mission in the region near the Lake Norman Community Sailing Center.

Using the assumption of an isotropic second-order stationary field, Figure 6.2 shows the hyper-parameters determined by fitting an experimental semivariogram to the complete data set. The hyper-parameters identified by fitting the Gaussian

semivariogram experimental data are  $\boldsymbol{\theta} = [\zeta, \omega, \sigma_0^2]^\top = [0, 153\text{m}, 7.27\text{m}^2]^\top$ .

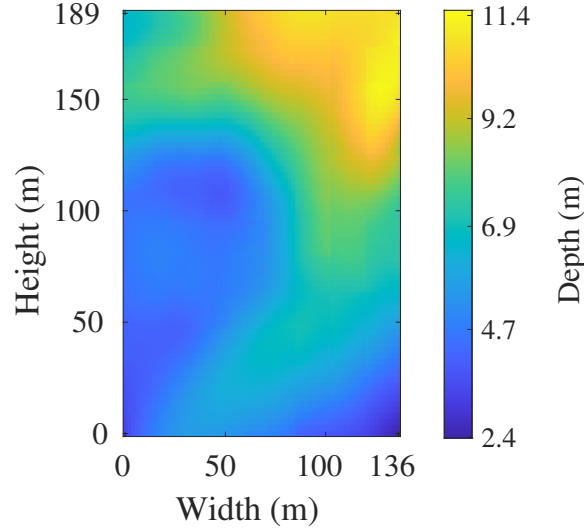


Figure 6.3: Estimate of bathymetry given echosounder measurements using kriging. The origin corresponds to the southwest corner of the white box in Fig. 6.1.

### 6.3 Experimental Methodology and Evaluation Criteria

For simplicity, a square subsection of the surveyed region was taken as the sample space. The white square in Figure 6.1 shows the  $140\text{ m} \times 140\text{ m}$  region chosen for this purpose. However, during experimentation, the adaptive sampling algorithm assumed a normalized space,  $\mathcal{Q} \triangleq [0, 1] \subset \mathbb{R}^2$ . This enabled the implementation used during simulation to be directly used for experiments.

Experimentation was conducted using both of the UNCC ASVs, Alpha and Bravo. Both agents were assumed to be traveling at the same speed with the homogeneous dynamics and therefore assigned the same number of swaths to cover within the sample space. Table 6.1 describes relevant parameters used to construct each mission that are common to both agents. Due to practical limitations, experiments

were limited to the assumption of homogeneous dynamics and sensing. Nonetheless, deploying in a realistic setting restricted to these assumptions proves the effectiveness of the proposed centralized, online adaptive path planning based in real-world environments.

Table 6.1: Common mission parameters among missions.

Parameter	Symbol	Value
Number of cycles	$N_c$	4
Number of waypoints	$N_w$	32
Number of waypoints per cycle	$C$	8
Preemptive re-planning offset	$p_r$	2
Sensing interval	$T_s$	10 s
Measurement noise variance	$\sigma_\eta^2$	Variable <sup>[1]</sup>
Mission time tolerance	$T_\Delta$	$\pm 10$ s

<sup>1</sup> Site specific measurement noise variance is 0.5% of measured depth [49].

Given the complexities of real time planning of multi-agent systems, the re-planning phase between cycles beginning at an offset of  $p_r$  waypoints prior to the next cycle. The first agent which reached this target triggered the computation on the ground station. This decision was made to help smooth the mission planning process in preparation for potentially long computation times. If planning time was sufficiently long, agents would drift from expected positions due to external disturbances, causing inaccuracies in estimation of mission time for proposed waypoint sets. Simulations were conducted under the assumption that  $p_r = 0$  given that planning time was effectively null as all agent simulation was paused during mission planning.

To compensate for a non-zero  $p_r$ , the waypoints remaining in the current cycle are left unmodified along with all previously reached waypoints in the `movePts` func-



tion [A2.7]. In the event that the two agents become out of sync, this process of freezing future waypoints ensures that all cycles re-plan a consistent number of waypoints for each agent through each mission. This consistency seeks to maintain as much similarity between the simulation and experimentation as possible.

Given limitations in the bandwidth during communication with multiple agents, the sensing interval,  $T_s$  was set at 10 seconds. Due to short mission times, the low number of expected measurements given infrequent communication, and the length-scale which is larger than any simulation scenario, *HFK* estimation was used rather than *HC* estimation. The measurement noise used for both agents with *HFK* was  $\sigma_\eta^2(\mathbf{x}) = 0.005d(\mathbf{x})$  where  $d(\mathbf{x})$  is the depth measured at spatial location  $\mathbf{x}$  based upon the manufacturer's specifications [49].

The expected time for the  $i$ th agent to pass through all of their respective waypoint assignments is denoted  $\tilde{T}_{m_i}$ . The effectiveness of the adaptive sampling algorithm in an experimental settings was judged based on three criteria: (1) mission time error in excess of allowable tolerance,  $T_{i,\epsilon} \triangleq \max\left(\left|T_m - \tilde{T}_{m_i}\right| - T_\Delta, 0\right)$ , (2) final total mission cost (4.6a), and (3) final mapping error (3.6) in comparison to a synthetically generated lawnmower survey with similar mission time constraints shown in Figure 6.3. Note that mapping error will not normally be available in field experimentation and the comparison is possible here due to the data obtained from the previous ground truth survey.

Comparison with an idealized control agent performing a variation of the original mission plan is used to compare effectiveness. Given that disturbances and deviation from the straight line path cause slower than expected average speed over the entire

Table 6.2: Mission parameters for experimental trials.

	Trial 1	Trial 2	Trial 3
Mission length (sec)	600	430	430
Target agent speed (m/s)	0.5	0.7	0.7
Number swaths per agent	3	2	2

mission, the set speed of the control agent is modified for equal comparison. The distance between waypoints  $j$  and  $j + 1$  for the  $i$ th agent,  $\mathbf{r}_{i,j}$  is summed to obtain the total ideal travel distance,  $d_{T_i} \triangleq \sum_{j=1}^{N_w-1} \|\mathbf{r}_{i,j}\|$ . The average mission velocity is then

$$\hat{v}_i = d_{T_i} / \tilde{T}_{m_i}. \quad (6.1)$$

The lawnmower path initially prescribed for the mission was modified to ensure that the control agent achieves all assigned waypoints within tolerance of mission target time by incrementally shrinking or expanding the search space and regenerating the waypoint pattern.

#### 6.4 Mission Description

On the 6th of May, 2022, multi-agent testing was conducted using the proposed adaptive algorithm on Lake Norman. A total of three successful missions were conducted during that time, the specific parameters of each mission are presented in Table 6.2. Agent speed and measurement noise variance are equal among the two agents for each mission.

Each agent was controlled from a position on the dock which was a maximum of 200 m from the furthest position of the ASVs. Each agent was driven close to the initial waypoint and the mission was started using the ground station GUI shown in

Figure B.1.

### 6.4.1 Experimental Trials

The first successful experimental mission began at 10:09:55 am. Given enough time to complete three swaths each when traveling 0.5 m/s, the paths taken by each agent are shown in Figure 6.4a, where each separate color shows the discrete planning cycles throughout the mission. The colored path in Figure 6.4b shows the sonar data sent to the ground station from each agent; close attention should be paid to the color scale on the right and the maximum and minimum values measured. In this case, the right-most agent, henceforth “Bravo”, measured a depth of up to 10.1 ft.

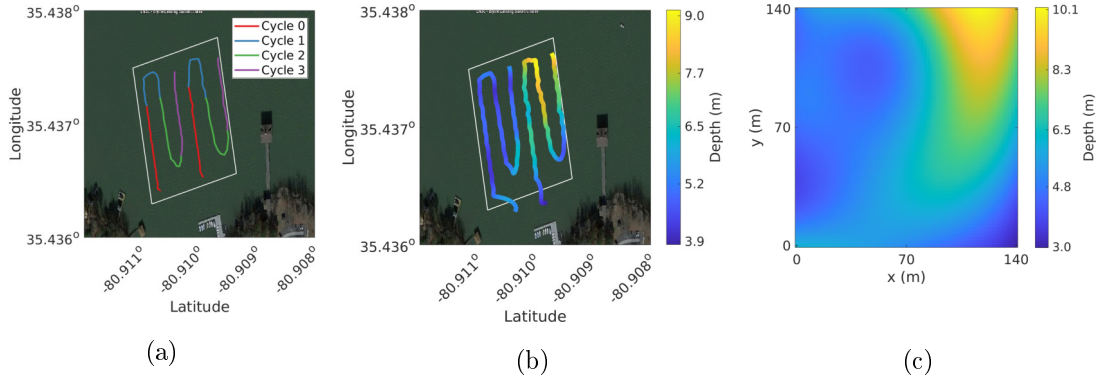


Figure 6.4: Experimental results from the first trial where  $T_m = 600$  s,  $v = 0.5$  m/s, and the number of pre-defined swaths was three per agent. Agent paths taken during each computation cycle is shown in 6.4a. Subplot 6.4b shows each agent’s mission path colored according to the bathymetric data gathered at that location. The final estimated scalar field given the observations gathered during the mission is shown in 6.4c.

Alpha, the left-most agent, has a relatively less interesting area over which it is surveying, i.e., it consists of no peaks or valleys within the total field. Bravo has the task of exploring the main valley in the upper right hand corner, best seen in

Figure 6.4c. Both agents can be seen avoiding thoroughly searching the area closest to the dock (south) given that the expectation is that very little variability occurs within this area. This allows each agent more time to explore the northern portion of the search space.

The second mission began at 10:30:37 am with  $T_m = 430$  s,  $v = 0.7$  m/s, and a target of two swaths each. Figure 6.5a and 6.5b both depict detailed information about the second trial. Immediate distinctions can be drawn from the previous Figure 6.4a with update parameters. Once again, the agents attempt to spend as little time in the southern portion of the search space. Alpha can be seen bending its path further towards the east to support Bravo in the search of the expectedly interesting space. Unfortunately, due to time constraints, Bravo was not able to return all the way to the northeast corner where the highest interest resides.

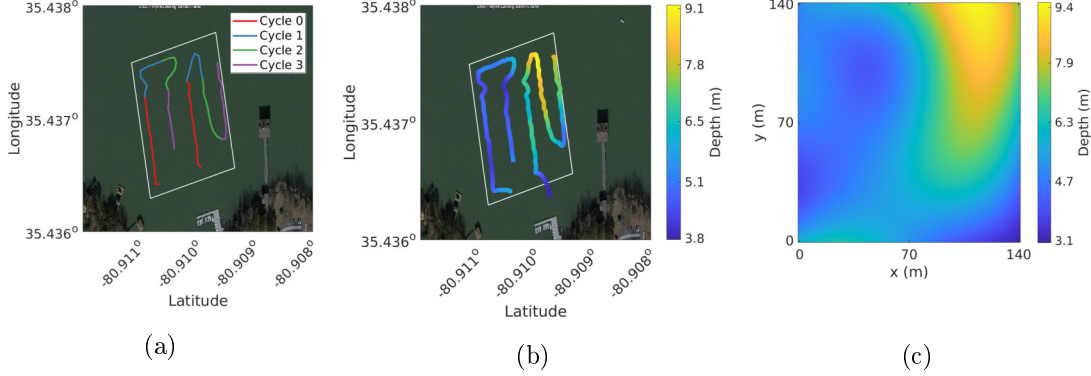


Figure 6.5: Experimental results from the second trial where  $T_m = 430$  s,  $v = 0.7$  m/s, and the number of pre-defined swaths was two per agent. The interpretation of subplot is the same as Figure 6.4.

The final mission began at 10:57:23 am with the same parameters as the previous trial. Notice that the paths and estimate are nearly identical. This is promising

for the reliability and repeatability of the real-world deployment of the developed algorithm.

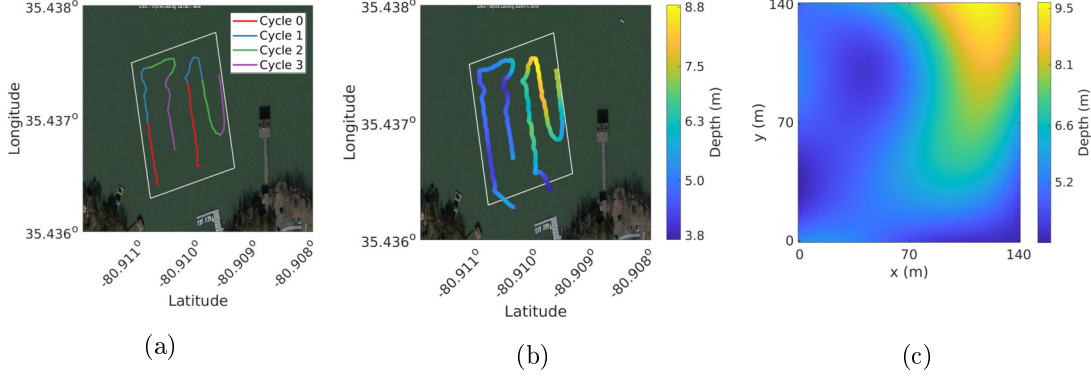


Figure 6.6: Experimental results from the third trial where parameters were identical to the second trial. Interpretations of the subplots are the same as Figure 6.4

#### 6.4.2 Discussion of Experimental Results

Using the ground truth scalar field, control agents were simulated in lawnmower patterns using the average mission velocity (6.1), total mission time, and initial swaths per agent to create the control path. Figure 6.7 depicts the comparison between the adaptive and control paths for each of the three trials wherein the adaptive value is normalized by the control value for each metric. Average mission time is the mean of the two agent's end time minus its start time. The final cost of the estimated field is calculated using (4.5) over a uniform grid of points and summed. The final mapping error is calculated using (3.6) comparing the mapping estimate and the map generated using the ground truth survey. The red line represents parity at a ratio of 1:1. When the bar exceeds the red line, the adaptive value is greater than the control and vice versa. The paths taken by each simulated control agent is

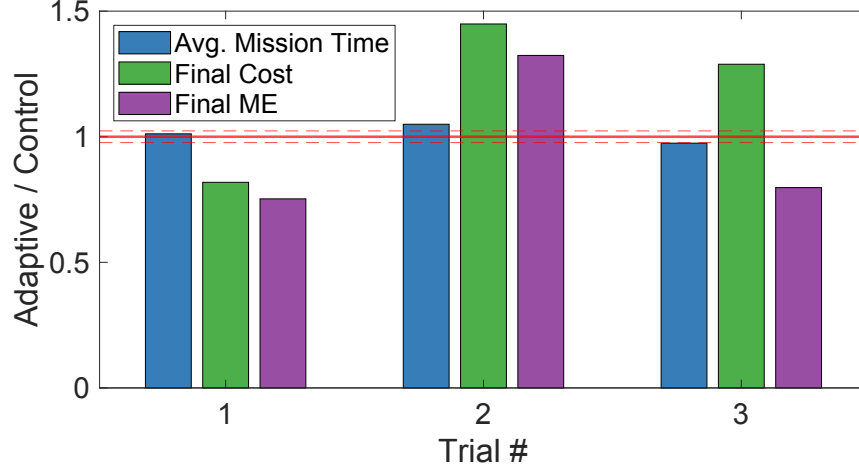


Figure 6.7: Comparison of experimental results when compared with mission objectives and control simulations. The solid red line indicates a ratio of 1:1, or parity, between the adaptive and control sampling missions. When the bar chart exceeds the line, the adaptive control is larger than control and vice versa. The dashed red lines represent the mission time tolerance,  $T_{\Delta}$ .

shown in Figure A.1.

As shown, in trial 1, each agent achieved the target mission time with high accuracy as well as achieving improved cost and mapping error performance when compared to the control simulation. This can be attributed to Bravo’s return to the point of greatest depth towards the end of its mission, shown in Figure 6.4b. The successes of the first trial are not mirrored in the next two trials. The control outperforms the adaptive in final cost in trials 2 and 3 while achieving long mapping error in trial 2. It should be noted that the higher final cost but lower mapping error for the adaptive agent in trial 3 is coincidental as the algorithm seeks to minimize cost, which serves as a substitute for actual mapping error, and therefore makes no guarantees for improved mapping error.

The significance of trials 2 and 3 is the similarity between distinct missions. When

comparing Figures 6.5b and 6.6b, it is apparent that the paths taken are incredibly similar. This is a promising sign of the repeatability of the adaptive algorithm. The variability in the final resulting cost and mapping error can be explained by the slight variability in external disturbances causing variable average mission speed which changes the reachable space among missions. Given the low variability in the scalar field being measured, drastic changes in final mapping error and cost can be achieved simply by moving closer to the largest measured depth in the northeast corner.

Given the restrictions for the selection of a real-world sampling region, the numerical results of the experimentation remain somewhat inconclusive. Future experimentation should address logistical constraints inherent in conducting large-scale experimentation in public spaces. An expanded survey region of approximately nine times the area of the conducted experiment should yield more conclusive results given that the space would then have a length-scale which represents roughly 33% of the space's width and height. However, this expansion may require extension of the current program which runs the algorithm to handle potentially convex shapes to accommodate the coast lines. Additionally, a larger team will be required for the coordination of manned boats to patrol the survey area and ensure there are no problems with recreational users of the space as well as the police, who had confronted the ASVs on previous outings. Finally, alterations to the battery and its management system would be required to ensure that missions of the required scale could be conducted more than once on a single charge. The designed system is currently incapable of continuing an interrupted mission which means that in the event of a

failure, the mission would need to restart from the beginning. The current design would only be capable of a maximum of two attempts per charge in a sample space of the recommended size.



## CHAPTER 7: CONCLUSION AND FUTURE WORK

A novel approach for the efficient computation of online kriging estimates of a stationary, isotropic scalar process with multiple heterogeneous sensing sources was described in §3. This work combined the methods of common data neighborhoods [41] and heterogeneous-measurement-error filtered kriging [34] to allow for rapid estimation of fields with sets of training data. Presented analysis concludes that *HC* yields lower computation times with minimal impact on accuracy when the length-scale of the sample field is smaller than the field’s major dimensions. The estimation process was then utilized in an algorithm developed for the time-constrained exploration of a square sampling region using multiple agents, described in §4. Founded on the principles of tessellation into centroidal Voronoi cells, this efficient planning algorithm was shown to outperform an offline lawnmower approach given varying field hyperparameters, agents dynamics, and agent sensing characteristics. Simulations of the proposed adaptive sampling algorithm show that gains are made given heterogeneous measurement noise variance and dynamics when compared with a lawnmower pattern.

An autonomous surface vessel platform was developed for UNC Charlotte’s Autonomous Robots and Systems Laboratory for research in inland bodies of water. A pair of these low-cost and modular vessels was constructed for the real-time implementation of the developed adaptive sampling algorithm. The process of design and

construction was outlined in detail in §5. An implementation of the adaptive algorithm was developed for use with the vessels in conjunction with a ground station and experimentation proceeded near the Lake Norman Community Sailing Center. A number of trials were conducted which sought to provide insight into the viability of the proposed algorithm and testing platform in real-world missions. Success of these missions was judged by key performance characteristics. Additionally, robustness of the designed system was determined through the repeated successful completion of multi-agent missions.

Ongoing work should focus on the extension of ASV battery life and improved controllability over 900 MHz radio without reliance on secure shell over Wi-Fi. Estimation should be improved through the parallelization of the *HC* algorithm as well as calculation of only neighborhoods which updates are expected given most recently measurements. Additionally, techniques utilizing recursive LU decomposition [52] and kriging with sparse matrices [53] should be explored for improved computation time on a per neighborhood basis. The algorithm should also be tested for robustness in situations where the number of agents exceeds two. Finally, the convergence methodology for updating of spring-mass-damper system parameters should be improved and convergence guarantees should be explored.

## REFERENCES

- [1] C. Zhang and J. M. Kovacs, “The application of small unmanned aerial systems for precision agriculture: a review,” *Precision Agriculture*, vol. 13, no. 6, pp. 693–712, 2012.
- [2] J. Pulido Fentanes, A. Badiee, T. Duckett, J. Evans, S. Pearson, and G. Cielniak, “Kriging-based robotic exploration for soil moisture mapping using a cosmic-ray sensor,” *Journal of Field Robotics*, vol. 37, no. 1, pp. 122–136, 2020.
- [3] T. O. Fossum, J. Eldsvik, I. Ellingsen, M. O. Alver, G. M. Fragoso, G. Johnsen, R. Mendes, M. Ludvigsen, and K. Rajan, “Information-driven robotic sampling in the coastal ocean,” *Journal of Field Robotics*, vol. 35, pp. 1101–1121, 2018.
- [4] E. Galceran and M. Carreras, “Robotics and autonomous systems,” *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.
- [5] S. Mishra, S. Rodriguez, M. Morales, and N. M. Amato, “Battery-constrained coverage,” in *IEEE International Conference on Automation Science and Engineering*, (Fort Worth, TX, USA), pp. 695–700, 2016.
- [6] M. Wei and V. Isler, “Coverage path planning under the energy constraint,” in *IEEE Internatinoal Conference on Robotics and Automation*, pp. 368–373, 2018.
- [7] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, “Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring,” *IEEE Transactions on Automation Science and Engineering*, vol. 18, pp. 1453–1468, 2021.
- [8] S. Agarwal and S. Akella, “Area coverage with multiple capacity-constrained robots,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3734–3741, 2022.
- [9] D. A. Paley and A. Wolek, “Mobile sensor networks and control: Adaptive sampling of spatiotemporal processes,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 91–114, 2020.
- [10] S. Bai, T. Shan, F. Chen, L. Liu, and B. Englot, “Information-driven path planning,” *Current Robotics Reports*, vol. 2, pp. 177–188, 2021.
- [11] N. Cressie, “The origins of kriging,” *Mathematical Geology*, vol. 22, pp. 239–252, 1990.

- [12] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Boston, USA: MIT Press, 2006.
- [13] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, “Grid coverage for surveillance and target location in distributed sensor networks,” *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.
- [14] J. Liu and C. D. Laird, “A global stochastic programming approach for the optimal placement of gas detectors with nonuniform uniform unavailabilities,” *Journal of Loss Prevention in the Process Industries*, vol. 51, pp. 29–35, 2018.
- [15] Y. Zou and K. Chakrabarty, “Sensor deployment and target localization based on virtual forces,” in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 1293–1303, 2003.
- [16] G. Wang, G. Cao, and T. F. L. Porta, “Movement-assisted sensor deployment,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, pp. 640–652, 2006.
- [17] A. Howard, M. J. Matarić, and G. S. Sukhatme, “Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem,” in *Proc. of the 6th International Symposium on Distributed Autonomous Robotics Systems*, (Fukuoka, Japan), pp. 299–308, June 2002.
- [18] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis, “Collective motion, sensor networks, and ocean sampling,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 48–74, 2007.
- [19] N. Sydney and D. A. Paley, “Multivehicle coverage control for a nonstationary spatiotemporal field,” *Automatica*, vol. 50, pp. 1381–1390, May 2014.
- [20] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsely, “Mobility improves coverage of sensor networks,” in *Proc. of the 6th ACM International Symposium on Mobile ad hoc Networking and Computing*, (Urbana-Champaign, Il, USA), pp. 300–308, 2005.
- [21] R. Cui, Y. Li, and W. Yan, “Mutual information-based multi-AUV path planning for scalar field sampling using multidimensional RRT\*,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 993–1004, 2016.

- [22] Y. Xu, J. Choi, and S. Oh, “Mobile sensor network navigation using Gaussian processes with truncated observations,” *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1118–1131, 2011.
- [23] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin, “Efficient planning of informative paths for multiple robots,” in *Proc. of the 20th International Joint Conference on Artificial Intelligence*, (San Francisco, CA, USA), pp. 2204–2211, 2007.
- [24] A. Krause and C. Guestrin, “Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach,” in *Proceedings of the 24<sup>th</sup> International Conference on Machine Learning*, pp. 449–456, 2007.
- [25] R. Graham and J. Cortés, “Adaptive information collection by robotic sensor networks for spatial estimation,” *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1404–1419, 2012.
- [26] R. Marchant and F. Ramos, “Bayesian optimisation for intelligent environmental monitoring,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vilamoura, Algarve, Portugal), pp. 2242–2249, 2012.
- [27] M. Jadaliha and J. Choi, “Environmental monitoring using autonomous aquatic robots: Sampling algorithms and experiments,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 3, pp. 899–905, 2013.
- [28] N. A. Cressie, *Statistics for Spatial Data*. New York: Wiley-Interscience, 1993.
- [29] K.-C. Ma, L. Liu, and G. S. Sukhatme, “Data-driven learning and planning for environmental sampling,” in *2017 IEEE International Conference on Robotics and Automation*, (Singapore), 2017.
- [30] J. H. Shapiro, “Lecture notes in optical propagation, detection, and communication.” <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-453-quantum-optical-communication-fall-2016/readings-and-lecture-slides/>, 1988. (accessed 2021-11-10).
- [31] R. A. Olea, *Geostatistics for Engineers and Earth Scientists*. Dordrecht: Kluwer Academic Publishers Group, 1999.
- [32] V. Schmidt, *Stochastic Geometry, Spatial Statistics and Random Fields: Models and Algorithms*. Switzerland: Springer International Publishing, 2015.

- [33] E. Brochu, V. M. Cora, and N. de Freitas, “A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning.” arXiv preprint arXiv:1012.2599, 2010.
- [34] W. F. Christensen, “Filtered kriging for spatial data with heterogeneous measurement error variances,” *Biometrics*, vol. 67, no. 3, pp. 947–957, 2011.
- [35] N. Cressie, “Spatial prediction and ordinary kriging,” *Mathematical Geometry*, vol. 20, no. 4, pp. 405–421, 1988.
- [36] J.-R. Sack and J. Urrutia, *Handbook of Computational Geometry*. Amsterdam, The Netherlands: Elsevier Science B.V., 2000.
- [37] Q. Du, V. Faber, and M. Gunzburger, “Centroidal Voronoi tessellations: Applications and algorithms,” *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.
- [38] S. P. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [39] Y. Xu, J. Choi, and S. Oh, “Mobile sensor network navigation using Gaussian processes with truncated observations,” *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1118–1131, 2011.
- [40] J. P. C. Kleijnen and W. C. M. van Beers, “Prediction for big data through kriging: Small sequential and one-shot designs,” *American Journal of Mathematical and Management Sciences*, vol. 39, no. 3, pp. 199–213, 2020.
- [41] M. Vigsnes, O. Kolbjornsen, V. L. Hauge, P. Dahle, and P. Abrahamsen, “Fast and accurate approximation to kriging using common data neighborhoods,” *Mathematical Geosciences*, vol. 49, pp. 619–634, 2017.
- [42] J. C. Hateley, H. Wei, and L. Chen, “Fast methods for computing centroidal Voronoi tessellations,” *Journal of Scientific Computing*, vol. 63, pp. 184–212, 2014.
- [43] OceanAlpha, “ME120: Hydrographic unmanned surface vehicle.” <https://www.oceanalpha.com/product-item/me120/>. (accessed: 2022-04-01).
- [44] EvoLogics, “Sonobot 5.” <https://evologics.de/sonobot-5-system>. (accessed: 2022-04-01).
- [45] Maritime Robotics, “The Otter: Unmanned surface vessel [USV].” <https://www.maritimerobotics.com/otter>. (accessed: 2022-04-01).

- [46] LORD-MicroStrain, “MSCL.” <https://github.com/LORD-MicroStrain/MSCL>, 2015. (accessed: 2022-04-04).
- [47] SiRF Technology Inc., “NMEA reference manual.” [http://inventeksys.com/wp-content/uploads/NMEA\\_Reference\\_Manual.pdf](http://inventeksys.com/wp-content/uploads/NMEA_Reference_Manual.pdf), 2007. (accessed: 2022-04-03).
- [48] United States Geological Survey, “The universal transverse mercator (UTM) grid.” <https://pubs.usgs.gov/fs/2001/0077/report.pdf>, 2001. (accessed: 2022-04-03).
- [49] BlueRobotics, “T200 performance charts.” <https://cad.bluerobotics.com/T200-Public-Performance-Data-10-20V-September-2019.xlsx>. (accessed: 2021-06-11).
- [50] “The GNU C library.” <https://www.gnu.org/software/libc/>. (accessed: 2022-05-20).
- [51] fishermap.org, “Norman Lake nautical chart.” <https://usa.fishermap.org/depth-map/lake-norman/>. (accessed: 2022-04-03).
- [52] K. Georgiev and J. Waśniewski, “Recursive version of lu decomposition,” in *Numerical Analysis and Its Applications* (L. Vulkov, P. Yalamov, and J. Waśniewski, eds.), (Berlin, Heidelberg), pp. 325–332, Springer Berlin Heidelberg, 2001.
- [53] B. Farmanesh and A. Pourhabib, “Sparse pseudo-input local kriging for large spatial datasets with exogenous variables,” *IISE Transactions*, vol. 52, pp. 334–348, jul 2019.

# APPENDIX A: Experimental Results Supplemental Information

Supplemental information pertaining to the experimental results gathered on Lake Norman, NC.

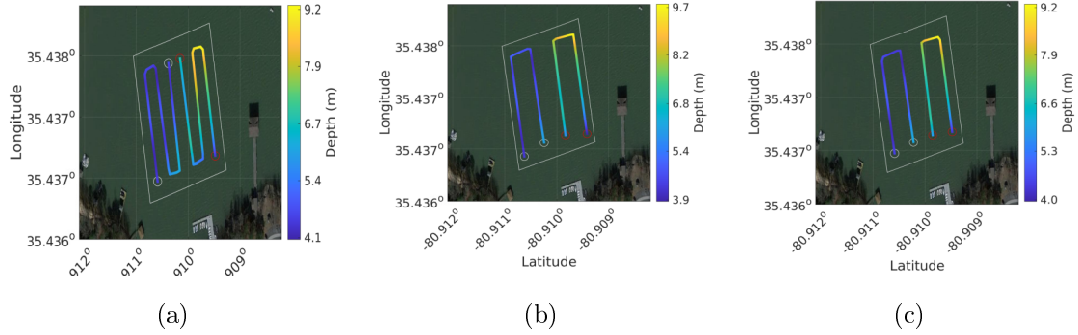


Figure A.1: Paths taken by control agents when comparing to experimental trials. (A.1a) Trial 1, (A.1b) Trial 2, (A.1c) Trial 3

Fishermapping.org is a freely available mapping service which has topographical information for lakes within Belarus, Kazakhstan, Ukraine, United Kingdom, and the United States. Figure A.2 is a snapshot of the region used for experimentation.

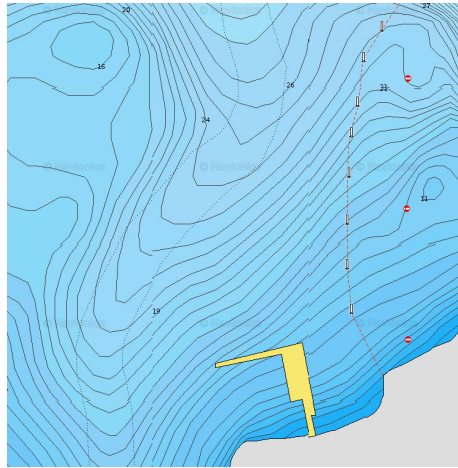


Figure A.2: Topographical map of Lake Norman [51].



## APPENDIX B: System Design Diagrams

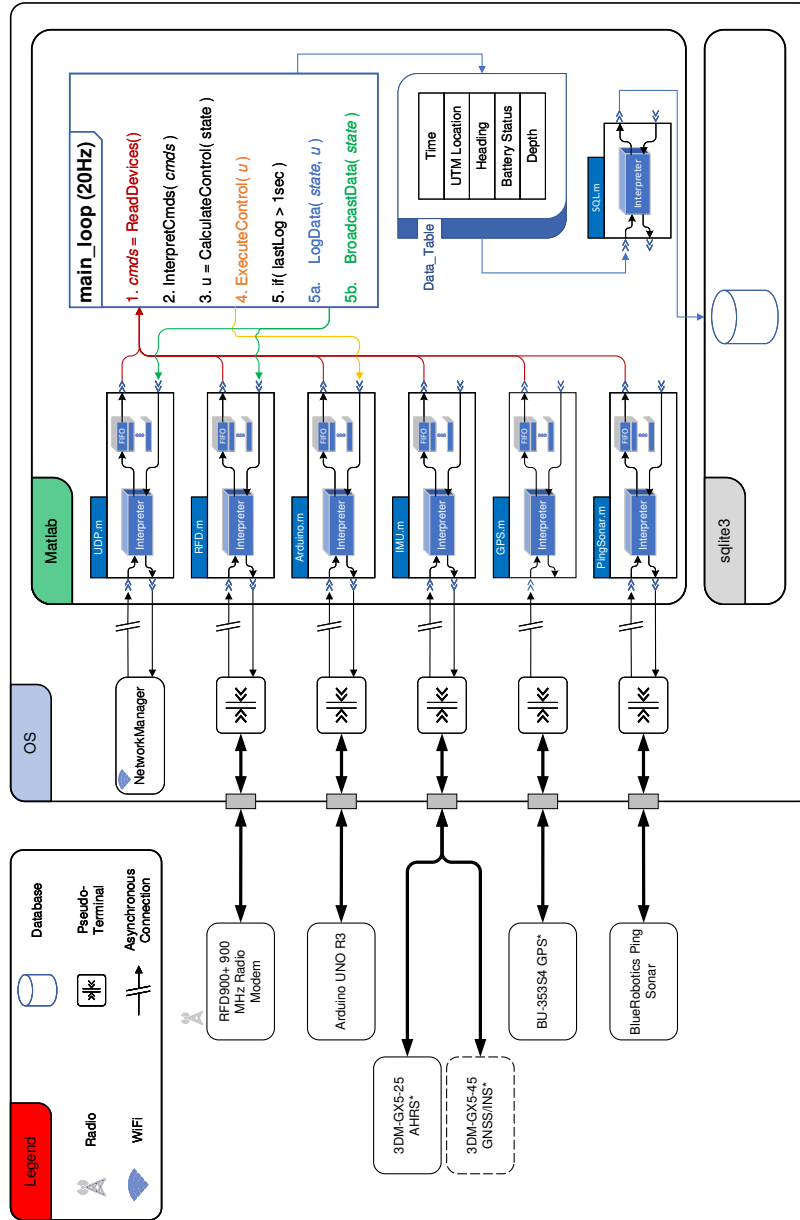


Figure B.1: Data flow diagram for ASV onboard software.

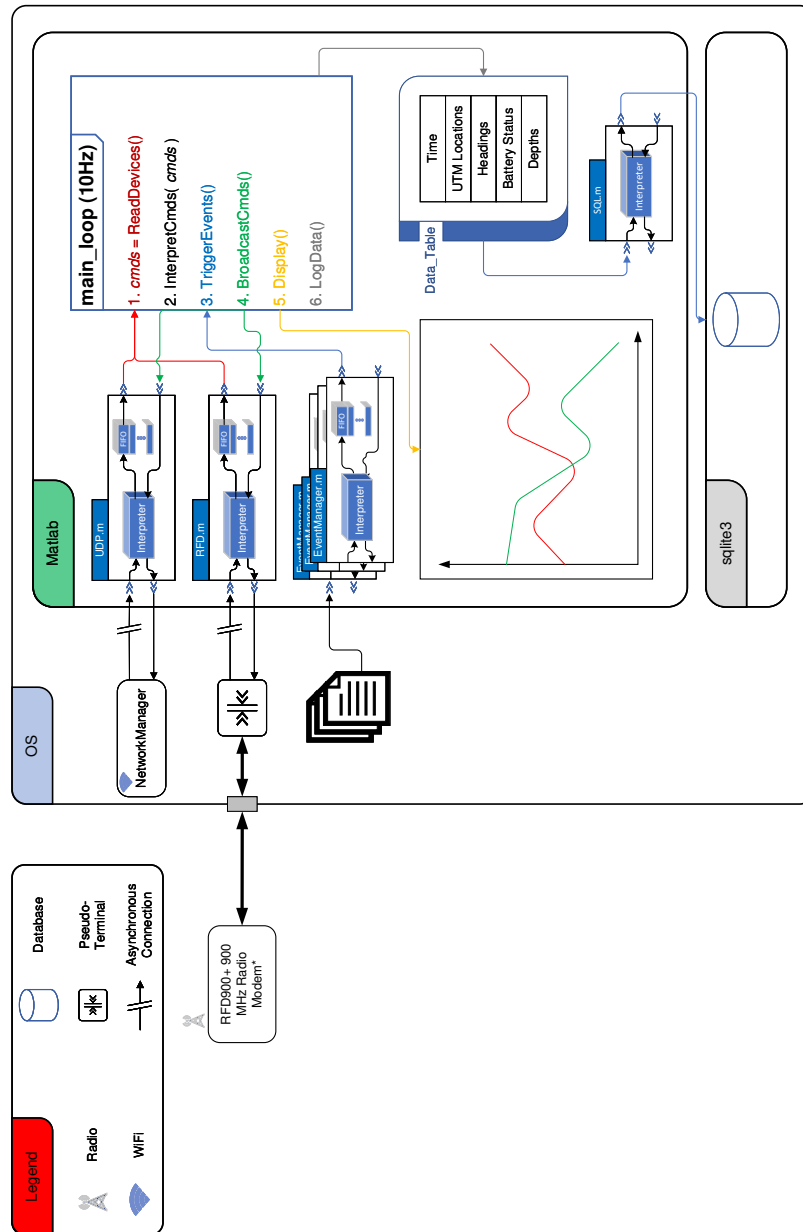


Figure B.2: Data flow diagram for the ground station.

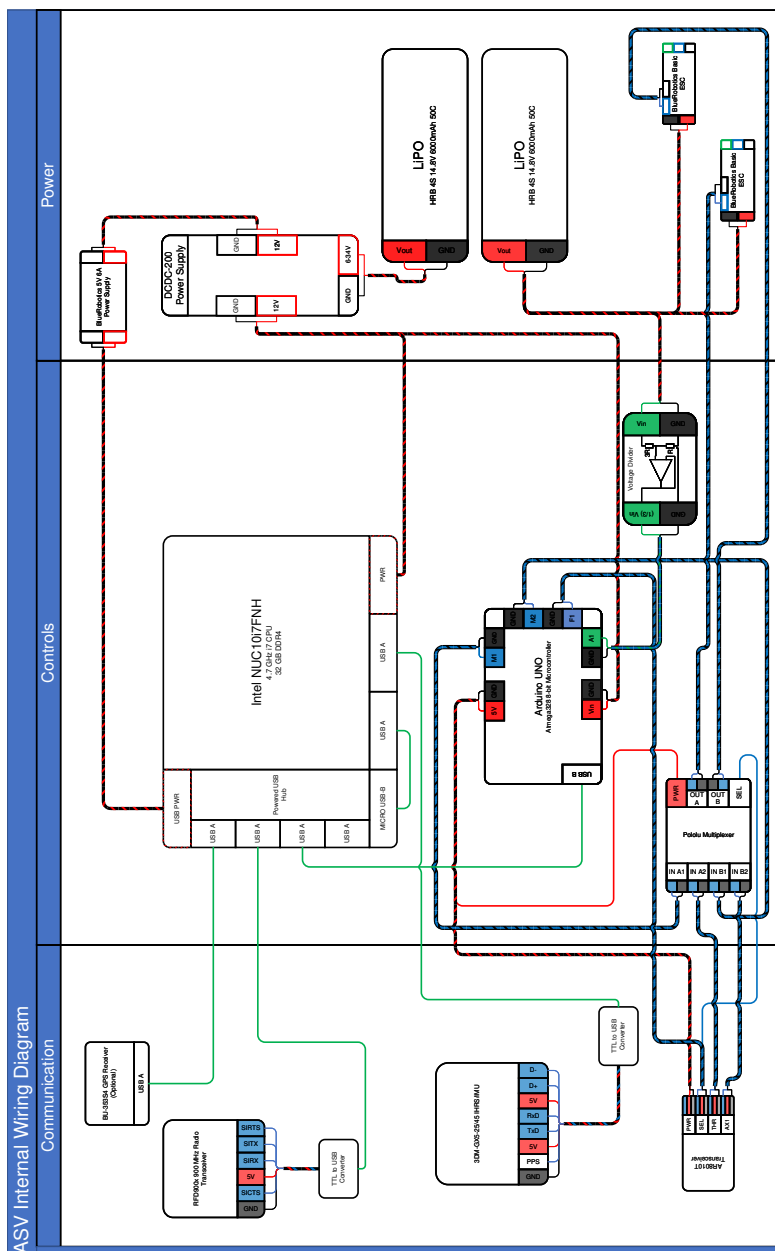


Figure B.3: Detailed ASV wiring diagram.

## APPENDIX C: Bill of Materials

The complete Bill of Materials (BOM) for each individual ASV is displayed in C.1. The components required for the ground station, only one required per network, is found in C.2. The BOM is divided into levels, with level 1 being the entire system. Level 2 is framing, propulsion, and electrical system. Level 3 is the sub-assemblies within those three categories with level 4 adding additional component definition where required. An asterisk following the BOM level indicates that it is an alternative component and not included in cost roll-up.

The components are broken down into four categories: assembly, sub-assembly, purchased component, and purchased component requiring major modification. In the BOM, these are identified as ASSY, SUB-ASSY, PUR, and MOD-PUR respectively. This helps identify how most components are used directly in the system without requiring major modification which is defined as any physical modification (e.g., cutting, grinding, soldering).

Table C.1: Detailed ASV Bill of Materials.

BOM Level	1	2	3	4	Name	Description	Type	Unit Cost (\$)	Quantity	Sub Total (\$)
1	X				Autonomous Surface Vessel	Top Level Assembly	ASSY	4,757.07	1	4757.07
2		X			Chassis	Framing, flotation and propulsion	SUB-ASSY			
3			X		Pontoons	Spring Creek Manufacturing Canoe Stabilizer	MOD-PUR	370.00	1	370.00
4				X	Framing	Extruded Aluminum Framing	SUB-ASSY			
3				X	Extruded Aluminum (2 ft)	20 mm T-slotted Extruded Aluminum pre-cut to 2 ft	MOD-PUR	8.03	5	40.15
4				X	Extruded Aluminum (3 ft)	20 mm T-slotted Extruded Aluminum pre-cut to 3 ft	MOD-PUR	10.38	1	10.38
4				X	Right Angle Bracket	Right Angle Bracket for 20 mm Extruded Aluminum	PUR	12.53	12	150.36
4				X	T-Bracket	T-Bracket for 20 mm Extruded Aluminum	PUR	15.07	10	150.70
4				X	Corner Bracket	Corner Bracket for 20 mm Extruded Aluminum	PUR	3.75	6	34.50
4				X	Swivel Leveling Mount	Swivel Leveling Mount	PUR	8.15	4	32.60
4				X	Hinge	Plastic hinge for 20 mm Extruded Aluminum	PUR	5.86	3	17.58
3			X		Large Corner Bracket	Galvanized Steel Corner Bracket	MOD-PUR	4.42	2	8.84
3			X		Water-tight Nut	Rubber-Coated Brass Insulating Rivet Nut	PUR	1.09	2	2.18
3			X		Locking Pin	1/4" T-Handle Locking Quick-Release Nut	PUR	26.86	2	53.72
3			X		1/4" Bolt	Zinc Plated Alloy Steel Socket Head Screw	PUR	0.33	2	0.66
3			X		1/4" Bolt	Stainless Steel Hex Nut	PUR	0.08	2	0.16
3			X		5/16" Bolt	Black-Oxide Alloy Steel Socket Head Screw	PUR	0.41	2	0.82
3			X		5/16" Nut	Stainless Steel Hex Nut	PUR	0.10	2	0.20
3			X		5/16" Washer	Stainless Steel Washer	PUR	0.14	2	0.27
2		X			Propulsion	Propulsion System	SUB-ASSY			
3			X		Thrustor	BlueRobotics T200 Thruster	PUR	179.00	2	358.00
3			X		Thrustor Mount	BlueRobotics Thruster Mounting Bracket	PUR	6.00	2	12.00
3			X		Bulkhead Connector	Blue Trail Engineering Cobalt Bulkhead Termination Kit	PUR	42.00	2	84.00
3			X		Cable Connector	Blue Trail Engineering Cobalt Cable Termination Kit	PUR	25.00	2	50.00
2		X			Electrical Components	Electrical components and corresponding structure inside Pelican Case	SUB-ASSY			
3			X		Electrical Enclosure	iM2600 Pelican Case with Foam	MOD-PUR	182.95	1	182.95
3			X		Electrical Rigging	Extruded Aluminum framing for mounting of electrical components	SUB-ASSY			
4			X	X	Extruded Aluminum (300 mm)	15 mm Extruded Aluminum framing from MakerBeam XL	PUR	4.34	5	21.70
4			X	X	Extruded Aluminum (150 mm)	15 mm Extruded Aluminum framing from MakerBeam XL	PUR	2.54	4	10.16
4			X	X	Right Angle Bracket	MakerBeam XL Right Angle Bracket	PUR	0.98	8	7.82
4			X	X	T-Bracket	MakerBeam XL T-Bracket	PUR	0.98	2	1.96
4			X	X	6 mm Bolts	MakerBeam 6 mm bolts	PUR	0.05	30	1.62
4			X	X	6 mm Nuts	MakerBeam 6 mm nuts	PUR	0.06	30	1.69
4			X	X	Terminal Block	12 position 300 V DC Terminal Block	PUR	5.71	3	17.13
4			X	X	Terminal Block Cover	Cover for 12 position 300V DC Terminal Block	PUR	6.33	3	18.99
4			X	X	Terminal Block Buffer	Spacer for under 12 position 300V DC Terminal Block	PUR	2.70	3	8.10
4			X	X	Terminal Block Jumpers	Jumpers for 3/8" spaced Terminal Block	PUR	0.26	10	2.58
3			X		RFD900b	RFD900b 900 MHz Radio kit including antennae	PUR	114.50	1	114.50
3			X		RP-SMA Bulkhead Connector	RP-SMA Male to RP-SMA Female Bulkhead Connector	PUR	6.99	1	6.99
3			X		Power Supply	Power Supplies including batteries	SUB-ASSY			
4			X		LiPo Battery	HRB 4S 14.8 V 6000 mAh 50C LiPo Battery	PUR	114.99	2	229.98
4			X		Battery Connectors	Castle Creations Polarized Bullet Connectors	PUR	9.93	2	19.86
4			X		12 V Power Supply	MiniBox DCDC-USB-200 Power Supply with Enclosure	PUR	85.95	1	85.95
4			X		DC Jack Cable	MiniBox P4 to DC Jack Cable	PUR	4.95	1	4.95
4			X		Power Supply P4 Cable	MiniBox P4-ATX Cable	PUR	3.95	1	3.95
4			X		5 V 6 A Power Supply	BlueRobotics 5 V 6 A Power Supply	PUR	22.00	1	22.00
4			X		USB Hub	4-Port Powered USB-A Hub	PUR	38.62	1	38.62
3*			X		AHRS/IMU	Lord Microstrain 3DM-GX-5-45 GNSS/INS	PUR	2,895.00	1	2,895.00
3			X		AHRS/IMU	Lord Microstrain 3DM-GX-5-25 AHRS/IMU	PUR	1,495.00	1	1,495.00
3			X		GPS Receiver	BU-35584 GPS Receiver	PUR	31.22	1	31.22
3			X		R/G Receiver	Spectrum AB8010T Receiver	PUR	79.99	1	79.99
3			X		Motor Controller	BlueRobotics Basic Electronic Speed Controller	PUR	27.00	1	27.00
3			X		Arduino UNO	Arduino UNO R3 for Control Computer	PUR	26.21	1	26.21
3			X		Arduino Breakout Board	Adafruit Proto-Screwshield for Arduino R3	PUR	21.99	1	21.99
3			X		Circuit Standoffs	Standoffs for mounting circuit boards	PUR	12.88	1	12.88
3			X		Spade Terminal	Vinyl Spade Terminal for 16-14 Gauge	PUR	0.26	20	5.18
3			X		Control Computer	Intel NUC10i7FNH	PUR	824.00	1	824.00
3			X		USB Cables	Assorted USB Cables	PUR	7.00	4	28.00

Table C.2: Detailed Ground Station Bill of Materials.

BOM Level	1	2	3	4	Name	Description	Type	Unit Cost (\$)	Quantity	Sub Total (\$)
1	X				Ground Station	Equipment for Ground Station (1 per Network)	ASSY			
2		X			Laptop	Dell Vostro 7500 [Intel i7-1075H, 15Gb RAM]	PUR	1,500.00	1	1,500.00
2		X			R/C Transmitter	Spektrum DX8e 8-Channel DSMX Transmitter	PUR	259.99	1	259.99
2		X			Wireless Router	NETGEAR Nighthawk WiFi Router R7000P	PUR	159.97	1	159.97
2		X			Radio Transmitter	RFD900x 900 MHz Radio Transceiver Kit	PUR	114.50	1	114.50
2		X			Portable Power Station	EnginStar 300 W Portable Power Station [80000mAh]	PUR	259.99	1	259.99