PROCEDURAL OPTIMIZATION AND MEASUREMENT OF PASSIVE ACOUSTIC SENSOR NETWORKS FOR ANIMAL OBSERVATION IN MARINE ENVIRONMENTS

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE UNIVERSITY OF HAWAIʻI AT MĀNOA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

AUGUST 2016

By

Gregory L P Burgess

Thesis Committee:

Kevin Weng
Martin Pedersen
Philip M Johnson, Chairperson

Keywords: Acoustic Tracking, Acoustic Network, Animal Telemetry, Network Design, Network Metrics, Network Optimization

To Doctors Kevin Weng and Martin Pedersen for adopting a stray grad student,

Professor Johnson for infinite patience,

Steve for endlessly pointing out bugs,

and Renee for countless evenings at Starbucks.

ACKNOWLEDGMENTS

Thank you to Weng Lab for material funding and The Joint Institute of Marine and Atmospheric Research for technical support.

ABSTRACT

Static Observation Networks (SONs) are often used in the biological sciences to study animal migration and habitat. These networks are comprised of self-contained, stationary sensors that continuously listen for acoustic transmissions released by sonic tags carried by individual animals. The transmissions released by these tags carry serial identification numbers that can be used to verify that a particular individual was near a given sensor. Because sensors in these networks are stationary, sensor placement is critical to maximizing data recovery. Currently, no open-source automated mechanism exists to facilitate the design of optimal sensor networks. SON design is often governed by loose "rules of thumb" and "by eye" readings of low resolution bathymetric maps. Moreover, there is no standardized method for evaluating the efficacy of a SON. In this paper, we present a system which takes advantage of high-resolution bathymetric data and advanced animal modeling to provide optimal network designs. Our system also allows for statistical analysis of existing network configurations in order to create efficacy-metrics that can be used to evaluate arbitrary network configurations. In this paper, we discuss the mathematical and conceptual models used within this system and analyze the computational complexities of our approach.

TABLE OF CONTENTS

Acknowledgments . . . . . **iv**

Abstract . . . . . **v**

List of Tables . . . . . **x**

List of Figures . . . . . **xi**

1 Introduction **1**

1.0.1 Sensor Assembly 1

1.1 Static Acoustic Observation Networks 1

1.1.1 Sensor Deployment & Recovery 2

1.1.2 Tag Deployment 2

1.1.3 Comparison of Technologies 3

1.1.4 Advantages of Acoustic Networks 3

1.2 The Cost of Data 4

1.2.1 Cost of Alternative Technologies 4

1.2.2 Operating Costs 4

1.2.3 Cost Efficiency 5

1.3 State of the Art 5

1.3.1 Rules of Thumb for Sensor Placement 5

1.3.2 Metrics 5

1.3.3 Data Quality 6

2 Related Work **9**

2.1 Acoustic Array Design 9

2.1.1 Heupel et al - Automated acoustic tracking of aquatic animals: scales, design and deployment of listening station arrays . . . 9

2.1.2 Steel et al - Performance of an ultrasonic telemetry positioning system under varied environmental conditions . . . 9

2.1.3 Kessel et al - Close proximity detection interference with acoustic telemetry: the importance of considering tag power output in low ambient noise environments . . . 10

2.2 Sensor Placement Algorithms . . . 11

2.2.1 Howard et al - Mobile Sensor Network Deployment using Potential Fields Potential Field Algorithm . . . 11

2.2.2 Poduri et al Constrained Coverage for Mobile Sensor Networks Constrained Coverage (K-Neighbor Networks vs Maximum Coverage) . . . 12

2.2.3 Akbarzadeh et al - Probabilistic Sensing Model for Sensor Placement Optimization Based Signal Simulation and Attenuation (Omni Directional Sensors) 12

2.2.4 Yuan et al - Fast Sensor Placement Algorithms for Fusion-based Target Detection . . . 13

2.3 The Economic Value of Information . . . 14

2.3.1 Hansen & Jones - The value of Information in Fishery Management . . . 14

3 Design . . . 16

3.1 Program Requirements . . . 16

3.1.1 Motivation . . . 16

3.1.2 Supported Workflows . . . 17

3.2 Conceptual Model . . . 17

3.2.1 Time/Space Modeling . . . 17

3.2.2 Bathymetric Modeling . . . 18

3.2.3 Animal Modeling . . . 21

3.2.4 Receiver Modeling . . . 23

3.3 Evaluation of Receiver Emplacements . . . 24
3.3.1 Goodness Grid . . . 25
3.3.2 Evaluation Algorithms (Bias) . . . 25
3.3.3 Line of Sight Evaluation . . . 27
3.3.4 Selection of Optimal Emplacements . . . 27
3.4 Suppression . . . 28
3.4.1 Suppression Area . . . 28
3.4.2 Suppression Algorithms . . . 29
3.5 Optimal Sensor Projection . . . 31
3.6 Time and Space Complexity . . . 31
3.6.1 Bathymetry Grid . . . 31
3.6.2 Behavior Grid . . . 31
3.6.3 Line of Sight Computation . . . 32
3.6.4 Goodness Grid . . . 32
3.6.5 Optimal Receiver Placement . . . 34
3.6.6 Suppression . . . 34
4 Implementation . . . 37
4.0.7 Parameter Dictionary . . . 37
4.0.8 Sub-Functions . . . 37
4.0.9 Dispatcher Functions . . . 37
4.0.10 Major Modules . . . 38
4.0.11 Implementation Hurdles . . . 43
5 Results . . . 46

5.1 Output Files 46
5.1.1 Grid Graphs 46
5.1.2 Data Recovery Graphs 46
5.1.3 Text Files 46
5.2 Distinguishing Characteristics 48
5.2.1 Availability 48
5.2.2 Adaptability 49
5.2.3 Drawbacks 50
5.3 Future Work 52
5.3.1 Emperical Analysis 52
5.3.2 Heuristic Algorithms 53
5.3.3 Interactive Visualizations 53
6 Conclusions 55
A Table of Notations 56
Bibliography 59

LIST OF TABLES

1.1 Cost Summary of Alternative Technologies 4

1.2 Lifespan & Total Expected Transmissions 5

1.3 Price per Transmissions 5

2.1 Study Site Characteristics 11

2.2 Cumberland Sound Range Test Data 11

4.1 A summary of the key-value pairs in the `result` dictionary. 39

4.2 A summary of the `params` key-value pairs used in Bathymetry Parsing module. . . 39

4.3 A summary of the `params` key-value pairs used in Animal Modeling module.
* optional parameters. 40

4.4 A summary of the `params` key-value pairs used in the Suppression module.
* optional parameters. 41

LIST OF FIGURES

1.1 An illustration of two acoustic emplacements. The left rig consists of a float (orange circle) capable of lifting the rig without the ballast, a long rope, an acoustic receiver (green cylinder) and an acoustic quick release (red) attached to a strong ballast (purple). The ballast is sufficiently heavy to keep the entire rig from drifting with the current. Upon signaling the quick release, the ballast is released, and the float carries the rig (receiver, line, and quick release) to the surface. This type of rig is suitable for deeper deployments, or areas where no solid structure exists (coral, rock formation). The right rig consists of an emplacement for shallower waters or near solid structures (coral, rock formation). Here, a steel rod is driven into a solid structure, cemented into place, and an acoustic receiver is attached. 2

2.1 An illustration of how ray tracing is used within a 3D environment to identify potential visual obstructions. Ray tracing is used to determine which cells potentially block the line of sight between the receiver-containing cell p and the target cell q. Shaded cells must be evaluated by a Line of Sight algorithm to determine which portion of the target area in q is visible from p. [1] 13

3.1 (a) illustrates the bathymetric shadowing model for two adjacent cells within a bathymetric grid. (b) shows how artifically increasing the resolution of the bathymetric grid from (a) using the duplication method of cell sub-division does not affect the bathymetric shadowing model but increases the number of cells to consider. (c) shows how artificially increasing the resolution using a smoothing function increased the number of cells to consider and can lead to inflated signal reception. 20

3.2 An example of a distribution given by the Ornstein-Uhlenbeck Model with a high attraction value in the x-direction, a low attraction value in the y-direction, and a correlation value of 0.7. 22

3.3 An illustration of how ray tracing and integration over a shape function can be used in Option 2 & Option 3 to compute the probability of detecting fish within a cell. Dotted lines indicate the maximum and minimum depths visible to the receiver. The normal distributions in green/red indicate the distribution of fish within a given cell as determined by a shape function. The green portion of the curve indicates the observable section of the distribution, while the red indicates the unobservable section. The observable section (green) is computed by integration over the shape function. 26

3.4 An illustration of how the critical slope, and D_{max} are computed between cells p and q. The slope between the receiver and all interveening cells are computed (blue and red arrows). Then, the greatest slope is selected as m_{crit} (dark red arrow) because it poses the greates obstruction to vision of cell q. Finally, m_{crit} is projected into cell q (the light red arrow) to determine D_{max}. 28

3.5 An illustration of the Exact Suppression Algorithm. In (a), we see a bathymetry grid with infinitely high walls (the white 'h' shape) on an otherwise flat plane (blue region). In (b), we see Behavior Grid with a distribution (given by the OU movement model) of animals around the walls. (c) shows the computed Goodness of Sensor (receiver) locations within the study site. The program first identifies location 1 (the blue circle) as having the highest unique data recovery rate, and places a receiver there. The dotted lines represent the receiver's Detection Range. In (d), the program suppresses the Behavior Grid by subtracting the ERT of each cell within Suppression Range. Here the Suppression Range Factor is 1.0, so the Suppression Area is the same as the Detection Area. In (e), the Goodness Grid is recalculated, taking into account the suppressed Behavior Grid. Additionally, the program identifies location 2 as having the highest Unique Data Recovery Rate. In (f), the Behavior Grid is again suppressed to account for the placement of receiver (2). 29

3.6 An illustration of how ray tracing is used within a 3D environment to identify potential visual obstructions. Ray tracing is used to determine which cells potentially block the line of sight between the receiver-containing cell p, and the target cell q. Shaded cells must be evaluated by the Line of Sight algorithm to determine which portion of the water column in q that is visible from p. [1] 33

4.1 A screenshot of the webapp GUI. On the right is a Google Map widget that assists users in choosing lattitude and longitude boundaries for their study site (defining their Bathymetry Grid). On the left is a list of simulation options implemented as drop-down lists, and simulation parameters implemetned as text boxes. At the bottom (hidden) is a "Sumbit" button that sends that validates the parameters and sends them to the server. All parameters fields and lists display "tool tip" style descriptions of the parameter that field/list represents. 45

5.1 A graphical representation of the cumulative and per-receiver UDRR. User-placed receivers are representd by a grey line, system-placed receivers are represented by the black line, and projected receivers are represented by the green line. 47

5.2 A graphical representation of an artificial Bathymetry file at a 1m resolution (each cell has an edge length of 1m). Darker cells represent greater depths, while white cells represent inaccessible terrain (dry land). The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines. 48

5.3 The Behavior Grid represented as a heat map Higher levels of animal residency correspond with pink cells, moderate levels as light blue, and white for non-residency (inhospitable habitat such as dry land). The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines. 49

5.4 The Goodness Grid represented as a heat map showing the sum total of Estimated Receivable Transmissions (ERT) for a receiver placed in a particular cell. The legend in the top right assigns color coding to various ERT values. The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines. 50

5.5 The Coverage Grid represented as a heatmap showing the quantity of Estimated Receivable Transmissions from each cell in the study site, for the designated receiver array configuration. The legend in the top right assigns color coding to various ERT values. The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines. The missing corner of of Receiver 4's Detection Area is due to the presence of an obscuring section of Bathymetry (dry land). 51

5.6 A first-person view of the 3D space modeled in the program. Users would be able to "swim" through the environment and gain a better perspective on what they are modelling, what their receiver array might look like in the environment, and where more receivers might be needed. 54

CHAPTER 1
INTRODUCTION

Static Acoustic Observation Networks (SAONs) are often used in the biological sciences to study aquatic animal migration and habitat. These networks are comprised of self-contained, stationary receivers (hydrophones) that continuously listen for sonic transmissions released by acoustic tags carried by individual animals. The transmissions released by these tags carry serial identification numbers that can be used to verify that a particular individual was within detection range of a specific receiver at a given time. Acoustic networks are relatively inexpensive (compared to GPS/VHF Radio/Satellite tags). The primary goal of any tracking study is to obtain a high number of high quality data points (relating individual animals to space and time) to gain insight into animal behavior. Acoustic tags provide a way to generate a large volume of data points at low cost. However, unless these data points are captured by acoustic receivers, the cost efficiency of the SAON is lost. Within SAONs, data capture rates are highly dependent upon the chosen locations for receivers within the study area. The malplacement of receivers (in locations that interfere with the reception of data or where no tagged individuals are present) leads to low data returns, wasted resources, and diminished cost-efficiency. We present an application [3] that takes advantage of high resolution bathymetry, flexible behavioral modeling, and simplified acoustic propagation models to maximize the data recovery of a SAON. Our application provides a reproducible, customizable, and distributable method for generating optimal receiver placements and analytical network metrics.

1.0.1 Sensor Assembly

1.1 Static Acoustic Observation Networks

SAONs are composed of stationary rigs that are responsible for maintaining the chosen physical location for a receiver. Because positional data is interpolated from the position of nearby receivers, it is important that receivers are deployed accurately and maintain their position throughout the entire experiment[7]. This is best accomplished by attaching receivers to permanent emplacements (such as a rigid metal frame driven into a rocky substrate) that will resist substantial amounts of force (such as strong currents and curious animals). However, when it is not possible to create such permanent emplacements (perhaps due to regulation or extreme depth), more creative approaches are called for. A popular rigging consists of an acoustic receiver attached to a length of wire/rope with a strong float on one end connected to a strong ballast by an acoustic quick release (a device that disconnects a shackle when sent an acoustic signal) [7]. Such a rig can be dropped in the ocean and allowed to sink to its desired location. Obviously, various situations will require different rig designs and may contribute significantly to network costs (acoustic releases cost $2700 per piece).

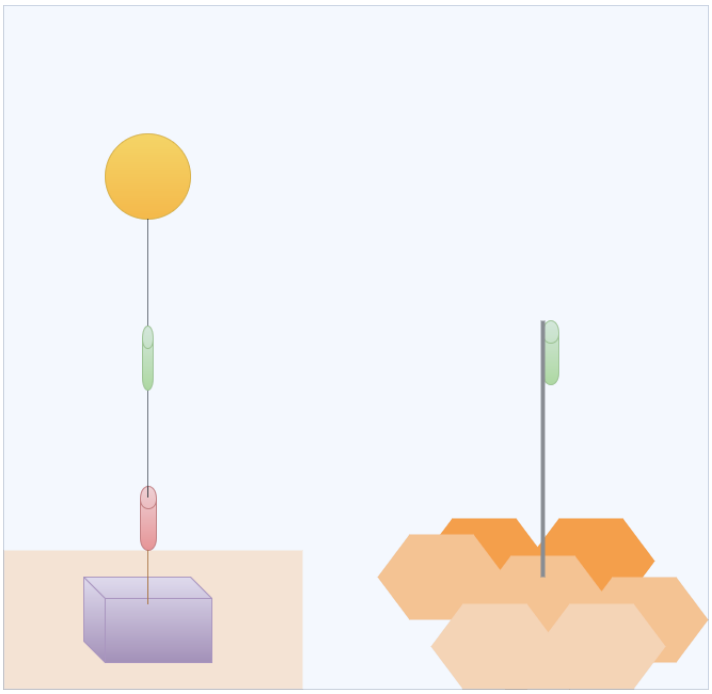

Figure 1.1: An illustration of two acoustic emplacements. The left rig consists of a float (orange circle) capable of lifting the rig without the ballast, a long rope, an acoustic receiver (green cylinder) and an acoustic quick release (red) attached to a strong ballast (purple). The ballast is sufficiently heavy to keep the entire rig from drifting with the current. Upon signaling the quick release, the ballast is released, and the float carries the rig (receiver, line, and quick release) to the surface. This type of rig is suitable for deeper deployments, or areas where no solid structure exists (coral, rock formation). The right rig consists of an emplacement for shallower waters or near solid structures (coral, rock formation). Here, a steel rod is driven into a solid structure, cemented into place, and an acoustic receiver is attached.

1.1.1 Sensor Deployment & Recovery

The labor required for receiver deployment and recovery depends on the design of the receiver assembly. Creating a permanent, rigid emplacement for a receiver can require multiple divers, special equipment, and hours of underwater elbow grease. The subsequent recovery of a permanent, rigid emplacement will most likely require a diver to physically remove the receiver from its emplacement. Deployment of ballast/float assemblies can be as simple as dropping the assembly overboard. Recovering a ballast/float assembly simply requires signaling the acoustic release with a hydrophone and allowing the buoy to carry the receiver assembly to the surface.

1.1.2 Tag Deployment

All telemetry technologies eventually require interaction with the individual to be tracked. This interaction consists of the physical deposition/implantation of tags on/into the individuals to be

tracked; a challenging and time consuming task. In aquatic tracking, this process is complicated by the potential sensitivity of a species to temperature, pressure, light, and sound. Aquatic animals must be captured, tagged, and released very quickly to avoid over-stressing/suffocating an animal. The improper handling, over handling, or improper release of an animal can result in its death and the loss of substantial time and resources.

1.1.3 Comparison of Technologies

1.1.3.1 Very High Frequency Radio

Very High Frequency radio (VHF) tracking involves attaching a VHF transmitter to an animal, and then using a VHF antenna and receiver to acquire transmissions. VHF transmissions have effective ranges on the order of tens of kilometers. Transmissions from VHS devices do not generally contain positional data, but instead serve as a means to estimate the distance and direction of a VHS device. Positional data is derived by noting the direction and strength of a signal from several different observational positions and estimating the transmitter's position by triangulation[5]. In an aquatic setting, VHF observation is generally performed from a plane or boat[19].

1.1.3.1.0.1 Satellite/GPS Satellite and GPS tracking are distinct but related technologies that rely on a network of satellites (either ARGOS or GPS, respectively) to compute the positional data of a tag. GPS tags rely on the GPS network of satellites to triangulate a tag's three dimensional position. GPS telemetry may be stored on-board a tag (requiring later retrieval) or transmitted via satellite to a remote server[5]. Satellite tags operate by transmitting messages to the ARGOS satellite system, which computes a tag's position by observing the Doppler effect on a tag's transmission[16]. Because the telemetry from satellite tags is transmitted back to remote servers, data recovery is automatic. Both technologies have fairly poor penetration into the ocean; therefore, Satellite/GPS transmissions generally occur only when an animal is near the surface of the ocean. This can lead to data sets with large spatial/temporal gaps between detections. Additionally, neither technology is desirable for observing animals that reside at significant depths. Due to the high cost of Satellite/GPS technology, studies using this technology generally have very small sample sizes.

1.1.4 Advantages of Acoustic Networks

After initial deployment, SAONs require no maintenance and incur no operating costs (unlike satellite and VHF technologies). This means that SAONs can operate around the clock and in conditions that would otherwise make it unsafe/impossible for field researchers to track animals (e.g. in a storm)[7]. However, it is necessary to retrieve the acoustic receivers at the end of the study in order to recover data[7]. Finally, SAONs allow for passive animal monitoring, removing the

potential disruption of natural behavior caused by active tracking (e.g. aircraft/boat noise/shadow scaring animals)[7]. SAONs also function at greater depths than satellite/VHF-based systems. Because the reception of acoustic transmissions (by acoustic receivers) occurs at the resident depth of the target species, an acoustic tag's transmission need not reach the surface to be detected (unlike Satellite and GPS based systems).

1.2 The Cost of Data

1.2.1 Cost of Alternative Technologies

SAONs are relatively cheap, with acoustic receivers costing $1300, and acoustic tags costing $330 each. Moorings for acoustic receivers can be significantly more expensive, with acoustic releases costing $2700. However, these costs are still significantly more affordable than satellite-based tags and collars, which cost upwards of $5000 each [17]. Additionally, recurring service fees and per-transmission charges may apply to data transferred over the satellite network. VHS radio tags are a seemingly cheaper alternative at $223 per unit, but require active monitoring to obtain each data point. The cost of paying for vehicles(boats/planes) and crews to periodically collect telemetry from these tags will significantly outweigh any initial cost savings.

Technology	Tag Cost	Receiver Cost	Operating Cost
VHF Radio Tag	$223[9]	$2940[2]	Agents & Transport
Satellite Tag	$3000-$5000[17]	$0	Service fees
Acoustic Network	$330	$1300	$0

Table 1.1: Cost Summary of Alternative Technologies

1.2.2 Operating Costs

While SAONs require no maintenance to operate, both Satellite and VHF based systems can incur operating costs while deployed. Satellite tags will require very little maintenance (precluding animal mortality), but satellite network operators may charge for access to, and transmission over their network[17]. VHF systems require little maintenance, but do require active field work in order to obtain positional information. Because a tag's location is interpolated from observations of its VHS signal from multiple locations, it is necessary for a field agent to routinely collect these observations to obtain telemetry data points. VHF networks have perhaps the highest operating cost, requiring a salary for one or more field agent(s) and transportation costs (renting a boat/plane). The operating costs for each technology should be included in the total cost of data collection, and subsequently the cost effectiveness of each solution.

1.2.3 Cost Efficiency

Technology	Tag Model	Transmit Period	Expected Lifespan	Expected Transmissions
VHF Radio Tag	Telonics FIS-040	1s	0.7 days	60,480
Satellite Tag	Telonics ST-18	60s	117 days	168,480
Acoustic Tag	Vemco VR-13	90s	1,135 days	1,089,600

Table 1.2: Lifespan & Total Expected Transmissions

Technology	Tag Model	Tag Price	Expected Transmissions	Price Per 1000 Transmission
VHF Radio Tag	Telonics FIS-040	$199	60,480	$3.29
Satellite Tag	Telonics ST-18	$3000	168,480	$1.78
Acoustic Tag	Vemco VR-13	$330	1,089,600	$3.03

Table 1.3: Price per Transmissions

Table 1.3 shows that the acoustic tags can generate data at significantly lower cost (Price Per Transmission) than alternative technologies. However, maintaining this price advantage requires the careful placement of the significantly more expensive acoustic receivers/emplacements.

1.3 State of the Art

1.3.1 Rules of Thumb for Sensor Placement

While animal tracking studies draw upon hard data to draw conclusions, the methodology for collecting this data is based upon loose rules of thumb driven by anecdotal evidence. Heupel et al's highly cited 2006 paper distills their prior experience with animal tracking into "rules of thumb" for designing a SAON. These rules point out issues such as avoiding areas of high noise, bathymetric obstruction, and acoustic echoing [7]. While Heupel et al's publication gives sound advice on design issues that warrant consideration, the discussion of analytical methods for measuring these issues falls out of the publication's scope.

1.3.2 Metrics

1.3.2.1 Data Recovery Rate

The most common metric used in analyzing the success of animal tracking studies is the Data Recovery Rate (DRR): $(\frac{pings\_emitted}{pings\_recovered})$. While it may seem intuitive to understand data recovery rates as an indicator of the quality of the dataset, one must bear in mind the objective of the study. As illustrated in section 1.3.3, the objective of the study defines how useful a particular dataset is

in addressing a research question. Therefore, Data Recovery Rate should be treated simply as a measure of how complete a particular dataset is, and how strongly it can support a claim.

1.3.2.1.1 Unique & Absolute Data Recovery Rates When discussing recovery rates, the Absolute Data Recovery Rate (ADRR) and Unique Data Recovery Rate (UDRR) can give insight into the qualities of a particular network design. ADRR is computed as the total number of pings that were received by all receivers in the network, divided by the total number of emitted pings. This means, that data recovery rates greater than 100% are possible.

1.3.2.2 Network Sparsity

Network Sparsity (δ) is a unit-less measure of the observational qualities of an acoustic network. This metric is useful in quickly expressing the density and intent of an acoustic network. For a list of n receivers within a SAON r_1, r_2, r_3, ... r_n, let a_i $(1 \leq i \leq n)$ be the distance to receiver r_i's nearest neighbor. Then, a is the median over all a_i. d_r is given as the detection range of a receiver (Section 3.2.4.2). Network sparsity is defined as $\frac{a}{2d_r}$. [13]

A δ of 0 describes an array of receivers that are virtually stacked on top each other. A *delta* between 0 and 1 indicates that receivers are placed such that their detection ranges overlap (a smaller δ indicates more overlap). A δ of 1 signifies that receivers in the array are positioned such that their detection ranges are just touching but not overlapping. A δ greater than 1 indicates that the receivers are farther apart, and that there are gaps between receiver coverage areas. [13] With this definition, it becomes obvious that Network Sparsity is a positive indicator for data fusion (section 1.3.3.1), and data resolution(section 1.3.3.1).

1.3.2.3 Sample Size

Another important factor to consider is the number of tagged individuals within a dataset. A dataset for a single tagged individual, no matter how complete, will not offer very much support to any species-wide conclusions. At the same time, a dataset with a large number of individuals and very low data recovery rates may not provide enough individual telemetry to provide definitive evidence.

1.3.3 Data Quality

1.3.3.1 Data Resolution

Acoustic receivers like the VEMCO VR2 log detections of acoustic transmissions as a tuple of time, tag number, and transmission strength. The strength of the received transmission can be used to approximate the distance between the tag and receiver. Data from a single receiver has a fairly low certainty of the exact position (low resolution) of the transmitting tag because only

a single distance can be observed. If multiple receivers are in close enough proximity to receive the same transmission, the strength of the transmission observed from several different known, fixed positions can be used to triangulate a more precise position (higher resolution) [13][14] . This process of combining multiple observations into a more accurate observation, known as **Data Fusion**, is useful for increasing the resolution of tag positional data and allows for the tracking of fine movements within a three-dimensional space. Detecting a tag using multiple receiver requires that those receivers have overlapping detection ranges[20]. Assuming a fixed number of receivers, placing receivers closer together (in order to overlap detection ranges) reduces the actual coverage area of the array. Thus, the coverage area of the array is inversely correlated with the resolution of the array. Alternatively, purchasing more receivers will achieve a higher resolution, but increases the cost of the array.

1.3.3.2 Meaningful Data

High-resolution data, while desirable, is not always critical to the study. First, consider an array of receivers placed in a tight cluster.

If the target species were highly sedentary, and the receiver cluster was placed around the area where a large number of individuals were captured and tagged, then the study would very likely yield a high Data Recovery Rate, but that dataset would be of little use in determining the spatial distribution of that species, as the data would be limited to the small area in which the receiver cluster was placed and the animals were captured. On the other hand, this dataset would be highly useful in confirming the sedentary nature of the species and defining a small home range. Additionally, data from multiple receivers could be combined to provide high-resolution telemetry for the location of an animal over time (see section 1.3.3.1) [7]. This high-resolution telemetry could be used to infer fine-scale co-location of two individuals, giving insight into social movement behaviors such as schooling or mating.

If the target species tended to roam over a large home range, then it is likely the cluster would receive only a few pings. In this case, little data will be gathered in regards to the extent of the specie's home range, but high resolution telemetry can be gathered for a short time if the animal passes through the cluster.

Next, consider an array of receivers spaced very far apart from each other over a very large spatial area. If the target species were highly sedentary, then tagged individuals might be observed by a very small number of receivers. If many individuals were tagged, then the dataset could describe the spatial distribution of the species over a large area.

If the target species tended to roam over a large home range, then it is likely the receiver array would pick up a tagged individual over a number of distant receivers. This data could be used to detect potential corridors for animal movement, establish individual home ranges [14], and to identify the spatial distribution of the species over a large area. While, the telemetry for individual

animals would be very low resolution, but the detection of many individuals by a single receiver could indicate areas of interest for future research.

While the above examples are brief and limited in scope, they highlight the importance in considering what kinds of data will be meaningful to specific research questions, and how to best deploy resources in order to collect the right kinds of data.

CHAPTER 2
RELATED WORK

2.1 Acoustic Array Design

2.1.1 Heupel et al - Automated acoustic tracking of aquatic animals: scales, design and deployment of listening station arrays

In their highly cited 2006 publication[7], Heupel et al discuss methods and issues related to the implementation of acoustic tracking network. They list the small size, low cost, and low maintenance requirements as primary drivers of the technology's popularity within the biological community. The authors point out that lower cost of acoustic instruments facilitates larger sample sizes and datasets. Additionally, because acoustic tracking is a passive process (researchers need not actively follow tagged animals to collect telemetry), data can be collected around the clock, while active tracking expeditions would be limited by inclement weather. Furthermore, because animals are not being shadowed by a noisy vessel, they are more likely to exhibit natural behavior.

The authors discuss the importance of identifying study goals and using those goals to drive array design. If high-resolution, positional accuracy is important, an array with high levels of overlapping receivers will allow for triangulation of a tag within 3D space. Animal residency within a specific area can be assessed with a *curtain* or *gate* (receivers placed in a pair of parallel lines/concentric circles across/around the area of interest) of receivers that will track animal ingress and egress. Presence/absence tracking and long-term survivorship can be can both be accomplished by a sparse (dispersed) array of receivers, as only occasional transmissions are necessary to acknowledge existence.

Heupel et al offer a plethora of practical advice for implementing acoustic arrays, with a focus on environmental phenomena that affect the propagation of acoustic signals. Environmental impedances to acoustic signal propagation include: background noise, sea floor composition, thermoclines and pycnoclines, salinity, and tidal flows. Other factors are less obvious, such as the positioning of the receiver with regards to other rigging elements (is any part of the rigging casting an acoustic shadow?), signal collision due to echoes and the presence of multiple tags, and the growth of fouling organisms on the receiver. The authors suggest that extensive field testing should be done prior to the commencement of any acoustic study.

2.1.2 Steel et al - Performance of an ultrasonic telemetry positioning system under varied environmental conditions

In 2014, Steel et al[18] investigated the accuracy of the VEMCO Positioning System (VPS), comparing its accuracy to GPS, and investigating possible sources of inaccuracy. VPS is a VEMCO

proprietary service that uses data gathered from multiple VEMCO acoustic receivers to triangulate "fix" (determine) the position of a tag in 3D space. To generate data, acoustic receivers were deloyed in various configurations in various study sites, acoustic tags were deployed as permanent emplacements (with known GPS coordinates) throughout each study site, and VPS fixes were compared against known GPS coordinates using Euclidian distance to find the Horiziontal Positional Error in meters (HPEm). The study also tracked Positional Efficiency, as the percentage of pings a receiver captured. The study was performed in river, estuary, and coastal locations. The following environmental variables were measured throughout the study: wind, wave period, wave height, water temperature, flow, turbidity, electrical conductivity of water, macrophyte growth rate, and discharge. The primary user-controlled variable was the array geometry. Generalized Linear Mixed Modeling showed that both Positional Efficiency and HPEm were most strongly correlated with position within the network. Specifically, tags in the center of the network had the smallest HPEm values, and the lowest positional efficiency. Tags placed in the middle of the network were observed by many more receivers than those on the outskirts of the array, which likely provided more references for positional fixing, and ultimately a more accurate VPS location. By the same account, tags in the middle of the array likely received acoustic interference from neighboring tags, which caused destructive interference in transmissions. Tags on the outskirts of the array had fewer neighbors, and likely less destructive interference. The authors concluded that array geometry was the most important predictor of positioning performance. They suggest that field testing both array geometries and environmental conditions is an important precursor to acoustic tracking studies.

2.1.3 Kessel et al - Close proximity detection interference with acoustic telemetry: the importance of considering tag power output in low ambient noise environments

In their 2015 publication, Kessel et al[10] discuss how acoustic transmission power affects acoustic reception. The team debunks a popular misconception that "higher tag power is better" by presenting evidence of Close Proximity Detection Interference (CPDI). While most researchers are concerned with the increasing the maximum detection range of their acoustic tags, they rarely ever consider the effect of increased transmission power at close range. To investigate the properties of CPDI, the authors conducted range testing in three distinct acoustic environments: (a)Cumberland sound, Baffin Island, Nunavut, Canada, (b)Lake Charlotte, Nova Scotia, Canada, and (c)Jupiter, Florida, USA. Range testing was done by deploying acoustic receivers (VEMCO VR2W 69kHz) and acoustic tags (V16-6H and V13-2H) at varying distances. Table 2.1 lists the physical characteristics of each study site.

The team observed that tags very close to a receiver had relatively low detection rates (Table 2.2). The team notes a "doughnut" shaped zone of poor reception around a receiver. At Cumberland sound, the team noted a strong CPDI effect, likely due to the hard sea floor, and low

Location	Depth	Receiver Elevation	Sea floor composition
Lake Charlotte	40m	5m	hard seafloor
Cumberland sound	30m	3m	soft mud
Jupiter	20m	2m	1.5m of sand over hard reef

Table 2.1: Study Site Characteristics

Tag	Range	Detection %	Range	Detection %
V16-6H	55m	8.3 %	370m	88.8 %
V13-2H	55m	17.9 %	221m	88.4 %

Table 2.2: Cumberland Sound Range Test Data

wind/wave action. At Lake Charlotte, the team recorded more pings than were released, indicating that acoustic echoes were being recorded in addition to primary transmissions. They also noted a reduced number of transmissions during periods of very high wind (attributed to fewer echoes). At Jupiter, the team found the weakest CPDI effect, accrediting it to the sandy bottom, reef structure, and noisy (wind, wave, boating activity & marine fauna) environment. They concluded that at close range, transmissions echo off the surface and sea floor, interfering with reception of the primary transmission. At greater ranges, the strength of these echoes drop off, and the primary transmission becomes the dominant signal. The team noted that hard surfaces likely promoted the echoing of transmissions, while softer sediment helps to absorb them. They also point out that strong wind likely caused surface distortions, which reduced the potential for acoustic reflection. Finally, background noise (such as that of human water activity, wave action, and marine fauna) helped to reduce CPDI, but decreased the maximum detection range.

2.2 Sensor Placement Algorithms

2.2.1 Howard et al - Mobile Sensor Network Deployment using Potential Fields Potential Field Algorithm

In 2002, Howard et al observe the simulated performance of an algorithm for the autonomous dispersion of a mobile sensor network in an unknown environment. Their model described each sensor a point charge that repelled other sensors and walls. At each iteration of the simulation, sensors pushed and pulled against each other, resulting in a net movement vector for that iteration. The simulation was iterated until all sensor movement ceased. A small static force allowed for the cessation of endless fine-scale movements and termination the simulation. The team found that by letting this simulation "settle", an optimal or near-optimal coverage solution could be found.

2.2.2 Poduri et al Constrained Coverage for Mobile Sensor Networks Constrained Coverage (K-Neighbor Networks vs Maximum Coverage)

In their 2004 paper, Poduri et al[15] expand upon the concept presented by Howard-et-al [8], describing an algorithm for maximizing the sensor coverage of an enclosed space while maintaining the property that each sensor has at least k-neighbors. Just as in Howard et al's work, the team's approach utilized a force dispersion algorithm where each sensor represented a point force, pushing against other nodes to extend coverage. To satisfy the k-neighbor requirement, sensors exhibited a strong attraction towards other nodes that had fewer than k-neighbors. The simulation settled when the k-neighbor requirement was satisfied for all nodes.

With respect to the design of acoustic tracking networks, this approach is useful in designing fine-scale movement tracking where maintaining the ability to triangulate tag positions is key. However, this approach tends to require a large number of sensors relative to the size of the study area. Small sensor to space ratios (only a few sensors for a very large area) would likely find that the k-neighbor network, while providing high-resolution tracking, covered too small of a small area.

2.2.3 Akbarzadeh et al - Probabilistic Sensing Model for Sensor Placement Optimization Based Signal Simulation and Attenuation (Omni Directional Sensors)

In their 2013 paper, Akbarzadeh et al[1] discuss the optimization of sensor coverage using probabilistic detection and attenuation models. Within the context of a wireless sensor network composed of small devices with limited, directed sensing capabilities, they discuss finding a minimal number of sensor placements such that areas of interest are covered. In their model, sensors had a limited degree of vision, requiring optimization of both sensor placement and angle. Their definition of *coverage* followed a probabilistic model, where sensors were subject to attenuation due to distance and obstruction (due to physical objects and environmental factors). The authors claimed that the traditional, omni-directional, attenuation-free, "disc" model for sensor coverage led to highly inflated coverage values. They also contend that a 2D x/y model is unrealistic, and that sensor placements should fall into a 3D space, where sensors can be placed at varying heights to achieve better coverage. They give the example of video surveillance in an urban environment, where certain areas of interest require sensor (camera) coverage, are subject to obstruction, and attenuation (increasing distance from an object decreases it's visual clarity). In this model, they attempt to find optimal 3D placements of sensors and sensor angles to achieve the required coverage with a minimal number of sensors.

They also present a model for determining Line of Sight based on a gridded 3D system. Within this system, a 2D grid of elevations represents 3D space as solid rectangular prisms. The left panel of Figure 2.1 shows how ray tracing is used to determine which cells potentially block the line of sight between the sensor-containing cell p and the target cell q. In the right panel, those shaded

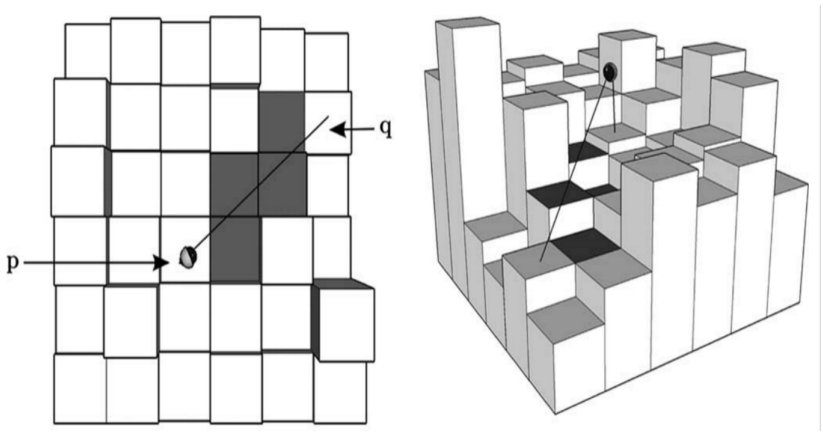

Figure 2.1: An illustration of how ray tracing is used within a 3D environment to identify potential visual obstructions. Ray tracing is used to determine which cells potentially block the line of sight between the receiver-containing cell p and the target cell q. Shaded cells must be evaluated by a Line of Sight algorithm to determine which portion of the target area in q is visible from p. [1]

cells must be evaluated to determine the portion of the target area in q is visible from p.

Our simulation is heavily based upon the probabilistic detection models presented by Akbarzadeh et al. We simplify the author's attenuation, obstruction, and probabilistic detection models to model our fixed-height, omni-directional acoustic receivers. Namely, we ignore the need to determine the optimal angle of receiver placement, and the need to find the optimal height of receiver placement. This vastly simplifies our computational complexity, allowing for faster simulation times, and iterative design.

2.2.4 Yuan et al - Fast Sensor Placement Algorithms for Fusion-based Target Detection

The 2008 Yuan et al publication discussed several approaches to maximizing sensor coverage of target areas using the fewest possible sensors. They defined coverage in terms of the probability of detection (P_d) and the probability of false positive detections (P_f). To be "covered", an area had to have sufficiently high P_d and sufficiently low P_f values. The team also proposed a probabilistic detection model, where multiple sensors observing the same target could combine their observations via data fusion to increase P_d and decrease P_f for that target. The model used called for the specification of some number of "target" zones to cover. The team relied heavily upon a Constrained Simulated Annealing (CSA) algorithm to solve the problem of optimal coverage. The team found that directly searching for a global optimal solution took exponential time (O(n!)). The team's second approach utilized a divide and conquer algorithm, which found local solutions for each target area individually, then combined local solutions into a single global solution. This approach took polynomial time, but resulted in an inefficient global solution. The authors proposed modifying

the divide and conquer algorithm to first choose locations that covered multiple spots, and then add sensors to solve any local coverage deficits. The final algorithm presented by the team combined a clustering algorithm and the proposed divide and conquer strategy. The clustering algorithm first grouped target locations into larger clusters, then solved for these clusters locally before combining them for a global solution. This approach both improved the runtime and reduced the total number of required sensors.

The goal of our framework is to recover the greatest number of unique acoustic transmissions given a fixed number of acoustic receivers. Yuan et al focus on minimizing the number of receivers required to achieve coverage over a given number of target areas. While their research is very closely related to the goal of our framework, the workflows are fundamentally different. We believe that SAON designers will more likely find themselves with a finite number of resources to collect as much data as possible (in line with our framework's goal), rather than attempting to achieve specific coverage for target areas. Still, the ability to address specific coverage requirements is valuable. Studies with little knowledge of a target species would likely use our framework, while studies with more refined understandings of their target species would want to utilize Yuan et al's workflow.

2.3 The Economic Value of Information

2.3.1 Hansen & Jones - The value of Information in Fishery Management

In their 2008 publication [6], Hansen & Jones address the value of information in ecological management through the lens of economic opportunity cost. In economics, opportunity cost describes the cost of taking a particular action as the value lost by not taking an alternative action. The authors argue that by investing too heavily in analysis, too few resources remain for responsive action. One assumption for their claim is the idea that there is a vast amount of uncertainty in ecology, and that fully realizing the complexities of an ecological system is impossible, no matter how much money is spent attempting to do so. They also argue that while collecting more information can reduce uncertainty, it is reasonable to assume that there are diminishing returns on the resulting actions (certainty is not linearly correlated with the benefit of the resulting action). Therefore, they argue that researchers should spend fewer resources on information gathering and more resources on ecological action. Given fixed research budgets, the opportunity cost of information gathering is then the ecological benefit of action.

The authors cite two case studies, the first involving the removal of juvenile parasitic sea lampreys from rivers feeding Great Lakes. Current treatment practices for the project begin by identifying which streams are most in need of treatment, chemically treating those streams, and counting the number of dead juveniles that flow downstream. The authors note that the cost of analysis (site identification and mortality determination) was nearly a third of the chemical treatment budget.

They argue that by taking an "adaptive" management approach (spending less on data collection and more on treatment), more streams could be treated, leading to a larger ecological impact, despite less accurate assessments.

The second case study was the designation of a global network of Marine Protected Areas (MPAs) sufficient to protect the biodiversity and sustainability of fisheries worldwide. The estimated cost for such a network is between 5 and 19 billion (USD). The authors claim a mindset exists that MPAs will be more effective if carefully chosen, and that no census exists on where MPAs should be placed. There is little research into correlation of costs for defining MPAs to the effectiveness of the resulting MPA network. It is believed that a multitude of designs could be "optimal", indicating that suboptimal configurations will result in minimal losses. The authors claim that all else being equal, spending less on MPA identification, and simply creating more MPAs will reduce the chances of failing to protect a critical location. Furthermore, the longer it takes to identify MPAs, the more the biodiversity at those sites degrades. The authors conclude that investments in information gathering should be carefully weighed against other management actions, with the goal of maximizing real-world impacts.

It is important to note that the authors neither excuse nor endorse action without data collection. Rather, they point out that focusing too many resources on information gathering diminishes ecological impact. Applying this rationale to our simulation, recreating a perfectly faithful representation of the real world is difficult if not impossible, as there are a great many environmental an physical phenomena that can affect acoustic transmissions. Modeling all of these phenomena would greatly increase the time and resources necessary to simulate them. It is important to keep in mind that in practice the framework will likely be used to run several times as researchers tinker with and refine their simulation parameters. Therefore, an overly complex simulation with a very long run time (hours, days, weeks) could cost many research hours. At the same time, over simplifying a simulation (to the point that it begins omitting significant phenomena) in order to reduce its runtime is a mistake. If these two "bad" simulations represent the extremes of simulation, then we contend that a "very good" simulation would consider phenomena that strongly affect the simulation results, and run in a reasonable amount of time.

CHAPTER 3
DESIGN

3.1 Program Requirements

3.1.1 Motivation

While detriments to SAON technologies are well documented [1] [7] [8] [10] [18], there few tools/services to analytically design SAONs around them. Furthermore, none of these tools/services are free and open-source.

3.1.1.1 Cost Efficiency

In Section 1.1, we discuss the costs of marine telemetry systems, noting that acoustic telemetry systems produce data at a significantly lower ($\geq$10x cheaper) cost than VHF or GPS/Satellite based technologies. In order to maintain the cost-efficiency of acoustic technology, a significant number of detections $\geq 10\%$ of the produced transmissions must be captured by the SAON's receiver array. Given the numerous (but avoidable) impediments to reception of these acoustic signals (1.3.1), the array-design process becomes critical to maintaining the cost-efficiency of SAON technologies. A free network design tool would help to maintain the cost-efficiency of SAONs by eliminating costs surrounding their design and evaluation, and increasing their data recovery rates.

3.1.1.2 Metrics

The computation of network metrics (Absolutes Recovery Rate, Unique Recovery Rate, Network Sparsity) is very labor intensive at large scale. Additionally, the process of computation may vary from experiment to experiment. An automated tool would solve both issues by providing a fast, repeatable, reliable, and well-documented method for computation. Metrics from such a tool would be useful in directly comparing different network designs.

3.1.1.3 Transparency

An open-sourced tool/service would make the design process more transparent, permitting peer-review and modification. This would provide increased confidence in the process, and increased adoption of the tool. Increased adoption would result in a larger number of efficient SAONs, leading to higher data recovery rates, better data quality, increased return-on-investment, and the ability to better address research questions.

3.1.2 Supported Workflows

3.1.2.1 Static Analysis

As mentioned in section 3.1.1.2, a primary motive for this tool was the ability to create a repeatable means of measuring the performance of a SAON. To this end, the ability to measure an existing network design is important. Users should be presented with network metrics after specifying bathymetry, receiver locations, network properties, and an animal model for a given study site.

3.1.2.2 Optimal Design

The primary motive for this tool is the ability to design optimal SAONs. Users should be presented with a network design (optimal receiver locations), and network metrics after specifying bathymetry, the number of receivers in the network, network properties, and an animal model for a given study site.

3.1.2.3 Optimal Addition

Similar to the problem of optimal design, is the problem of optimal addition: the augmentation of an already existing SAON. Users should be presented with a network design (optimally augmenting receiver locations), and network metrics after specifying bathymetry, the number of receivers to add to the network, network properties, existing receiver locations, and an animal model for a given study site.

3.2 Conceptual Model

3.2.1 Time/Space Modeling

3.2.1.1 Spatial Modeling

To model various attributes of a 3-dimensional underwater environment, we use a two-dimensional Grid of cells (in the x and y dimensions) containing numerical values. Numerical values in those cells, combined with user-defined values, can then be used in various shape functions to generate a third dimension (z) of values for that cell. In this way, we save significant amounts of memory by computing values for a specific three-dimensional cell as needed, instead of storing an additional dimension of values.

3.2.1.2 Temporal Modeling

Modeling a 3-dimensional environment over time is computationally intensive, as individual values can vary with time. The primary temporally-dependent phenomena we consider is that of animal movement. The animal model does not represent individual animal movement at a specific time,

but the summary cell-residency of all tagged individuals over the entire study period. As a result, we need not consider temporally-related phenomena.

3.2.2 Bathymetric Modeling

3.2.2.1 Bathymetric Grid

Bathymetry files are generally given as two-dimensional matrix of numerical values or a list of x,y,z values. Bathymetric files describe a three-dimensional space as a regular grid of rectangular prisms (cells) with constant length (x-dimension) and width (y-dimension), but varying negative (depth is negative) heights (z-dimension). The resolution of a bathymetric file is given by the x and y (length and width) dimensions of its cells. For example, a 50 meter Bathymetry file has cell sizes of approximately 50 meters square (although these cells are not necessarily perfectly square). Bathymetric files include meta-data listing the file's beginning and ending coordinates (North/South Latitudes and East/West Longitudes), the grid size (the number of rows and columns) in cells, and grid resolution (either in linear distance or degrees of latitude/longitude).

Our program works on a two-dimensional, grid-based system, taking advantage of the grid given by the user-provided bathymetric file. The Bathymetric Grid is a 2D Grid containing numerical values that describe a third dimension (depth). The exact spatial extent and resolution of this description are given by the bathymetric file. Thus, the resolution of our program's simulation and output are dictated by the resolution of the input bathymetric file.

We refer to a cell x (with row index i and column index j) on the Bathymetry Grid (A) as A_x or $A_{(i,j)}$.

3.2.2.2 Bathymetric Filetypes

Two highly popular file formats for Geographical Information Systems are provided by NetCDF and ArcGIS. NetCDF provides an open source file format that lists a header of meta-data and a white-space delimited matrix of numerical values. ArcGIS is a private institution that supplies many different file types, formats, and encodings for a family of GIS-related software systems. ArcGIS also supports the encoding and transcription of its proprietary formats to the NetCDF format. Due to the large number of possible format and encoding combinations in ArcGIS file formats and the ability to translate file these various formats to NetCDF, we natively support the NetCDF standard, and assume that users are capable of converting their data into the NetCDF format.

3.2.2.3 Bathymetric Resolution

Two key components of our program are the animal model and the bathymetric shadowing model. These models make decisions based upon the depth at a particular cell and the distance between

cells, data which is governed by the input bathymetry file. As stated in Section 3.2.2.1, the resolution of the program's output is dependent upon the resolution of the input bathymetry file. Obviously, higher resolution grids will offer higher resolution results; but, high-resolution bathymetric files tend to be difficult to come by. These files are often held by private agencies or simply never released to the public. It may then seem useful to artificially increase the resolution of the simulation by dividing the input file's bathymetric cells into sub-cells of finer resolution, but doing so increases the computational size of the program without meaningfully increasing the accuracy of the results.

When subdividing cells, either the sub-cells are given the same depth as their parent cell, or the depth of a sub-cell is interpolated from surrounding cells by some smoothing function. Subdividing a cell into sub-cells with the same depth makes the assumption that all sub-cells are actually the same depth. Furthermore, this results in the animal and bathymetric shadowing models making the same depth-based decision for all sub-cells that they would for the larger parent cell, needlessly increasing the computational load (see Figure 3.1b). Subdividing a cell into sub-cells with a depth governed by a smoothing function makes the assumption that there are no impeding obstacles between neighboring cells, and that there is a smooth transition between them (see figure 3.1c). Subdividing the two cells in Figure 3.1a half (ignoring for now the y component of our grid) results in the four sub-cells with depths given by a smoothing function (Figure 3.1c) would result in the large depth change at the sheer cliff face in Figure 3.1a being smoothed into smaller changes in depth, which would allow the unobstructed transmission of acoustic signals.

Both strategies (duplicating depth and applying a smoothing function) for artificially increasing the resolution of a bathymetric file disregard the manner in which the bathymetry was originally observed. Bathymetry is almost always computed as the average observed depth at several points within a geographic area of a given size (resolution). For example, imagine a particular cell in a bathymetric grid has a steep cliff running across the middle. Assuming that the sea floor at the top and base of the cliff were perfectly flat, and one measurement was taken at the top of the cliff and one at the bottom, the cell would have a depth equal to the average of the two observed depths. This average depth would then represent the depth for that entire cell, modeling it as a perfectly flat surface. Obviously this is problematic as the true nature of the sea floor is misrepresented. This misrepresentation leads to two conflicting arguments, the first is that the application of smoothing functions or duplicating depths of already averaged data makes faulty assumptions about real-world bathymetry. On the other hand, because source bathymetric files already represent aggregate data, one could argue that any conclusions drawn from the source bathymetry files are already faulty. We argue that the conclusions one can draw are only as good as the bathymetric information available. As the resolution of measured bathymetry (bathymetric measurements taken from the real world) increases (becomes finer), so too does the accuracy of the simulation. While artificially increasing the resolution of a source bathymetric file may skew results, it is sometimes useful for meshing two

bathymetric source files of varying resolution into one larger bathymetric dataset. Our program does not currently support modifying the resolution of source bathymetric files.

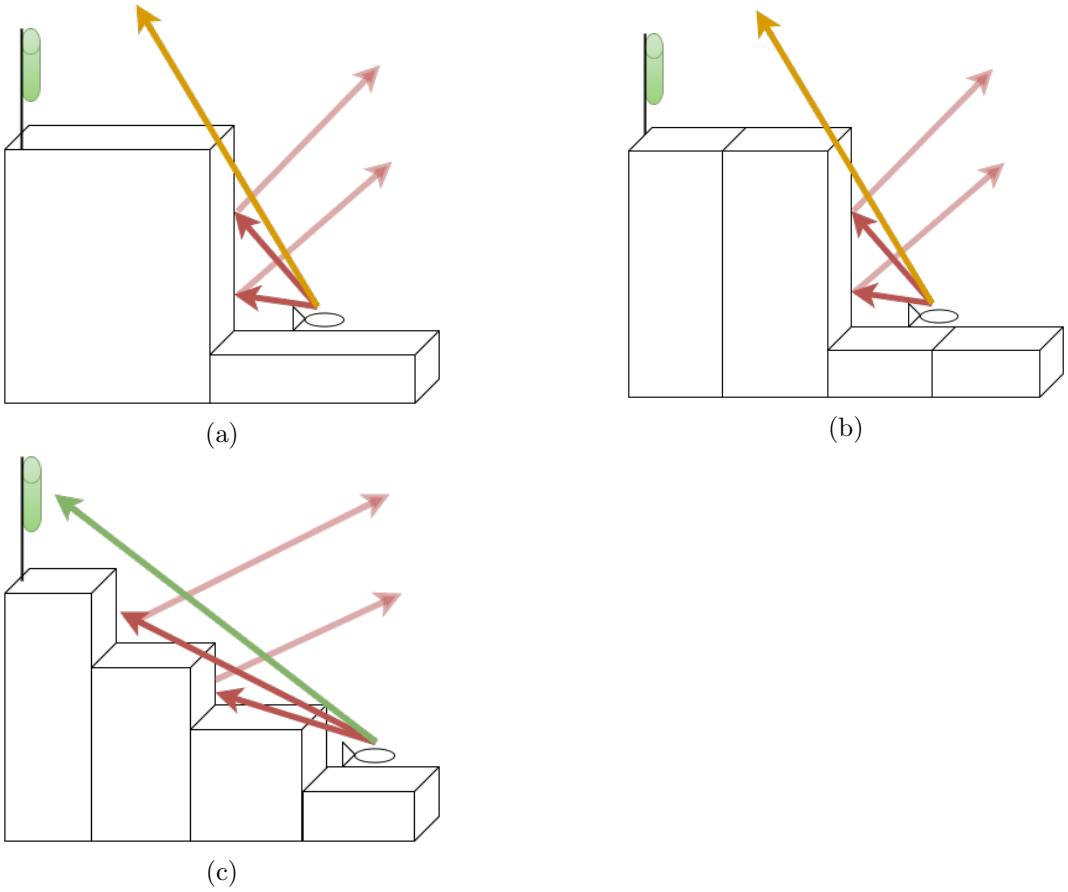

Figure 3.1: (a) illustrates the bathymetric shadowing model for two adjacent cells within a bathymetric grid. (b) shows how artifically increasing the resolution of the bathymetric grid from (a) using the duplication method of cell sub-division does not affect the bathymetric shadowing model but increases the number of cells to consider. (c) shows how artificially increasing the resolution using a smoothing function increased the number of cells to consider and can lead to inflated signal reception.

3.2.2.4 Bathymetric Shadowing

In the real world, the transmission of an acoustic ping originates from the tagged animal and propagates to the receiver. This propagation is governed by complex interactions with the surrounding environment (including the bottom substrate, water density, distance to the surface/sea-floor, thermoclines, the number of tagged animals in the vicinity, and ambient noise). Because it is difficult to model these phenomena without significant data on a large number of variables over a large area, our simulation uses a simplified propagation model relying on direct line of sight between a

tagged animal and a receiver. Simply put, in order to observe a transmission, there must exist a physically-unobstructed path between a tagged animal and a receiver.

3.2.3 Animal Modeling

3.2.3.1 Behavior Grid

Animals exhibit many different movement patterns and habitat preferences (both of which can vary in three-dimensional space). This greatly affects their distribution within the study space, and thus the network configuration that should be deployed to capture their movement. To describe the distribution of transmissions released from tagged animals, we utilize a two dimensional Grid with the same dimensions and resolution as the Bathymetry Grid (Section 3.2.2.1). Each cell in this "Behavior Grid" (B) contains a positive real value indicating, out of all transmissions that will be released over the entire study period, the percentage of transmissions that are expected to be released from within that cell's water column. This value can loosely be though of as the "animal-residency" of a cell. We refer to a cell x (with row index i and column index j) on the Behavior Grid (B) as B_x or $B_{(i,j)}$.

To populate this 2D grid with values, we provide two behavioral models to simulate the horizontal distribution of animals and two models for the vertical distribution. *Note*: we use the terms "animal" and "transmission" interchangeably as we are interested in capturing acoustic transmissions, which are given off by the tags carried by the target species.

3.2.3.2 Horizontal Movement Modeling

To simulate tagged animal movement across the horizontal x/y space (as one would expect to see on a map from above), we provide two basic probabilistic movement models: Random Walk, and Ornstein-Uhlenbeck(OU).

3.2.3.2.1 Random Walk Model The Random Walk model assumes that animals move randomly through the environment. As a result, over the entire study period, each valid grid cell (as defined by the Restricted Vertical Habitat Range (Section 3.2.3.3.2)) will see an equal amount of animal traffic. The result is that every valid cell in the grid will have the same chance of capturing an animal's acoustic transmission. We assume that tagged individuals will be willing and able to very briefly (in probabilistically negligible time frames) pass through inhospitable (over dry land, through impassible terrain) cells to get to other cells. This means that disjoint sections of habitat will contain an equal share of acoustic transmissions.

3.2.3.2.2 Ornstein-Uhlenbeck Model The Ornstein-Uhlenbeck(OU) model[11] supports the idea that over time, animals will tend to congregate near certain points of interest. This concept models an animal's desire to seek out and remain near a physically significant structure, a region

of high food availability, breeding grounds, shelter, etc. The OU algorithm allows for the modeling of the attraction towards a focal point in the x and y directions separately. Assuming a center point at the origin of a Cartesian grid, increasing the attraction in the x-direction will bring the distribution closer to the x-axis, and decreasing it will spread the distribution away from the x-axis. The same holds true for the attraction in the y-direction/y-axis. A correlation value ($-1 \leq x \leq 1$) allows for tilting the angle of the distribution. A positive correlation value tilts the distribution clockwise, and a negative correlation value tilts it counter clockwise. Correlations of 0 (no tilt), 1 (180° tilt), and -1 (-180° tilt) will have no observable effect on the distribution.

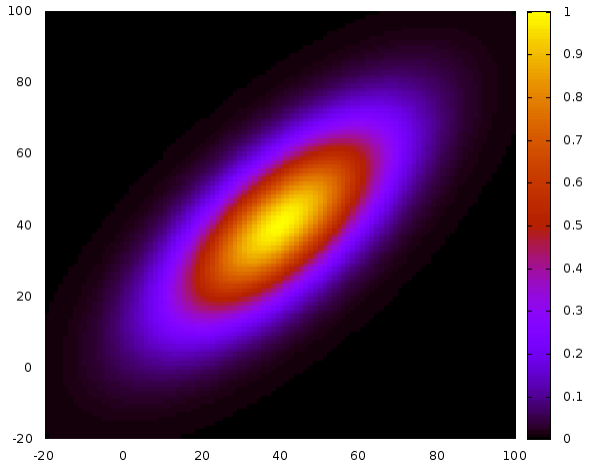

Figure 3.2: An example of a distribution given by the Ornstein-Uhlenbeck Model with a high attraction value in the x-direction, a low attraction value in the y-direction, and a correlation value of 0.7.

3.2.3.3 Vertical Movement Modeling

To model the z-axis movement behaviors of animals within the water column (as one would expect to see from the side of an aquarium), we provide two optional movement models: Habitat Preference, and Restricted Vertical Habitat Range. The first controls the distribution of animals within the water column, and the second prohibits animal residency within cells that fall outside specified depth ranges.

3.2.3.3.1 Habitat Preference Some animals exhibit the preference to reside within a specific section of the water column. For example, prey animals may prefer hiding in reef heads at the bottom of the water column, while predators will prefer to hover several meters off the bottom. This preference can be incorporated into the animal model by specifying mean ("Preferred Depth") and standard deviation("SD of Preferred Depth") values. These values are given as a measure of the distance (in meters) from the bottom. For example, specifying a depth of '0' for Preferred

Depth indicates that the animal prefers to live on the sea floor, while a value of '5' indicates that the animal prefers to live 5m off the sea floor. Allowing a standard deviation value allows for the modeling of animals that tend to be sedentary within the water column (a small deviation), and those that migrate through the water column (a large deviation). The simulation does not support the modeling of sub-surface animals, as any part of the distribution curve falling below the sea floor will be redistributed along the rest of the curve.

3.2.3.3.2 Restricted Vertical Habitat Range Some animals will live only in a specific depth range. For example, a deep sea fish may live only in depths of 300-400 meters. To incorporate this into the behavioral model, users can specify a minimum and maximum vertical habitat range for their animal. If this option is selected, the program will only simulate animals in cells whose maximum depths are between the given minimum and maximum depths. As in the Random Walk Model, we assume that animals are willing and able to move between disjoint areas of habitat.

3.2.4 Receiver Modeling

3.2.4.1 Acoustic Attenuation

In practice, the highest probability of observing an acoustic transmission occurs when a tag is several dozen meters away from the receiver (due to CPDI effects [10]), and dropping off as the tag moves further away from the receiver. We model the attenuation of acoustic signal as a deteriorating function following a Gaussian distribution. Given a particular distance between a tag and receiver, this distribution describes the probability of capturing the transmission. This distribution is defined by user-specified values for mean value, peak value (at the mean), and standard deviation. Within this framework, CPDI effects can be modeled by modifying the mean value. Herein, we denote the probability of detecting an unobstructed acoustic transmission from x meters away as $P_{Attenuation}(x)$.

3.2.4.2 Detection Range

The distance at which a transmission from an acoustic tag can be captured is largely determined by the transmission power of the acoustic tag. In our model, we assume that all animals have the same model of acoustic tag and that "transmission range" is instead a property of the receiver, referred to as "Detection Range" (D_{range}). We define the Detection Range of a receiver to be the average distance (specific to the given study site) at which the probability of detecting an acoustic transmission drops to 5%.

3.2.4.3 Detection Area

We define the Detection Area of a receiver to be the square plane of cells around a receiver which will be considered when evaluating a potential receiver emplacement. Theoretically, the Detection Area of a receiver is a circle with a radius equal to the receiver's Detection Range (D_{range}). Within our program, we model the Detection Area as a square Grid of cells with an edge length ($d_{detection}$) equal to $2^*D_{range} + 1$, centered on the cell where a receiver will theoretically be placed. The additional cell ensures odd dimensions for the square, so that there exists a central cell to model as the receiver's location. We model a square instead of a circle for simplicity of book keeping (we simply keep track of the $d_{detection}$, and the coordinates of the Grid's top left cell). The theoretical Detection Area can then be imagined as a circle circumscribed in the square of our square Detection Area. It might seem that this simplification would alter the probabilistic distribution models as additional "gray" cells (those cells outside the circle but within the square) are being considered. However, this is not the case as those models utilize normal functions based on the absolute distance from the receiver. Gray cells will therefore contribute exponentially less to the total probability than those cells within the Detection Range, and their effect on the final probability will be negligible.

3.2.4.4 Network Specification

There are three distinct ways to place receivers into the model: user specification, optimal placement, and optimal projection. User Specified Receivers (USR) represent receivers that are already deployed at the study site and are being integrated into a larger network. Program Placed Receivers (PPRs) are receivers that will be optimally placed by the program, taking into account existing (user specified) receivers placements using the suppression dynamic explained in section 3.4. Projected Receivers are PPRs that do not count towards the network statistic rates returned by the program. Instead, their contributed recovery rates are given separately, and represent the marginal benefits of placing incrementally more receivers.

3.3 Evaluation of Receiver Emplacements

Section 3.2.3 discusses the models used to simulate animal movement. These animal models populate the "Behavior Grid" (B), a 2D grid (of the same size as the specified Bathymetry Grid) which gives a distribution summary of transmissions across the study area, and a shape function which describes the distribution of animals within the water column at various depths. Section 3.1a discusses the Line of Sight (LoS) model used to simulate the obstruction of acoustic transmission. Section3.2.4 discusses how attenuation affects the propagation of acoustic signals and thus the probability of detecting acoustic transmissions from a particular distance ($P_{Attenuation}(x)$). Together, these models are used to evaluate receiver locations throughout the study site.

3.3.1 Goodness Grid

In modeling our 3D environment (the study site), we make extensive use of 2D grids with a third dimension. However, we assume all receivers in the study will have the same elevation off the sea floor. Recall that our program will evaluate every cell in the Bathymetry Grid as a potential receiver emplacement. Together with the assumption of fixed receiver elevation (of k meters), the search space considered by our program is a 2D surface parallel to the sea floor. Thus, it makes sense to store the metric for this search space in a 2D Grid with the same dimensions and resolution as the Bathymetry Grid. This Gird is referred to as the "Goodness Grid" (G). We refer to a cell x (with row index i and column index j) on the Goodness Grid (G) as G_x or $G_{(i,j)}$.

3.3.2 Evaluation Algorithms (Bias)

The Evaluation or Goodness algorithms define a process for evaluating the goodness of a particular cell as a receiver location. Given a particular cell at row i and column j, an Evaluation Algorithm will compute a non-negative, rational value representing the quantity of observed transmissions that a receiver placed k meters off the bottom of cell (i, j) would recover.

While users are able to write their own Evaluation Algorithms, three basic algorithms are provided. All Evaluation Algorithms compute the Goodness of a potential receiver location ($G_{i,j}$) by summing the number of Estimated Receivable Transmissions (ERT) from all cells within Detection Range of cell (i, j). Each Evaluation Algorithm has a particular bias in its approach to computing ERT. $ERT(x, T, a, b)$ denotes the ERT value for the cell (a,b), relative to the receiver in cell T, as determined by Evaluation Algorithm x. Note: we utilize the terms "bias" and "algorithm" interchangeably when referring to Evaluation Algorithms.

Explicitly, $G_{i,j} = \Sigma_{a=i_0}^{i_n} \Sigma_{b=j_0}^{j_n} ERT(x, (i, j), a, b)$, where x is the chosen Evaluation Algorithm/Bias, i_0 and j_0 represent the top-left row/column indexes (respectively) for the Detection Area, and i_n and j_n represent bottom-right row/column indexes (respectively) for the Detection Area.
Note: here we denote the receiver-containing cell T as (i, j).

3.3.2.1 Animal Only (Option 1)

This option prefers to place receivers in areas of high animal activity, completely oblivious to the surrounding bathymetry. This is useful for when no bathymetric information is available or transmissions are unlikely to be blocked by bathymetry (in a very flat area, or when animal activity occurs well above the sea floor). The "Animal Only" option computes ERT for a cell as the animal-residency of a cell (a, b)(according to the Behavior Grid), multiplied by the probability of detection due to attenuation for that cell's distance from the receiver cell(T).
Explicitly, $ERT(1, T, a, b) = B_{a,b} * P_{Attenuation}(Range(T, a, b))$

3.3.2.2 Visible Fish (Option 3)

This option chooses receiver locations that have the best view of areas with high animal residency. Both animal presence and visibility due to topography are considered. Figure 3.3 depicts this idea. The green and red normal curves represent the distribution of animals in the water-column. The green portion of the normal curve illustrates the portion of observable (due to bathymetry) animals.

This option is most useful when both well-documented animal behavior and high resolution bathymetry are available. ERT is computed as animal-residency (according to the Behavior Grid), multiplied by the percentage of observable transmissions (due to bathymetric obstruction), multiplied by the probability of detection due to attenuation.
Explicitly, $ERT(3, T, a, b) = B_{a,b} * P_{observation}(T, a, b) * P_{Attenuation}(Range(T, a, b))$

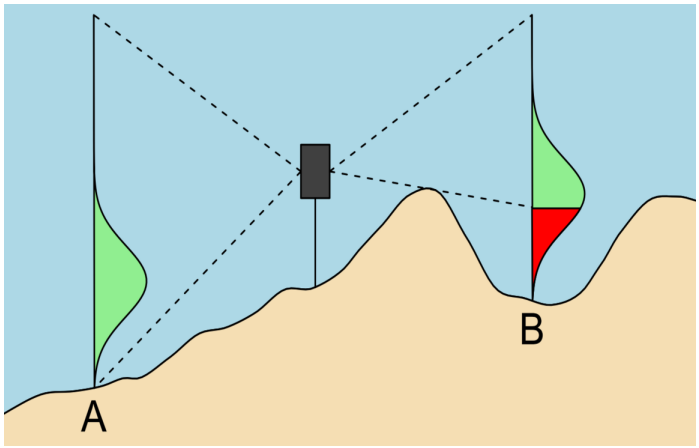

Figure 3.3: An illustration of how ray tracing and integration over a shape function can be used in Option 2 & Option 3 to compute the probability of detecting fish within a cell. Dotted lines indicate the maximum and minimum depths visible to the receiver. The normal distributions in green/red indicate the distribution of fish within a given cell as determined by a shape function. The green portion of the curve indicates the observable section of the distribution, while the red indicates the unobservable section. The observable section (green) is computed by integration over the shape function.

3.3.2.3 Topography Only (Option 2)

This option places receivers in areas that have the best visibility of the surrounding area, regardless of the expected animal-residency. This is useful for experiments where animal habitat is unknown or to be determined. Here, ERT is computed as the number of observable transmissions from a uniformly distributed animal model. While this algorithm is referred to as "Topography Only", it is implemented using the "Visible Fish/Option 3" algorithm with a simplified animal model. Specifically, we assume that animals are uniformly distributed throughout the environment. Because animals are uniformly distributed, the more of the water column a receiver can see, the more

transmissions it will receive. Thus, the algorithm will value cells with the best possible view of the surrounding water columns.
Explicitly, $ERT_{2,T,i,j} = B_{a,b} * P_{observation}(T, a, b) * P_{Attenuation}(Range(T, a, b))$
Note: With this option, B is initially described as a uniform distribution.

3.3.3 Line of Sight Evaluation

Section 3.2.2.4 explains the concept of Bathymetric Shadowing, which requires a bathymetrically unobstructed Line of Sight (LoS) to exist between an acoustic tag and a receiver in order for transmissions from that tag to be received. Section 3.3.2.2 discusses how the evaluation algorithms utilize this concept to compute the quantity of transmissions that are receivable (ERT). For this computation, the evaluation algorithm requires knowledge of the deepest depth ($D_{max}(p, q)$) in a cell q that is visible to a receiver in cell p. This depth is used by Evaluation Algorithms 2 and 3 to compute the finite integral for their ERT values.

To determine the D_{max} for a cell q, relative to a receiver in cell p, we must first determine which cells could potentially obscure LoS between p and q. We refer to a list of n such potentially obscuring cells as C. Figure 3.1a illustrates this process. The shaded cells in the left image potentially obstruct the vision from cell p to cell q. Recall that B_x refers to the depth of the cell x as given by the Bathymetry Grid.

Next, we determine the slope between the receiver at p and each cell in C. The slope between a receiver elevated k meters off the sea floor of cell p and v ($m_{p,v}$) is defined as the difference in elevation between $B_p + k$ and B_v divided by the Euclidean distance ($Edist_{p,v}$) between cell p and cell v. Explicitly, $m_{p,v} = \frac{(B_v - B_p + k)}{Edist_{p,v}}$. The greatest (largest signed value) slope $m_{p,v}$ for all cells v in C is the Critical Slope (m_{crit}). If we project a line with this slope from the receiver at p to our target cell q, the line would be tangent to the tallest obstruction along our LoS. Thus, the critical slope determines D_{max} for cell q. Explicitly, $D_{max} = m_{crit} * Edist_{p,v} + B_p + k$. Figure 3.4 gives a visual depiction of how m_{crit} is used to determine D_{max}.

3.3.4 Selection of Optimal Emplacements

Section 3.3 describes the evaluation of cells as potential receiver locations. The Optimal Design and Optimal Addition work flows (section 3.1.2) require the identification and selection of a user-specified number of optimal receiver emplacements. Once all cells within the area of interest have been evaluated, the program selects the user-specified number of optimal receiver locations, and then the user-specified number of projected receivers.

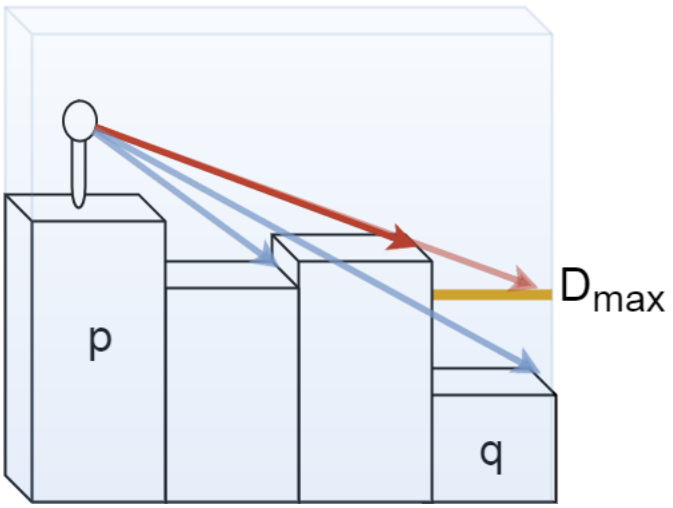

Figure 3.4: An illustration of how the critical slope, and D_{max} are computed between cells p and q. The slope between the receiver and all interveening cells are computed (blue and red arrows). Then, the greatest slope is selected as m_{crit} (dark red arrow) because it poses the greates obstruction to vision of cell q. Finally, m_{crit} is projected into cell q (the light red arrow) to determine D_{max}.

3.4 Suppression

As previously mentioned, the optimality of a network depends greatly upon the way in which Data Quality is defined. Some users will want to design a network that covers as much of a study area as possible, while others might want to heavily saturate a small area with receivers to facilitate higher resolution telemetry. Still others may wish to find the receiver locations that return the highest number of unique data points. Unique Data Recovery Rate (UDRR) and Absolute Data Recovery Rate (ADRR) [Section 1.3.2.1] also play a role in the definition of Data Quality. To allow for the control of sensor distribution, we provide the Suppression mechanic.

3.4.1 Suppression Area

As an input to the program, users can specify a Suppression Range Factor (f_{supp}) as a positive real number. This factor is used to scale the Detection Range (D_{range}) (Section 3.2.4.2) and define a Grid similar to the Detection Area (Section 3.2.4.3). The resulting Grid is referred to as the Suppression Area. The Suppression Area is centered over a receiver placement and cells within this area have their number of undetected transmissions reduced according to a Suppression Algorithm. The Suppression mechanic is used during the selection of receiver locations. After selecting each optimal receiver location (Section 3.3.4), the Suppression Area around that receiver emplacement is suppressed. As a result, receiver emplacement selection is a sequential process, where selecting an emplacement depends upon the suppression of the previous selection.

Assuming a uniform Goodness Grid, setting the Suppression Range Factor to 1.0 ensures that

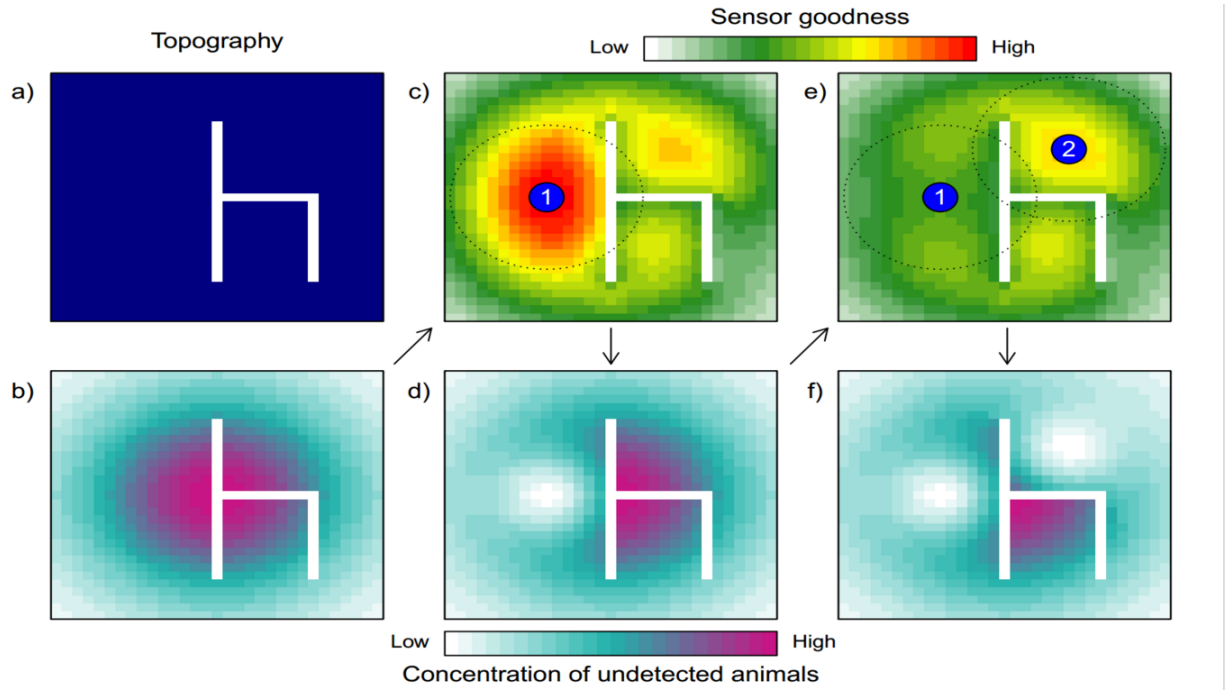

Figure 3.5: An illustration of the Exact Suppression Algorithm. In (a), we see a bathymetry grid with infinitely high walls (the white 'h' shape) on an otherwise flat plane (blue region). In (b), we see Behavior Grid with a distribution (given by the OU movement model) of animals around the walls. (c) shows the computed Goodness of Sensor (receiver) locations within the study site. The program first identifies location 1 (the blue circle) as having the highest unique data recovery rate, and places a receiver there. The dotted lines represent the receiver's Detection Range. In (d), the program suppresses the Behavior Grid by subtracting the ERT of each cell within Suppression Range. Here the Suppression Range Factor is 1.0, so the Suppression Area is the same as the Detection Area. In (e), the Goodness Grid is recalculated, taking into account the suppressed Behavior Grid. Additionally, the program identifies location 2 as having the highest Unique Data Recovery Rate. In (f), the Behavior Grid is again suppressed to account for the placement of receiver (2).

receiver Detection Areas do not overlap, maximizing network coverage. Setting the Suppression Range Factor to 0.5 means that receiver Detection Areas will exactly overlap (the edge of one Detection Area will touch a receiver). A Suppression Range Factor of 0.0 will cause receivers to be stacked on top of each other, over the highest rated cell in the Goodness Grid. Using this approach, it is possible to disperse sensors across physical space, while optimizing the number of captured transmissions.

3.4.2 Suppression Algorithms

To promote the flexibility of our program, we provide users three algorithms for suppression. The first offers a lower computation time at the cost of divergence from our theoretical model. The sec-

ond offers a higher fidelity model, but requires significantly more computation. The final algorithm provides very high fidelity, but at very high cost.

3.4.2.1 Static Suppression

The Static Suppression algorithm takes a "black-out" approach and replaces the goodness of all cells within the Suppression Area by a user-specified value.

3.4.2.2 Scalar Suppression

The Scaled Suppression algorithm reduces the number of undetected transmissions in the Suppression Area using a template. This algorithm requires a minimum and maximum suppression value ($Supp_{min}$ and $Supp_{max}$). The algorithm first creates the Suppression Template, a Grid with the same dimensions the Suppression Area. This Suppression Template is populated with a radial gradient of suppression values. The central cell in the Suppression Template receives the largest suppression value ($Supp_{max}$), and the cell farthest away from the center receive the smallest suppression value ($Supp_{min}$). Cells between the central cell and the furthest cell receive a suppression value negatively correlated with their distance from the central cell (according to the Suppression Function). To suppress a particular cell, the algorithm simply preforms a scalar multiplication (corresponding cells in the Suppression Template and Suppression Area are multiplied together). Because this computation is very simple, it runs very quickly and is therefore useful for simulations that intend to place a very large number of receivers. The cost of this computational simplicity is that the algorithm ignores line of sight model. For instance, a cell whose line of sight to the receiver is obstructed will be reduced by the same factor as a cell near the receiver with an unobstructed line of sight. Obviously this is not a faithful representation of the conceptual models discussed earlier, but supplied as a computationally-simple alternative.

3.4.2.3 Exact Suppression

The primary purpose of this algorithm was to discount transmissions which will be observed, according to an Evaluation Algorithm, by a placed receiver. This algorithm uses the ERT value given by the Evaluation Algorithms (the user-specified Evaluation Algorithm is used in both Goodness and Suppression calculations) to determine the number of transmissions within a particular cell to discount. $ERT_{T,i,j}$ denotes the ERT value in cell (i,j) observed from cell T. Exact Suppression works first by computing $ERT_{T,i,j}$ for all cells (i,j) within the Suppression Area. Then, each corresponding cell in the Behavior Grid, $B_{i,j}$, is reduced by $ERT_{T,i,j}$. Finally, the Goodness of all cells within a distance of (Detection Range plus Suppression Range) cells of a suppressed cell (all those that would have their Goodness affected by the suppression) is recalculated. Figure 3.5 illustrates the Exact suppression algorithm.

3.5 Optimal Sensor Projection

In normal research situations, users will have a set number of receivers to place within their study site. The process of arriving at this number is likely unscientific, perhaps relying on user estimation. Rather than guessing at the number of receivers to use, and hoping for an adequate data recovery rate, users should be able to calculate the marginal benefit (additional detections, increased Data Recovery Rates) of utilizing a variable number of receivers. To this end, the program returns the marginal gain in Data Recovery for a given number of additional receivers. Within the program, users to specify a number of receivers to project, and receive graphs and metrics of the marginal increase in Data Recovery Rate. With this data, users can determine an appropriate number of receivers to purchase or construct an argument for purchasing more receivers.

3.6 Time and Space Complexity

To evaluate the temporal and spatial complexities of various elements of our program, we define the following variable inputs to the program:

n	The square root of the number of cells in the Bathymetry Grid, (the edge length of a square grid). Also recall that the Bathymetry, Behavior, Goodness, and Coverage Grids are all of identical dimension.
D_{range}	The Detection Range of a receiver in the simulation..

3.6.1 Bathymetry Grid

The bathymetry data for the study site is represented as a Grid of n^2 cells. This data must be copied into local memory (RAM) from the input Bathymetry File, each cell in the grid takes O(1) time to copy, and O(1) space to hold. Thus, the creation of the Bathymetry Grid will take O(1) * n^2 = O(n^2) time and space.

3.6.2 Behavior Grid

Animal residency is computed as a function of the depth of a particular cell (Restricted Vertical Habitat), and the cell's location (OU/RW modeling). This computation takes a constant [O(1)] amount of time. The space required to store a single residency value for each of n^2 cells is also O(1). Therefore, the population of the Behavior Grid takes O(1) * n^2 = O(n^2) time and space.

3.6.3 Line of Sight Computation

As discussed in Section 3.3.3, the LoS algorithm is given as:
Step 1) Determine the m intervening cells $C_{0...m}$ between p and q
Step 2) Compute the slopes between p and C_i for all i in $[0...m]$
Step 3) Choose the Critical Slope
Step 4) Project a line from p to q along the Critical Slope to find D_{max}.

Step 1 finds intervening cells by ray tracing, which can be done in linear ($O(n)$) time. Temporarily storing this list of intervening cells requires $O(n)$ space. The number of cells considered (n), depends upon the distance between the receiver and the target cell[12]. Given that the Detection Area has a radius of D_{range}, and that receivers are located in the center of the Detection Area, the greatest distance between a receiver and a target cell in the Detection Area occurs between a receiver and the corner cells of the Detection Area (a distance of $\sqrt{2} * D_{range}$). Thus, the number of cells considered when creating a list of intervening cells is $O(\sqrt{2} * D_{range} = O(D_{range})$.

Step 2 computes the slopes for each of the m cells in C. Computing the slope between two points takes $O(1)$ time to compute and space to store. Therefore computing m slopes takes $O(1) * O(m) = O(m)$ time and space. As stated above, m is at worst $O(D_{range})$ on the Detection Area. Thus, the slope computation is at worst $O(D_{range})$, requiring $O(D_{range})$ extra storage.

Step 3 chooses the largest slope amongst all m slopes. Therefore, we must consider m items, each requiring $O(1)$ time to consider as the largest. Since, m is $O(D_{range})$ on the Detection Area, this step takes ($O(D_{range})$) time to compute, and $O(1)$ space to store.

Step 4 is a direct computation, requiring $O(1)$ time to compute and $O(1)$ space to store the result.

In total, the LoS computation time is $O(D_{range}) + O(D_{range}) + O(D_{range}) + O(1) = O(D_{range})$, requiring $O(D_{range}) + O(D_{range}) + O(1) + O(1) = O(D_{range})$ temporary storage space.

3.6.4 Goodness Grid

Recall that the population of the Goodness Grid is performed by Evaluation Algorithms (Section 3.3.2), which compute ERT values for each of the D^2_{range} cells in the Detection Area. These ERT values are then summed (requiring $O(D^2_{range})$ time) to provide the Goodness value for a single cell, out of the n^2 cells, in the Goodness Grid. Thus, if we define the computational complexity of the ERT computation of Evaluation Algorithm x to be E_x, the computational complexity for populating the entire Goodness Grid is the time for ERT computation across the D^2_{range} cells in the Detection Area plus the time for summing those D^2_{range} ERT values, for each of the n^2 cells in the Goodness

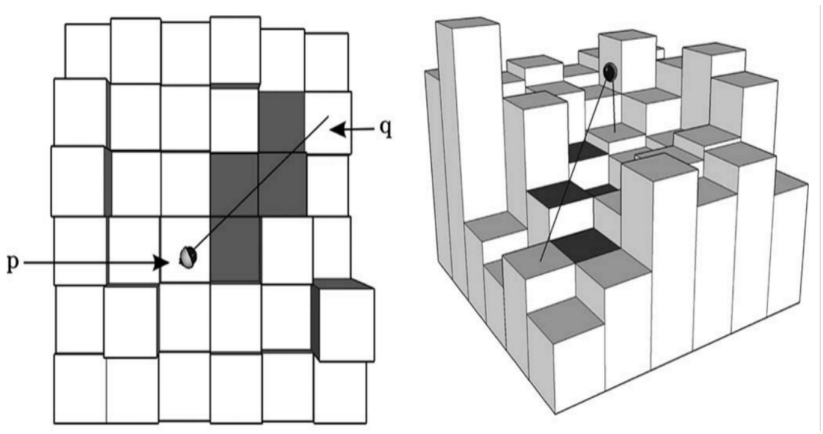

Figure 3.6: An illustration of how ray tracing is used within a 3D environment to identify potential visual obstructions. Ray tracing is used to determine which cells potentially block the line of sight between the receiver-containing cell p, and the target cell q. Shaded cells must be evaluated by the Line of Sight algorithm to determine which portion of the water column in q that is visible from p. [1]

Grid. Explicitly, this is $O(n^2 * [D^2_{range} * E_x + D^2_{range}]) = O(n^2 * D^2_{range} * [E_x + 1]) = O(n^2 * D^2_{range} * E_x)$.

3.6.4.1 Evaluation Algorithm 1

Section 3.3.2.1 gives the ERT computation for Evaluation Algorithm 1 as:
$ERT_{1,i,j} = B_{i,j} * P_{Attenuation}(Range(i,j))$

Accessing the value $B_{i,j}$, finding the distance from i to j ($Range(i,j)$)), and computing the attenuation due to range ($P_{Attenuation}(Range(i,j))$) and finding the product of the two can each be done in $O(1)$ time. Thus, the ERT computation for this algorithm is $O(1) + O(1) + O(1) = O(1)$. This computation will require $O(1)$ additional pieces of temporary storage for each intermediary value and the resulting product.

3.6.4.2 Evaluation Algorithm 2 & 3

As discussed in Section 3.3.2.2, the ERT computation for Evaluation Algorithms 2 and 3 is given as:
$ERT_{2/3,i,j} = B_{i,j} * P_{observation} * P_{Attenuation}(Range(i,j))$

As mentioned in Section 3.6.4.1, the product of $B_{i,j}$ and $P_{Attenuation}(Range(i,j))$ can be computed in $O(1)$ time with $O(1)$ temporary storage. The computation of $P_{observation}$ begins with determining the deepest depth (D_{max}) that can be seen, without bathymetric obstruction, in target cell (i,j). As previously described in Section 3.6.3, the computation of D_{max} can be done in $O(D_{range})$ time with $O(D_{range})$ temporary storage. Finally, ($P_{observation}$) is computed as the

definite integral over the shape function in $O(1)$ time. Thus, the ERT computation for this algorithm is $O(1) + O(D_{range}) + O(1) = O(D_{range})$. The algorithm will need to use $O(1)$ temporary storage for temporary storage of each of $B_{i,j}$, $P_{Attenuation}(Range(i,j))$, and $P_{observation}$. An additional $O(D_{range})$ temporary storage space is required to compute Line of Sight, and $O(1)$ temporary space to store the resulting integral. Thus, the total temporary storage required is given by $O(D_{range} + 1 + 1 + 1 + 1) = O(D_{range})$.

3.6.5 Optimal Receiver Placement

The selection of the top R_{opt} receiver locations requires the Goodness Grid to have been populated. As previously stated, the selection process is iterative (requiring that suppression (if selected) be applied after each selection), as suppression causes values in the Goodness Grid to be altered. The process of selecting a single receiver location is $O(n^2)$ since the algorithm must consider approximately n^2 possible receiver locations at each iteration. After each location selection, suppression must be applied to the chosen location. Due to the variability of suppression algorithm complexity, we use a variable ($E_{suppression}$) to represent the expected runtime of the suppression algorithm. Thus the computation time required for each selection-suppression step is $O(n^2 + E_{suppression})$. The number of iterations required to run is given by the number of optimal (R_{opt}) and projected (R_{proj}) receiver placements requested. In total, the runtime of the Optimal Receiver Placement step is $[R_{opt} + R_{proj}] * [O(n^2 + E_{suppression})] = O([R_{opt} + R_{proj}] * [n^2 + E_{suppression}])$.

3.6.6 Suppression

As discussed in Section 3.4, the suppression mechanic is applied after the placement of a receiver. The suppression mechanic has several options, each affecting the time and space complexity of the mechanism. To better capture the complexity of the algorithms involved, we denote the following variables:

D_{range}	The Detection Range of a receiver.
$d_{detection}$	The edge size of the Detection Area, equal to $2 * D_{range} + 1$ cells.
f_{supp}	The Suppression Range Factor. A non-negative real number.
r_{supp}	The Suppression Range. Equal to $f_{supp} * D_{range}$
d_{supp}	The square root of the number of cells in the Suppression Area. The edge dimension of the Suppression Area. Equal to $2 * r_{supp} + 1$ cells.

3.6.6.1 Static Suppression Algorithm

As stated in Section 3.4.2.1, the Static Suppression Algorithm multiplies all cells within the Suppression Area by a scalar constant. This is a scalar multiplication that can be done in place, requiring only $O(1)$ extra space, but a multiplication operation for each cell in the Suppression Area, which is

$O(d_{supp}^2) = O([2*f_{supp}*D_{range}+1]^2) = O([4*f_{supp}^2*D_{range}^2+4*f_{supp}*D_{range}+1]) = O(f_{supp}^2*D_{range}^2)$. In most cases (excluding those where very sparse acoustic networks are desired), the Suppression Range Factor, f_{supp}, will be rather small (less than 2.0), which reduces the runtime complexity to $O(D_{range}^2)$.

3.6.6.2 Exact Suppression

As stated in Section 3.4.2.3, the Exact Suppression Algorithm uses the user-specified Evaluation Algorithm to calculate the number of transmissions that have already been observed and should be discounted from future consideration. We define E_x to denote the expected runtime for Evaluation Algorithm x's computation of a single cell's ERT. In addition, we define T_x to denote the expected temporary storage requirement for Evaluation Algorithm x's single-cell ERT computation. The Exact Suppression Algorithm is broken into three distinct steps: Suppression Area ERT computation, Behavior Grid Update, and Goodness Grid Recalculation.

Suppression Area ERT Computation

The Suppression Algorithm first determines the ERT for each cell within the Suppression Area, and stores them in the ERT Area, a temporary Grid with the same dimension as the Suppression Area. This step is similar to the Evaluation process, with the exception that the Suppression Area's dimensions differ from those of the Detection Area by a factor of f_{supp}. The computation of the ERT Area requires E_x time and T_x temporary space for each of the d_{supp}^2 cells in the Suppression Area. An additional $O(1)$ temporary space will be needed to store the resulting ERT values for each cell. Therefore, the ERT computation will require $O(E_x * d_{supp}^2)$ time and $O((1 + T_x) * d_{supp}^2) = O(d_{supp}^2 * T_x)$ temporary storage.

Behavior Grid Update

Next, the suppression algorithm subtracts the ERT Area from the corresponding area in the Behavior Grid. This is a simple subtraction operation between each of the d_{supp}^2 pairs of corresponding cells in the Grids. Because each subtraction takes $O(1)$ time and temporary storage, the time and space complexities are both given by $O(1) * O(d_{supp}^2) = O(d_{supp}^2)$.

Goodness Grid Recalculation

Finally, the Goodness values of all cells within Detection Range of the Suppression Area must have their Goodness value updated (as we have just reduced the ERT of one or more cells within their Detection Area). The area affected by suppression, and therefore requiring recalculation, is a square with edge diameter $2 * (d_{supp} + D_{range}) + 1$. Each cell in this area will require $O(d_{detection}^2 * E_x)$ time to re-compute its Goodness Value. The total time for updating the Goodness Grid is then: $O([2 * (d_{supp} + D_{range}) + 1]^2 * O(d_{detection}^2 * E_x)) = O((d_{supp} + D_{range})^2 * d_{detection}^2 * E_x)$. The Goodness Grid update will require $d_{detection}^2 * (T_x + 1) = O(d_{detection}^2 * T_x)$ temporary storage to

compute and store intermediate ERT values. This temporary storage can be recycled for each sequential Goodness Cell computation.

The total computation time for the Exact Suppression Algorithm is then given by O(ERT Computation) + O(Behavior Grid Update) + O(Goodness Grid Recalculation). This is a total of $O(E_x * d_{supp}^2) + O(d_{supp}^2) + O((d_{supp} + D_{range})^2 + d_{detection}^2 * E_x)$ time. The algorithm will also require $O(MAX(d_{supp}^2, d_{detection}^2 * T_x)$ temporary storage, which can be recycled between each of the above stages.

CHAPTER 4
IMPLEMENTATION

This code base is intended to be treated as a tool box for the optimization of acoustic networks. While a fully-functional application is provided, its intent is to serve as a template for integrating various functionalities. It is expected that researchers will want to define and use customized versions of the provided functions (herein referred to as "subfunctions"). To support this, the framework utilizes generic "dispatcher functions" and dictionary-based parameter passing.

4.0.7 Parameter Dictionary

The framework takes as an input to the main program a dictionary of named values (`params`), and passes it throughout the program. This allows for dispatcher functions with concrete function signatures (volatile parameters can be incorporated into the parameter dictionary), and sub-functions with widely varied signatures. The benefit of this is that the function signatures of the dispatcher functions will rarely need to be changed, leading to a stable but flexible framework. The cost of this approach is that data validation of dictionary-based parameters must be preformed before the execution of sub-functions in order to avoid run-time errors within the varied sub-functions.

4.0.8 Sub-Functions

Section 3 discusses the framework's conceptual models such as Goodness computation and animal modeling. These models have loose operational requirements that can be thought of as roles. For example, it is expected that an animal model populates the Behavior Grid, and that an Evaluation Algorithm populate the Goodness Grid. As long as a user-customized sub-function fulfills its role within the prescribed model (suppressing a given location, populating a Grid, etc.), it can be easily integrated into the framework.

4.0.9 Dispatcher Functions

The primary purpose of dispatcher functions is to create a uniform interface within the framework for calling various sub-functions. Each model has a corresponding dispatcher function, and these dispatcher functions are responsible for preparing and calling sub-functions which belong to that model. For example, the Animal Model dispatcher is responsible for functions which populate the Behavior Grid. A dispatcher function performs a series of if-else checks over one or more "option" variables in the parameter dictionary, and executes the corresponding sub-function. Once a user defines a new sub-function, they should add the new sub-function as an option within the corresponding dispatcher. This will allow a user to use the new sub-function by calling the dispatcher function with the new option. Any sub-function-specific data validation should be done

at the beginning of the program, before any computationally-expensive operations are preformed, by the `checkParams()` function (Section 4.0.10.1).

4.0.10 Major Modules

4.0.10.1 Parameter Checking

As previously mentioned in Section 4.0.7, the cost of flexible inputs is substantial data validation. Within our framework, data validation should occur early on so that errors can be identified before too much time has been invested in computation or data-loading. To this end, the `checkParams()` function is provided to handle all data validation and reporting. The function takes in the `params` parameter dictionary and a `stop` boolean (set to True by default). The function and validates all internal dependencies, reporting any errors found, and halting the program if `stop` is True. The function also assigns necessary default values if no user-defined values are provided. The function returns the validated `params` parameter dictionary (with default values if necessary).

4.0.10.2 File Output

The File Output Module is responsible for generating graphical and textual results for the framework. The primary function in this module is the `graph()` function, which takes in three parameters, a `result` parameter dictionary, a `params` parameter dictionary, and a boolean, `showPlots`. The `result` parameter dictionary, contains results (Table 4.1) from successfully completing a simulation. The `params` parameter dictionary contains parameters passed into the program, and generated by the framework. `showPlots` directs file output. If `showPlots` is set to False, the function writes the output files described in Section 5.1 to disk, and returns a dictionary of file paths that point to the the newly written files. If `showPlots` is set to False, then the same results are displayed within an R session instead of being written to disk. Filenames for output files follow a `timestamp-FileName` convention, where `timestamp` is a user-specified string (defaulting to the time the request passed Parameter Validation) and `FileName` is a contextual identifier ("GoodnessGrid", "BehaviorGrid", etc). Graphic output from this module is generated using the R `graphics` package. Zip file creation uses the R `zip` package.

4.0.10.3 Bathymetry Parsing

The Bathymetric Parsing Module is responsible for the parsing, loading, and validation of bathymetric data, with the ultimate purpose of generating a Bathymetry Grid. The dispatcher function for this module is the `getBathy()` function, which requires a `params` parameter dictionary containing the key-value pairs described in Table 4.2. The function returns a validated Bathymetry Grid of `XDist` columns and `YDist` rows. As previously mentioned in Section 3.2.2.2, there are a multitude of data formats for bathymetric data, each requiring a unique data parser. To support

Key	Value	Description
`topographyGrid`	The Bathymetry Grid	An alias for the unmodified Bathymetry Grid.
`behaviorGrid`	The Behavior Grid	The unmodified Behavior Grid.
`goodnessGrid`	The Goodness Grid	The unmodified Goodness Grid.
`coverageGrid`	The Coverage Grid	The coverage computed from the resulting receiver array.
`recoveryRates`	Receiver array data	A dictionary of receiver array-related results obtained as an output from the simulation. The dictionary is generated by `sensorFun()`.
`stats`	Receiver array statistics	A dictionary of receiver array-related statistics obtained from calling getStats().

Table 4.1: A summary of the key-value pairs in the `result` dictionary.

this diversity of formats, the `getBathy()` function reads the `inputFileType` from the `params` dictionary to select an appropriate parser. The selected parser begins reading data at column `startX` and row `startY` from the data file specified by `inputFile`. The `getBathy()` function also checks to makes sure that all cells in the Bathymetry Grid have rational depth values. N/A, INF, or -INF values are invalid as they will contaminate future computations, resulting in runtime errors. The module replaces invalid values with a value of '0'. By default, the framework supports the NetCDF type, and thus requires the `ncdf` or `ncdf4` R packages. Both the `ncdf` and `ncdf4` packages serve as an R interface for the NetCDF C/FORTRAN library, and thus requires that a working NetCDF distribution be installed on the system.

Key	Value	Description
`inputFile`	File path to Bathymetry File.	A relative or absolute Bathymetry File path.
`inputFileType`	Parser name.	Indicates which parser to use to read the Bathymetry File.
`XDist`	Bathymetry Grid column count.	Number of columns in the Behavior Grid.
`YDist`	Bathymetry Grid row count.	Number of rows in the Behavior Grid.
`startX`	Column index to read from.	The column index in the Bathymetry File to begin reading from.
`startY`	Row index to read from.	The row index in the Bathymetry File to begin reading from.

Table 4.2: A summary of the `params` key-value pairs used in Bathymetry Parsing module.

4.0.10.4 Animal Modeling

The Animal Modeling module is responsible for simulating animal distribution, and ultimately generating the Behavior Grid. The dispatcher function for this module is the `fish` function, which takes in a Bathymetry Grid and a `params` parameter dictionary containing the key-value pairs

described in Table 4.2.

As previously mentioned in Section 3.2.3, the Animal Model offers separate horizontal and vertical movement models. The animal movement models are handled by the `fish()` function, which requires a `params` parameter dictionary. The `fishModel` dictionary key specifies which horizontal movement model should be employed by the module to generate a population distribution. Because the vertical movement models operate independently of the horizontal movement model and each other, we provide separate option variables for both vertical movement models. The Restricted Vertical Habitat Range model is used if two keys, (`mindepth` and `maxdepth`), are specified within the `params` dictionary. Similarly, the Habitat Preference model is used if `depth_off_bottom` and `depth_off_bottom_sd` keys are present within the `params` dictionary.

Key	Value	Description
`fishModel`	Behavior Model Name.	The chosen Horizontal Behavioral Model.
`mindepth`*	Min depth for animal habitat.	Minimum depth in the Restricted Vertical Habitat Range Model.
`maxdepth`*	Max depth for animal habitat.	Maximum depth in the Restricted Vertical Habitat Range Model.
`depth_off_bottom`*	Preferred animal habitat depth.	Preferred depth in the Habitat Preference Model.
`depth_off_bottom_sd`*	SD of preferred depth.	Standard Deviation of preferred depth in the Habitat Preference Model.

Table 4.3: A summary of the `params` key-value pairs used in Animal Modeling module.
* optional parameters.

4.0.10.5 Goodness Computation

The Goodness Computation module is responsible for computing the goodness of a particular cell (Section 3.3). The dispatcher function for this model is the `goodnessGridFun()`, which takes in `params`, a parameter dictionary, and `grids`, a dictionary containing the Bathymetry Grid, and Behavior Grid. The `bias` option is used to tell the module which sub-function should be used to compute Goodness. All necessary parameters for the chosen sub-function are passed in via the `params` dictionary. The module is responsible for computing and adding the GoodnessGrid to the *grids* dictionary. Unlike the Animal Population and Bathymetric parsing modules, this module does not directly return a Grid object, but inserts it into the `grids` dictionary. This behavior reflects the idea that the Goodness is logically dependent upon bathymetry and animal behavior. Furthermore, this behavior is intended to guide users towards creating a Bathymetry and Behavior Grid before generating a GoodnessGrid.

Because the Goodness computation can take a substantial amount of time, sub-functions which compute Goodness should inform users what percentage of the computation has been completed.

The `silent` parameter is a boolean that disables this update when set to True.

4.0.10.6 Suppression

The Suppression module is responsible for suppressing (Section 3.4) the Goodness of the area around a particular cell on the Goodness Grid. The dispatcher function for this module is the `suppressionFun` function, which takes in `params`, a parameter dictionary, `grids`, a dictionary containing the Goodness, Bathymetry, and Behavior Grids, and `loc`, a dictionary mapping the keys 'r' and 'c' to the row and column indexes of the cell to be suppressed. The module is responsible for reducing the values around an area, given by `loc`, on the GoodnessGrid and returning the suppressed GoodnessGrid. Note that the suppression module is only responsible for performing suppression operation on a single target location, `loc` (recall that the process of selecting a receiver location and suppressing it is an iterative process, and therefore must be interleaved). The method in which suppression is applied is given by the `suppressionFcn` option in `params`. The dimensions of the square area affected by suppression is given by `suppressionRange`. Section 3.4.2 describes how `minsuppressionValue`, `maxsuppressionValue`, and `shapeFcn` can be used to define a suppression template or gradient.

Key	Value	Description
`suppressionFcn`	Suppression Function Name.	The name of the suppression
`suppressionRange`	Distance from loc to suppress.	How far (in cells) out from loc to suppress.
`receiverElevation`	Elevation off sea floor.	Sensor elevation (in meters) off the see floor.
`minsuppressionValue`*	Min suppression multiplier.	Minimum penalizing coefficient.
`maxsuppressionValue`*	Max suppression multiplier.	maximum penalizing coefficient.
`shapeFcn`*	Distribution Function ID.	Name of the statistical distribution model to use.

Table 4.4: A summary of the `params` key-value pairs used in the Suppression module.
* optional parameters.

4.0.10.7 Sensor Placement

As mentioned in Section5.3.2, a tight coupling exists between the heuristic and Evaluation Algorithms. The sample application provided is very basic, utilizing a brute-force goodness computation on all cells in a the study area, and an exhaustive search to identify potential candidates for receiver placement. Also recall that the selection of receiver placements and suppression are an iterative process. To satisfy both of these relationships, the three operations are combined into a larger Sensor Placement module. The dispatcher function for this module is `sensorFun`, which handles goodness computation, the selection of receiver locations, and the suppression of those locations. It is important to note that while this function is important and warrants mention, it

is not an extensible "module". That is to say, it is merely a driver function that binds together three independent modules (Goodness Computation, Sensor Placement and Suppression), handling the (minimal) dependencies between each. Modifying the nature in which receiver placements are chosen for consideration will require a customized driver. The `sensorFun` function takes in `grids`, a dictionary containing the Bathymetry and Behavior Grids and a parameter dictionary, `params`. The module is responsible for outputting the `result` dictionary, which contains the Bathymetry, Behavior, and Goodness Grids, a list of receiver placements (both user and system placed receivers), statistics regarding those receivers (absoloute and unique data recovery rates, and a delta value for the array), the `params` parameter dictionary, and a list of warnings/errors.

4.0.10.8 Sample Application

To provide a demonstrable product, and program outline, a simple, cohesive application, `acousticRun`, is provided. The application makes use of framework functions to validate parameters, construct Bathymetry, define Behavior, compute Goodness, and choose optimal receiver locations with suppression. The application supports all initial functionality provided in the framework, and allows users to call these functions over their own datasets. As previously noted, the application implements a brute-force computation of goodness for every cell in the Goodness Grid, and employs an exhaustive search to select optimal receiver locations. Due to this, the application is suitable for small, sample simulations, but might not be appropriate for very large scale simulations. Documentation [4] for the application lists all available parameters, descriptions, and constraints. Due to the large number of parameters available, the program also provides a "default" simulation when no parameter are given.

4.0.10.9 Web Server

Included alongside the framework is a web-application (webapp) with a Graphical User Interface (GUI) that runs a simple demonstrative (demo) application using the framework. The `Rook` R package is used to handle HTTP interactions between the GUI and the R demo application. The GUI is a simple HTML page that generates a list of parameters (including a unique *timestamp* identifier) based on user input, creates a JSON (JavaScript Object Notation) string, and sends an HTTP request to the Rook server with that JSON string. Using the `rjson` library, the Rook server decodes the JSON string into R data objects, and calls the demo application using the decoded parameters. The Rook application signals the successful reception and decoding of the JSON string as an HTTP 200 status update. The GUI then begins waiting for the completion of the simulation (which may take some time) by continuously checking (ignoring failures) for the existence of a text document (named for the unique *timestamp* identifier) on the server. When the simulation finishes its execution (either via error or normal completion), it writes its several output files (Section 5.1), and a text document containing the JSON representation of the outputs (file paths to graphics and

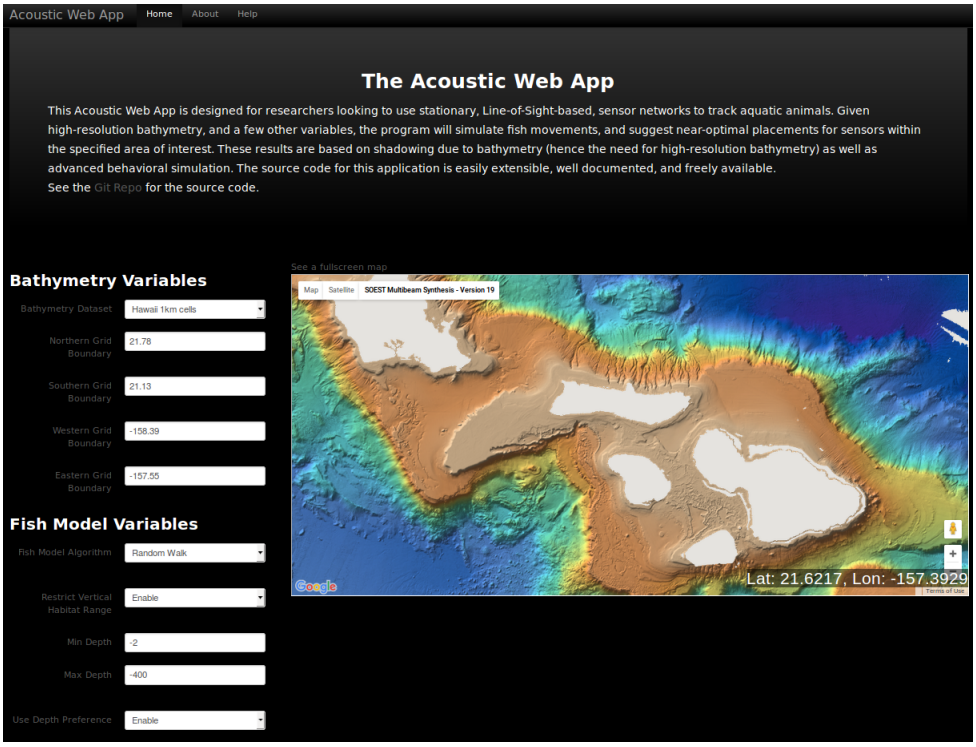

Figure 4.1: A screenshot of the webapp GUI. On the right is a Google Map widget that assists users in choosing lattitude and longitude boundaries for their study site (defining their Bathymetry Grid). On the left is a list of simulation options implemented as drop-down lists, and simulation parameters implemetned as text boxes. At the bottom (hidden) is a "Sumbit" button that sends that validates the parameters and sends them to the server. All parameters fields and lists display "tool tip" style descriptions of the parameter that field/list represents.

a dictionary of results). All output files are tagged with a unique *timestamp* identifier. Once the GUI finds an output file with the correct *timestamp* signature, it reads the file, parses the results, and displays them within an HTML page. If the local system supports it, a zip file of all result files is also supplied. The webapp serves to simplify the selection and specification of parameters to the demo application. This allows users to quickly and easily understand the utility of the framework and application.

4.0.11 Implementation Hurdles

4.0.11.1 Complexity and Usability

The primary trade-off that occurs with any application is that between simplicity and control. While a simpler, less-detailed application may lower the effort required, it inevitably reduces the level of control a users has over the application. A significant number of variables and options may serve to increase control over an application, but also require increased time and effort to

understand and use effectively. Put simply, very specific actions require very detailed instruction, which in turn requires significant effort. On the other hand, simpler applications may be more easily understood and adopted, but ultimately lack higher levels of control. To address this trade-off, both a demo application (Section 4.0.10.9) and framework are provided. The demo application provides an easy-to-use application that works "out-of-the-box" and has a useful subset of features, while the framework provides access to more detailed functionalities for advanced users.

4.0.11.2 Coordinate Conversion

As mentioned in Section 3.2.2.1, bathymetric files are composed of a regular grid of cells. However, because the Earth is a 3D sphere, translation is necessary to map that 3D sphere to a 2D grid. This translation introduces potential error into the specification of a study site. One such source of error occurs when the method of translation is not specified, leading to ambiguity in how grid cells should be read. Another possible source of error can be attributed to rounding errors in the number of degrees latitude and longitude represented by each cell. In grids spanning hundreds of thousands of cells, small rounding errors can lead to large positional discrepancies. To deal with these issues, the framework requires users to specify their study site as grid cell indexes. This puts the burden of translating latitude and longitude into gird coordinates on the user (who likely knows best how to handle their data). Additionally, final receiver placements are given in terms of global (relative to the input bathymetric file) and local (relative to the defined study site) indexes. This facilitates error checking and helps to identify translational errors.

4.0.11.3 Rook

The *Rook* package was selected as a webserver because it offered support for cloud-based web hosting on Heroku and Amazon Web Services (AWS). This feature was desirable as it allowed users to quickly create a working instance of our web application without modification to their local computers. Unfortunately, the *Rook* webserver is implemented using a single, blocking thread. This means that *Rook* is only able to handle one request at a time, and will wait for a request to complete before accepting any other other requests. Because the same Rook thread that receives a request must also call the demo application (which takes about 30 seconds to complete), the entire webserver is unresponsive to new requests until the demo application completes its execution. This unresponsiveness can lead users to feel like the application is "frozen" or "broken". To solve this problem, a loading icon (spinning circle) is used to assure users that the webapp is working. Another issue with the *Rook* package is that it imposes a 250 character limit on the length of POST requst data. Because of the substantial number of parameters sent to the demo application, POST data strings are often very long ($\geq$500 characters). To compensate, POST data (the JSON string representing user-specified parameters) from the HTML page must to be cut into shorter segments and sent with sequential identifiers and re-assembled on the server.

CHAPTER 5
RESULTS

5.1 Output Files

Our program outputs 6 files:
1) The Bathymetry Grid as a heat map
2) The Behavior Grid as a heat map
3) The Goodness Grid as a heat map
4) The Coverage Grid as a heat map
5) The marginal gain in Unique Recovery Rate as a function of the number of receivers placed
6) Text representations of the 5 Grid files, tabular receiver data, and the specified input parameters.

5.1.1 Grid Graphs

The four Grid files (Bathymetry, Behavior, Coverage, and Goodness) are visualized as a heat maps. All of these are overlaid with the resulting receiver locations as numbered circles. Receiver locations show their detection range as dotted circles centered over the receiver locations. User placed receivers are colored gray, receivers placed optimally by the system are colored blue, and projected receivers (also placed by the system) are colored green. The numbering on receivers denotes the rank of each receiver's Unique Data Recovery Rate. User and system-placed receivers are ranked separately. The highest ranked system-placed receiver locations are returned as the optimal receiver locations, with the lower ranked locations as the projected receiver locations.

5.1.2 Data Recovery Graphs

The program also produces a graph of the marginal increase in and cumulative sum of the Unique Data Recovery Rate as a function of the number of optimally placed receivers used (Figure 5.1). The graph of the marginal increase in UDRR (the lower graph) is especially useful. Given that Receivers have a fixed per-unit cost, it makes sense to weigh a receiver's effectiveness (in this case, UDRR contribution) against its cost to determine the utility of purchasing an additional receiver. The graph of cumulative UDRR is useful to quickly identify the number of receivers necessary to reach a goal UDRR.

5.1.3 Text Files

The program returns a comprehensive text representation of the program output (text dump of gridded data and input parameters), and a short, human-readable document that lists the primary

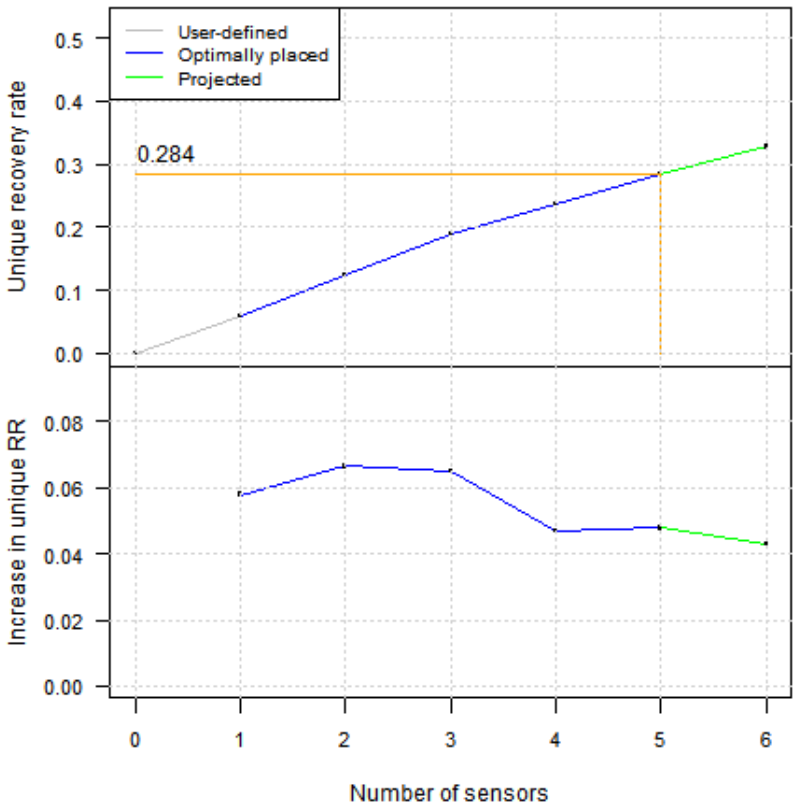

Figure 5.1: A graphical representation of the cumulative and per-receiver UDRR. User-placed receivers are representd by a grey line, system-placed receivers are represented by the black line, and projected receivers are represented by the green line.

simulation parameters (Evaluation Algorithm, Suppression Algorithm, Behavioral Model, Input Grid Size, and Detection Range), as well as a tabular output of receiver placements (coordinates), data recovery rates for each receiver, and network sparsity. Projected receivers are excluded from the total UDRR, ADRR, and sparsity values.

Coordinates are returned in both a Global (with respect to the original Topography file) and local (the user specified area of interest) frame. While the curvature of the earth is well documented, different bathymetric maps may handle the mapping of a 3D curved plane to a 2D Grid differently. For example, one Grid may implement some scaling on Grid cells a function of the cell's latitude or distance from a certain point. Other Grid files may simply provide non-square Grid files. By proving a small-scale (local) and large-scale (global) frame of reference for our receiver locations, irregularities of this nature are more easily detected.

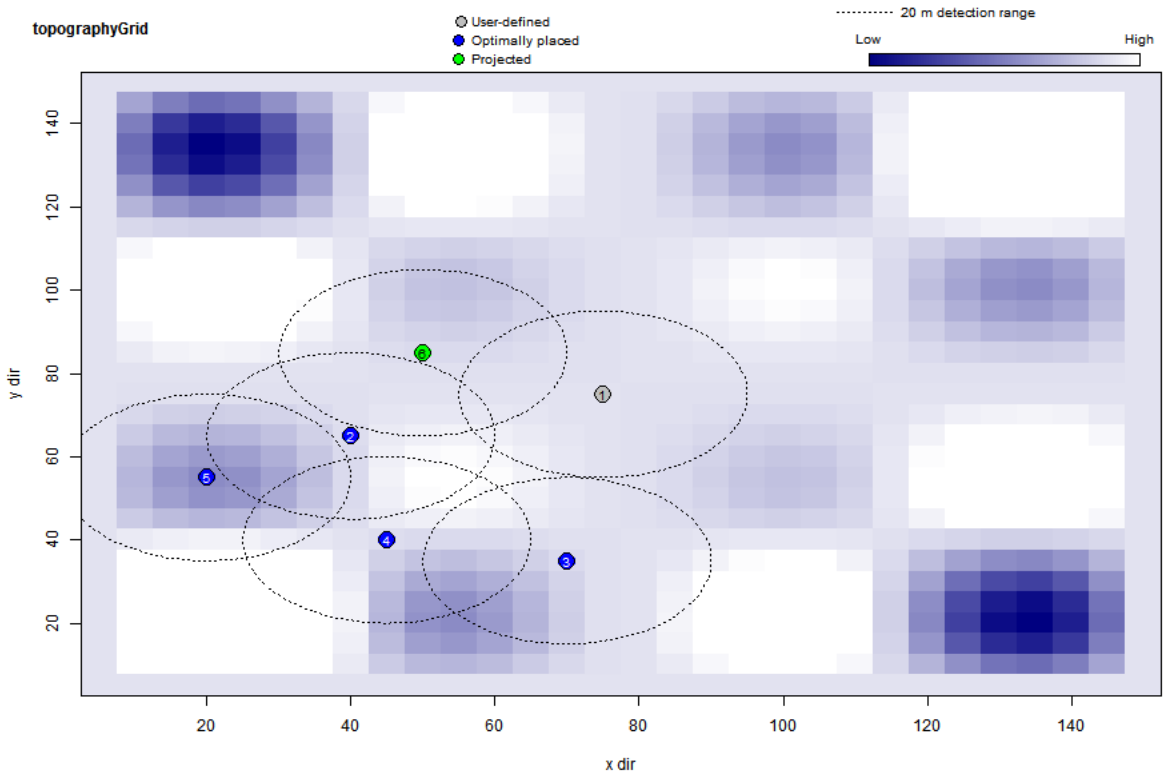

Figure 5.2: A graphical representation of an artificial Bathymetry file at a 1m resolution (each cell has an edge length of 1m). Darker cells represent greater depths, while white cells represent inaccessible terrain (dry land). The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines.

5.2 Distinguishing Characteristics

The challenge of acoustic network design is one that is common within the acoustic tracking community, yet rarely addressed. While there are many methodologies, applications, and services that focus on achieving various objectives, this framework distinguishes itself in its availability, and adaptability.

5.2.1 Availability

As previously stated, our framework is freely available [3] and falls under the GNU General Public License (GPL 2). This availability means our methodologies are open-sourced, fully transparent, and are available for peer-review. Furthermore, our framework is accessible to researchers without requiring expertise in R. The web application provides a useful set of well-documented functionalities with a user-friendly interface.

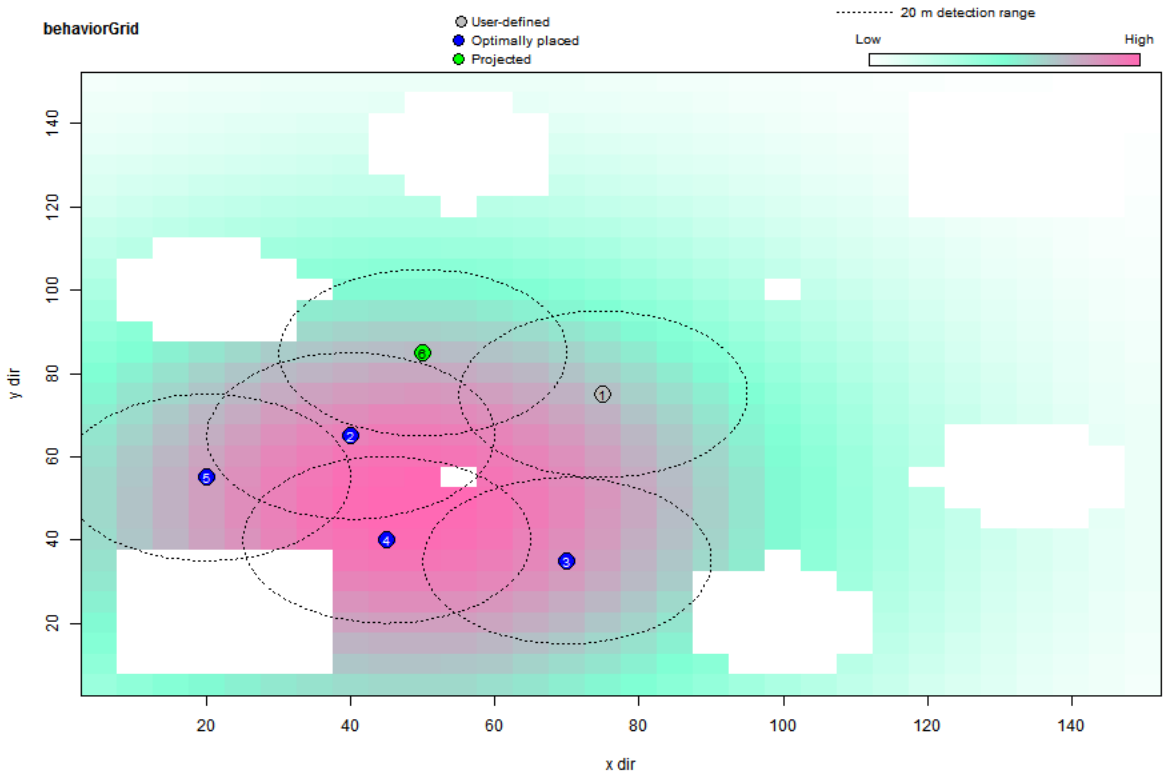

Figure 5.3: The Behavior Grid represented as a heat map Higher levels of animal residency correspond with pink cells, moderate levels as light blue, and white for non-residency (inhospitable habitat such as dry land). The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines.

5.2.2 Adaptability

5.2.2.1 Flexability

Many of the modules in this framework can be used as stand-alone functions that perform useful operations (reading bathymetric data, calculating distribution densities, etc). This modularity allows the framework to function more like a library for acoustic simulation than a fixed application.

5.2.2.2 Simple Data Formats

Many key functionalities of our framework (Evaluation Algorithms, Suppression, Animal Modeling, Network Statistics) operate over very simple data containers (Grids). This simplified data format makes it easy to import, export, and convert between other data formats. This simplified format also makes it easy to move data between this framework and other R packages.

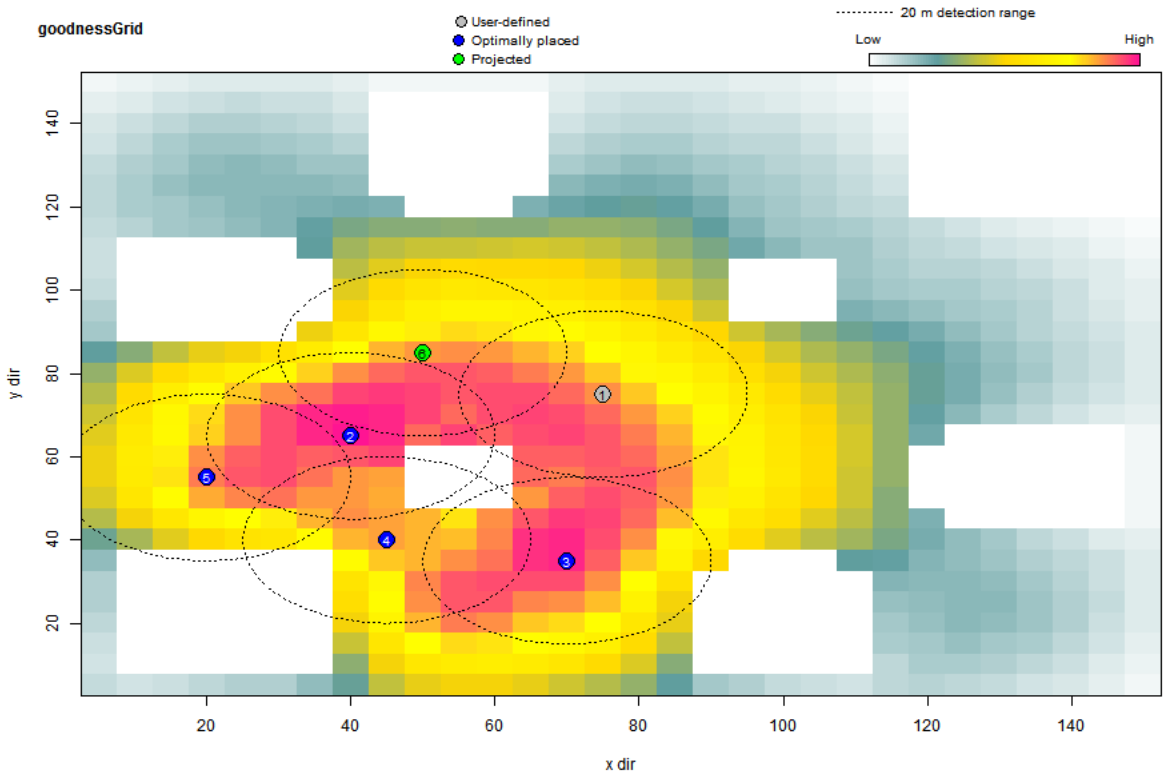

Figure 5.4: The Goodness Grid represented as a heat map showing the sum total of Estimated Receivable Transmissions (ERT) for a receiver placed in a particular cell. The legend in the top right assigns color coding to various ERT values. The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines.

5.2.2.3 Customizable

While other applications and methodologies optimize for fixed-objective functions (k-neighbor coverage, fixed detection probability, etc), our framework allows for user-customized objective ("Goodness") functions. Section 4.0.9 discusses dispatcher functions, which serve as abstract entry points into stand-alone modules, and allow highly-customized sub-functions to be aggregated together into uniform modules. These dispatcher functions makes it easy to integrate new functionality into the framework without affecting other sub-functions.

5.2.3 Drawbacks

5.2.3.1 Explicit Data Validation

In most all software systems, future efforts to add new functionality require a different set of parameters than those initially specified. Thus, future developers will likely need to modify function

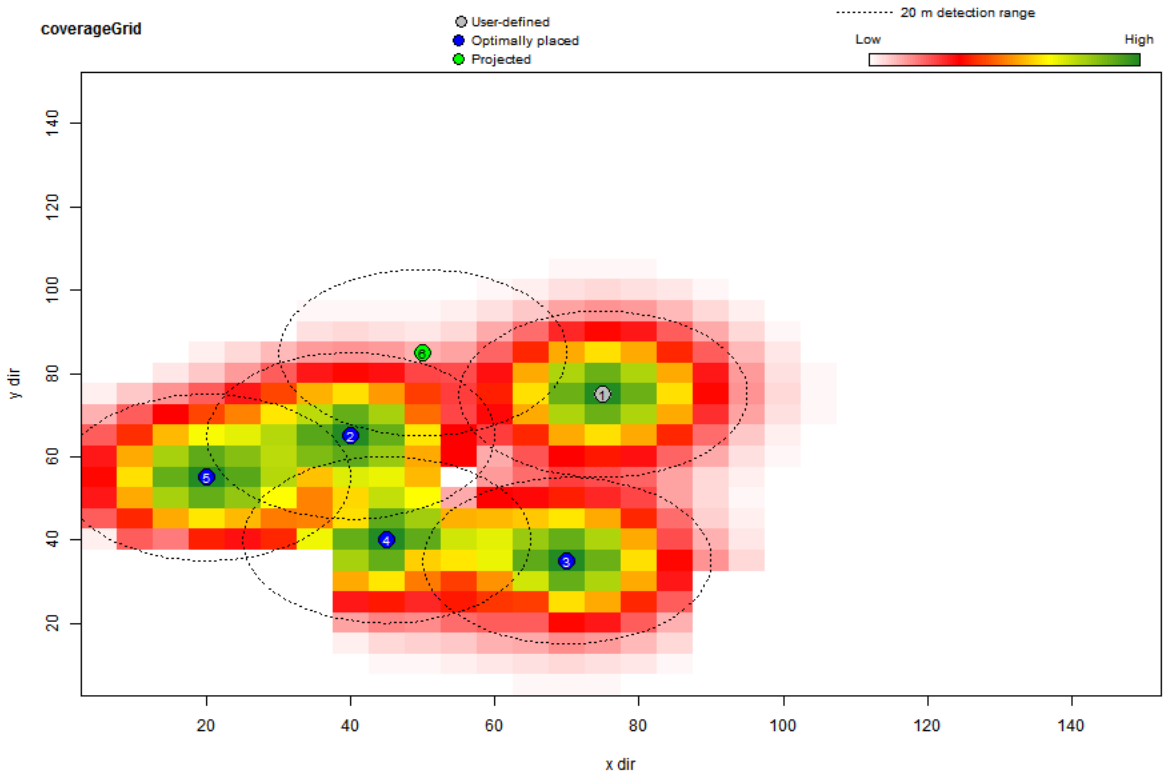

Figure 5.5: The Coverage Grid represented as a heatmap showing the quantity of Estimated Receivable Transmissions from each cell in the study site, for the designated receiver array configuration. The legend in the top right assigns color coding to various ERT values. The optimal receiver locations are shown on the Grid as blue numbered circles, user-placed sensors as grey circles, and projected receivers as green circles. All receivers have their Detection Range shown as dotted lines. The missing corner of of Receiver 4's Detection Area is due to the presence of an obscuring section of Bathymetry (dry land).

signatures to incorporate said functionality. This modification can substantially impact the way users interact with the system, and will require other developers to become familiar with a modified interface. Additionally, these changes can potentially break older functions that relied upon the old function definition. To reduce this brittleness (a small change causing a system to break) associated with explicit parameters, we utilize dictionary-based parameter passing. While explicit parameters cause brittleness within a system, they serve to create a contract between users and functions, such that if users provide those explicit parameters, the function will operate as intended. This contract is enforced by the programming language, resulting in compilation/interpretation errors when these parameters are violated. Without explicit parameters, this contract is gone, and the responsibility of enforcing the relationship (by way of key-value validation within the dictionary) between users and functions falls to either the user or the developer. In short, dictionary-based parameter passing provides reduced brittleness and simplified future-development but passes the

responsibility of data validation to users and developers.

5.2.3.2 Technical Capability

While a fully functional application is provided as a demonstration of this framework, the framework's greatest usefulness to a tracking project will likely not be realized unless custom Goodness functions are defined. Because substantial expertise in programming is required to author these custom functions, the framework is less useful to users with little R experience.

5.2.3.3 Language Limitations

This framework was implemented in the R programming language primarily due to its popularity within the biological community. While R is sufficient for small and medium scale simulations, a compiled language would offer significantly faster execution at all simulation levels. Additionally, the R platform does not offer good support for reverse compatibility. Specifically, R packages need to be recompiled under the local system's version of R. If this compilation is unsuccessful (perhaps due to a difference in R versions, or operating system limitations), the package will not function. If an application requires a significant number of R packages, then it is likely to be limited to operating within specific versions of R.

5.3 Future Work

5.3.1 Emperical Analysis

The analytical framework described herein is largely theoretical, despite a basis in well documented physical phenomena. In order to better understand the correctness of our analysis, it would be useful to analyze empirical data from small-scale acoustic networks designed with this application. Specifically, we are interested in the accuracy of our network coverage projections, and the affect of bathymetric resolution on this coverage.

5.3.1.1 Acoustic Coverage

Recall that estimated data recovery rates are calculated based upon acoustic coverage and animal distribution. While we provide basic Animal Models, the accuracy of these models (and as a result the accuracy of our data recovery rates) is very species-dependent and outside the scope of this paper. While data recovery rates are based on animal models, our acoustic coverage model is not. Thus, we are very interested in the accuracy and optimality of our bathymetric shadowing and network coverage models. It would be interesting to compare the acoustic coverage of a real-world network designed by our application against our theoretical coverage of that same network.

5.3.1.2 Bathymetric Resolution

Section 3.2.2 describes the nature of artificially increasing bathymetric resolution. It would be interesting to see how varying the native resolution of bathymetry affects the acoustic coverage of network topology both within our framework and in the real world.

5.3.1.3 Workflows

The workflows described in Section 3.1.2 were based upon both personal experience and imagined use. It would be good to investigate the actual use cases, workflows, and problems that researchers face at various points throughout the lifetime of an acoustic tracking project. By identifying unconsidered problems and workflows, the framework can be modified/expanded to encompass new roles and features, leading to greater usefulness.

5.3.2 Heuristic Algorithms

Recall that the function of the Goodness Grid is to store Goodness values for various locations within the study area. The sample application simply computes every possible goodness value in the Goodness Grid, and then finds the top values Goodness values via R's $max()$ function. While this process is guaranteed to find optimal receiver positions, it is computationally intensive, accounting for the vast majority of the program's run time. Utilizing heuristic algorithms can significantly reduce the number of Goodness values that are computed, but can also reduce the optimality of the final acoustic array. Heuristics are currently not included in the framework because functions involving Evaluation Algorithms and any new heuristic algorithms will likely be tightly coupled (a heuristic algorithm will likely need to consider an Evaluation Algorithm's definition of *Goodness*). Furthermore, to avoid computational overhead (such as passing large lists of heuristic candidate cells), it is likely that the heuristic algorithm and Evaluation Algorithm will be called from within the same function. This will likely lead to a large number of customized functions that implement heuristic algorithm-Evaluation Algorithms combinations. It would be very useful to evaluate the cost of programatically de-coupling Evaluation and heuristic algorithms and passing large lists of heuristic candidates. This data could then be compared against the measured benefits of various heuristic algorithms to evaluate the best methodology for integrating heuristics into the framework. Another useful avenue would be the investigation of Evaluation Algorithm agnostic heuristics, which could provide better run times without tight couplings.

5.3.3 Interactive Visualizations

It could be useful to allow users to interact with the output graphs as a graphical user interface. Users would able to move receivers around the graph and see how new placements affects network statistics in real-time. Given that the Goodness Grid can be pre-computed, moving receivers and

updating the effects of that movement in real-time requires very little re-computation. In this case, it might be unnecessary to even pre-compute the Goodness Grid, but instead only preform the Evaluation Algorithm on cells where a user places a sensor. This would allow users to quickly see how various receiver arrangements affect their network design, and facilitate rapid, impromptu prototyping of various configurations.

Utilizing 3D visualizations could further enhance the user's understanding of network topology. Figure 5.6 illustrates a first-person view of the 3D environment modeled by the program. Here, bathymetry is rendered as 3D mounds, with animal distributions represented by green fish, and receiver coverage by red spheres. Users might be able to "swim" through the 3D environment, viewing their network from various angles. Such a visualization would help users better understand what is being modeled, how their array is laid out, and where modifications might be helpful.

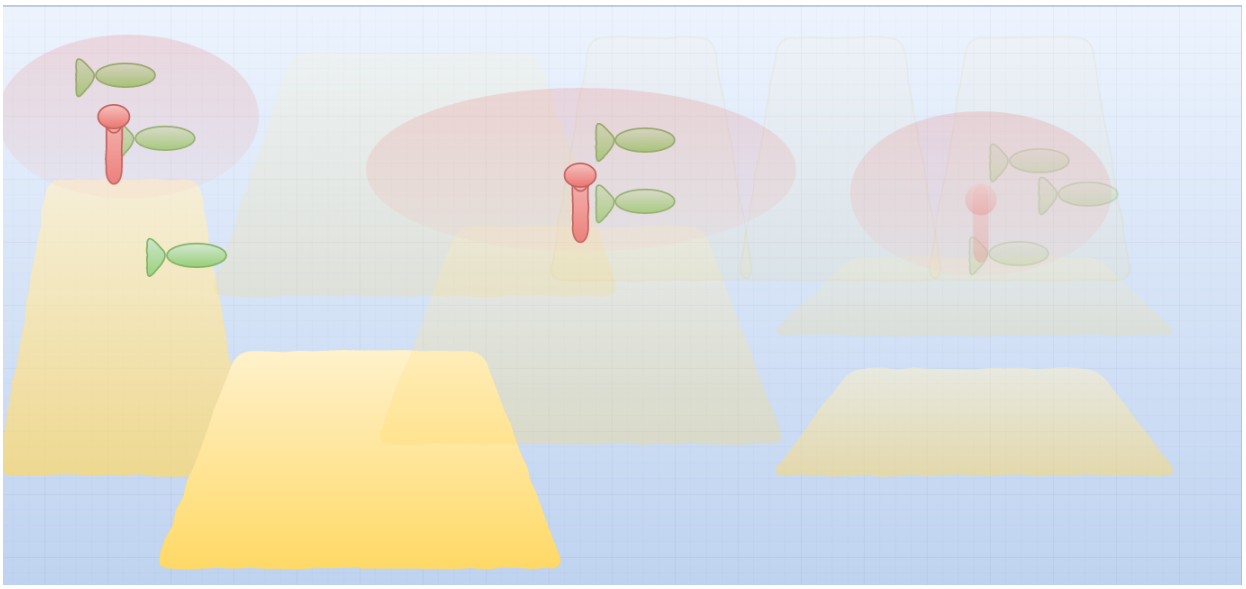

Figure 5.6: A first-person view of the 3D space modeled in the program. Users would be able to "swim" through the environment and gain a better perspective on what they are modelling, what their receiver array might look like in the environment, and where more receivers might be needed.

CHAPTER 6
CONCLUSIONS

We have presented a framework for the optimization and measurement of Static Acoustic Observation Networks (SAONs), the benefits of which are decreased cost of data collection, increased volume of collected data, and higher data quality. While these benefits are largely realized early on during the design of a SAON, the framework's utility extends into the augmentation and assessment of acoustic tracking studies. The ability to include an already existing SAON in the design process allows for the optimal augmentation and integration of SAONs. This is useful to projects that operate on annual budgetary allotments and periodically purchase and deploy of hardware. By supporting this incremental growth pattern, our framework can be used to make optimal use of limited resources. This functionality also provides a mechanism for researchers from separate studies to pool resources, mesh disjoint networks, and collaboratively increase data recovery rates. Our also framework provides automated metrics for the evaluation and comparison of SAONs. These metrics facilitate the evaluation of existing and theoretical network topologies, which is useful when demonstrating the need for additional hardware.

Our framework is customizable, allowing users to develop their own Behavior, Evaluation, and Suppression Algorithms, Shape Functions, and metrics. Coupled with the fact that our program is open source (the source code is freely available and editable [3]), and free to use, we expect this tool will have a strong impact on the acoustic tracking communities.

APPENDIX A
TABLE OF NOTATIONS

A	Bathymetry Grid - A 2D Grid of non-negative real numbers that describe the depth of a section of the sea floor. $A_{i,j}$ refers to the value in the cell at row i, column j. A_x refers to the value in cell x.
B	Behavior Grid - A 2D Grid of non-negative real numbers that indicate the percentage of all transmissions that are expected to be released from that cell over the entire experiment. $B_{i,j}$ refers to the value in the cell at row i, column j. B_x refers to the value in cell x.
G	Goodness Grid - A 2D Grid of real numbers with the same dimensions as the Bathymetry Grid. Contains real numbers that indicate how good a particular cell will serve as a receiver placement. $G_{i,j}$ refers to the value in the cell at row i, column j. G_x refers to the value in cell x.
n	The square root of the total number of cells in the Bathymetry Grid (A). Approximates the edge dimension of a square Bathymetry Grid.
k	The fixed elevation of receivers off of the sea floor in meters. Assumed to be constant for all receiver emplacements in the simulation.
$ERT_{x,T,i,j}$	Estimated Receivable Transmissions - The number of transmissions that are expected to be received from a cell at row i, column j by a receiver at cell T, according to Evaluation Algorithm x.
$P_{Attenuation}(x)$	Probability of Detection due to attenuation - Describes the probability of detecting an acoustic transmission originating at a range of x meters away from a receiver.
$P_{observation}(T, a, b)$	Percentage of transmission in cell (a, b) which are observable (due to bathymetric obstruction) from cell T . Given a receiver in cell T, and some distribution of animals in the water column of cell (a, b), this value represents the percentage of observable animals in (a, b) from A. Animal distribution is handled by the Vertical Movement Model (Section 3.2.3.3). Section 3.3.3 gives the algorithm for determining bathymetric obstruction and Line of Sight.
$Range(T, i, j)$	Distance from a particular receiver location T to the cell at row i, column j. Computed as the Euclidean distance between the center of a receiver-containing cell T and the center of the cell at row i, column j on a 2D Grid.
R_{opt}	The number of optimal receivers to be placed by the system (excluding projected receivers).
R_{proj}	The number of projected receivers to be placed by the system.
$D_max(p, q)$	The deepest observable depth within a target cell q, relative to the receiver containing cell p. See Section 3.3.3

$Edist_{p,v}$	The Euclidean distance between cells p and v.
$m_{p,q}$	The slope between cells p and q, expressed as the change in bathymetric elevation (of the sea floor) between cells p and q, divided by the Euclidean Distance between cells p and q. See Section 3.3.3
D_{range}	The detection range of the receiver. Assumed to be the same for all receivers in the network. Given as the distance between a receiver and tag at which there is a 5% chance of detecting an acoustic transmission.
$d_{detection}$	The edge size of the Detection Area, equal to $2r + 1$ cells.
f_{supp}	The Suppression Range Factor. A positive real number.
r_{supp}	The Suppression Range. Equal to $f_{supp} * D_{range}$
d_{supp}	The square root of the number of cells in the Suppression Area. The edge dimension of the Suppression Area. Equal to $2 * r_{supp} + 1$ cells.

BIBLIOGRAPHY

[1] Vahab Akbarzadeh, Christian Gagné, Marc Parizeau, Meysam Argany, and Mir Abolfazl Mostafavi. Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage. *IEEE Transactions on Instrumentation and Measurement*, 62(2):293–303, 2013.

[2] Perry Barboza. Innovative technology/lab support proposal online form. `<https://www.uaf.edu/tab/past-proposals/proposalDetails.xml?id=667>`, 2014. [Online; accessed 19-November-2015].

[3] Gregory Burgess, Martin Pedersen, and Kevin Weng. Acousitc-deploy. `<https://github.com/gregorylburgess/acoustic-deploy/archive/Update.zip>`, 2013-2016. [Online; accessed 27-July-2016].

[4] Gregory Burgess, Martin Pedersen, and Kevin Weng. Acousitc-deploy wiki. `<https://github.com/gregorylburgess/acoustic-deploy/wiki>`, 2013-2016. [Online; accessed 27-July-2016].

[5] Rey Farve. Technology and development at the usda forest service, satellite/gps telemetry for monitoring lesser prairie chickens. `http://www.fs.fed.us/t-d/programs/im/satellite_gps_telemetry/wildlifetrackingtelementry.htm`. [Online; accessed 11-September-2015].

[6] Gretchen J. A. Hansen and Michael L. Jones. The value of information in fishery management. *Fisheries*, 33(7):340–348, 2008.

[7] M. R. Heupel, J. M. Semmens, and a. J. Hobday. Automated acoustic tracking of aquatic animals: Scales, design and deployment of listening station arrays. 57(1):113, 2006.

[8] Andrew Howard, M.J. Mataric, and G.S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02)*, 5:299–308, 2002.

[9] Telonics Inc. Vhf systems for fish (fis). `<http://www.telonics.com/products/vhfImplants/vhfFish.php>`. [Online; accessed 5-November-2015].

[10] Steven Thomas Kessel, Nigel Edward Hussey, Dale Mitchell Webber, Samuel Harvey Gruber, Joy Michelle Young, Malcolm John Smale, and Aaron Thomas Fisk. Close proximity detection interference with acoustic telemetry: the importance of considering tag power output in low ambient noise environments. *Animal Biotelemetry*, 3(1):1–14, 2015.

[11] Ross A. Maller, Gernot Müller, and Alex Szimayer. *Handbook of Financial Time Series*, chapter Ornstein–Uhlenbeck Processes and Extensions, pages 421–437. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[12] James McNeill. Playtechs: Programming for fun: Raytracing on a grid. `http://playtechs.blogspot.com/2007/03/raytracing-on-grid.html>`, March 2007. [Online; accessed 5-May-2016].

[13] Martin W. Pedersen and Kevin C. Weng. Estimating individual animal movement from observation networks. *Methods in Ecology and Evolution*, 4(10):920–929, 2013.

[14] Martin W. Pedersen and Kevin C. Weng. A state-space model for estimating detailed movements and home range from acoustic receiver data. `<http://orbit.dtu.dk/en/publications/a-statespace-model-for-estimating-detailed-movements-and-home-range-from-acoustic-receiver-data(85d741c7-33ae-47f0-aac0-f0af6b244e3d).html>`, 2013. [Online; accessed 19-April-2016].

[15] S. Poduri and G.S. Sukhatme. Constrained coverage for mobile sensor networks. *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, 1, 2004.

[16] Collecte Localisation Satellites. How argos works. `<http://www.argos-system.org/web/en/337-how-argos-works.php>`. [Online; accessed 5-November-2015].

[17] Seaturtle.org. Wildlife tracking. `<http://www.wildlifetracking.org/faq.shtml>`. [Online; accessed 5-November-2015].

[18] Anna E Steel, Julia H Coates, Alex R Hearn, and a Peter Klimley. Performance of an ultrasonic telemetry positioning system under varied environmental conditions. *Animal Biotelemetry*, 2(1):15, 2014.

[19] Wikipedia. Animal migration tracking. `<https://en.wikipedia.org/wiki/Animal_migration_tracking#Radio_tracking>`, 2015. [Online; accessed 5-November-2015].

[20] Yuan Zhaohui, Tan Rui, Xing Guoliang, Lu Chenyang, Chen Yixin, and Wang Jianping. Fast sensor placement algorithms for fusion-based target detection. *Proceedings - Real-Time Systems Symposium*, pages 103–112, 2008.