Depth-Sensitive Path Planning for Lakebed Coverage with an Autonomous Surface Vehicle

By

Justin Baum

Hunter Damron

Joshua Nelson

Andrew Smith

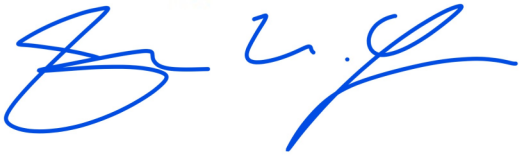
Submitted in Partial Fulfillment
Of the Requirements for
Graduation with Honors from the
South Carolina Honors College

May 2021

Approved:

Ioannis Rekleitis
Co-Director of Thesis

Joshua Cooper
Co-Director of Thesis



Steve Lynn
For South Carolina Honors College

Abstract

In this project, we aim to perform lakebed coverage using an autonomous surface vehicle (ASV), taking into account how the varying depth affects sensor coverage. Because the sensor footprint is broader in deeper waters, we use techniques from differential geometry to determine a set of sensor positions which can cover the whole space with minimal overlap. We also give an overview of our hardware setup which will be used to perform robotic coverage with an obstacle avoidance protocol.

Table of Contents

Abstract ii

Chapter 1 Introduction 1

Chapter 2 Proposed System 5

2.1 1-Dimensional Sensor Coverage 6

2.2 2-D Distortion Map with Volume Elements Proportional to Depth . . 11

2.3 Optimization using Divergence-Free Transformations 17

2.4 Hardware Overview 20

Chapter 3 Discussion 23

3.1 Variations on 2D Lakebed Coverage 23

3.2 Iterated Approach to Coverage of a Lakebed with Unknown Depth . . 24

3.3 Obstacle Detection and Avoidance 24

Chapter 4 Acknowledgements 26

Bibliography 27

Chapter 1

Introduction

Harmful algal blooms are becoming more prevalent globally and specifically in South Carolina over the last two to three decades (Lewitus et al. 2003). In this paper, we apply coverage path planning to autonomous surface vehicles in lakebeds for the purpose of detecting algae blooms. Coverage path planning is the task of generating a path for a sensor-laden robot to follow such that the entirety of the environment is observed. Common examples of coverage path planning include: autonomous robot vacuums (Yasutomi, Yamada, and Tsukamoto 1988), lawn mowing (Cao, Huang, and Hall 1988), and window cleaning (Farsi et al. 1994).

In one of the first works on coverage planning (Cao, Huang, and Hall 1988), Cao et al. carefully defined what a robot must do to successfully cover an environment. The requirements are as follows:

1. Robot must move through all the points in the target area covering it completely.
2. Robot must fill the region without overlapping paths.
3. Continuous and sequential operation without any repetition of paths is required.
4. Robot must avoid all obstacles.
5. Simple motion trajectories (e.g., straight lines or circles) should be used (for simplicity in control).
6. An "optimal" path is desired under available conditions.

Figure 1.1 Autonomous Surface Vehicle in Lake Murray, SC.

Enric Galceran notes in his recent survey on coverage path planning for robotics (Galceran and Carreras 2013), it is not always possible to optimize all of these criteria; thus, sometimes priority must be set when developing a coverage path planning algorithm. For certain applications, these criteria are straightforward and well-defined; however, in other applications, it is not as clear how to optimize Cao's coverage requirements.

One popular coverage technique is the boustrophedon pattern laid out by Howie Choset (Choset 2000) which mimics the path of an ox plowing a field. While this technique may not be truly optimal, it yields a very simple path which guarantees coverage. While many coverage planners assume the robot maintains a constant altitude (this is a reasonable assumption for drones), Galceran (Galceran and Carreras 2012) developed a system for underwater coverage which breaks the space into multiple parts by depth and performs boustrophedon coverage over each area independently to avoid excessive overlap in the covered area.

In this project, we build upon (Galceran and Carreras 2012) and use techniques from differential geometry to take full advantage of the fact that the sensor footprint

is broader in deeper waters. Following Moser's construction in (Moser 1965), we transform the space in such a way that volume elements are proportional to the sensor width. We then use a conventional path planning technique like boustrophedon coverage which assume a constant depth and project the path back into the original space.

Path planning is a well-studied problem, in just boustrophedon decomposition there has been many strides. This mainly relies on transforming the problem into a graph that can then be modelled for a travelling salesperson problem. This is effectively still a very difficult problem (Choset 2000). These similar ideas have improved on previous works, such as "teach-in travel" from a vacuum cleaning robot (Yasutomi, Yamada, and Tsukamoto 1988), where the robot only had the concept of avoiding obstacles. This brings on the next issue, whether the environment is known, unknown, or limited in knowledge. Work has been done on unknown environment strategies with boustrophedon, where the robot learns the environment in real time (Cao, Huang, and Hall 1988).

In 2014, Xu et al. (Xu, Viriyasuthee, and Rekleitis 2014) improved the boustrophedon coverage for aerial applications by ensuring the robot can return to its starting location without overlap. They also showed that this technique is asymptotically optimal, i.e. the path length decreases asymptotically to the shortest possible as the area of the space to cover increases with respect to the sensor footprint. More recently, boustrophedon coverage was adapted for vehicles operating under Dubins constraints (Lewis et al. 2017) and for multi-robot coverage (Karapetyan et al. 2017).

When this problem is viewed in the scope of traversing a lake bed or seafloor, you have additional constraints. One such is the scopes of the sensors. In shallower areas your ability to scope the bottom of the body of water becomes more restricted. In deeper areas, the scope widens. In (Galceran and Carreras 2012), they use the smallest possible scope and then boustrophedon decomposition. By segmenting the

area into smaller components, you can run boustrophedon on each component and end up with complete coverage. However as Galceran points out, this is inefficient as you can have overlap of coverage from each back and forth movement of the robot. The idea however, of segmenting the area into components and then using the smallest possible scope is efficient as the areas become infinitesimally small. By introducing differential geometry (Moser 1965), this can lead to optimal coverage.

Overall, the accomplishments of this project are: a proposed method for coverage path planning of a body of water with variable depths, novel mathematical model for sensor coverage with environment affected sensor width, and contributions to the documentation and ROS driver support of the PereptIn DragonFly Millimeter-Wave Radar.

Chapter 2 outlines the proposed method, including a warm-up of the 1D case in Section 2.1 and the details of our coverage technique for 2D in Section 2.2. In Section 2.3, we discuss how divergence-free transformations may be used to optimize our coverage technique over various parameters such as turning radius, and in Section 2.4, we describe the hardware setup for navigation and obstacle detection. Finally, we conclude with Chapter 3 which outlines several possible extensions to our work.

Chapter 2

Proposed System

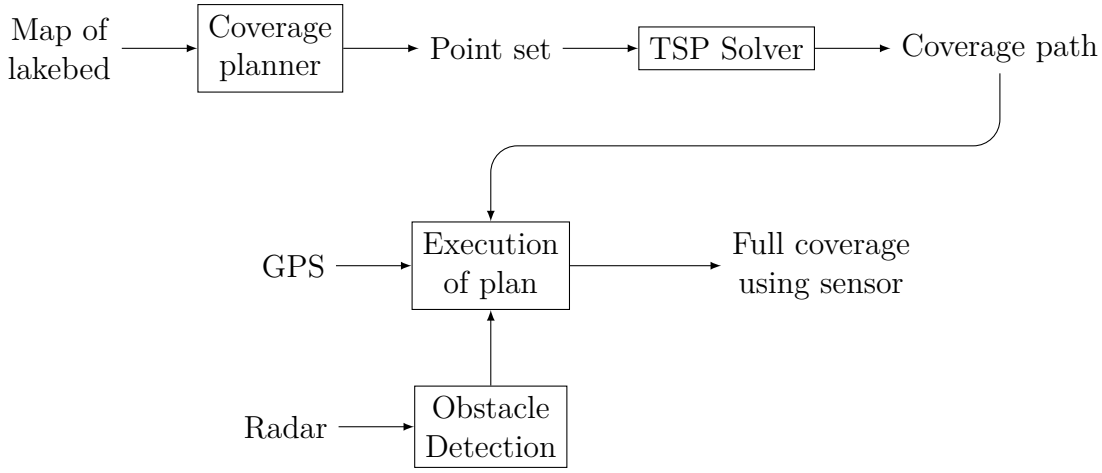


Figure 2.1 Overview of the full pipeline.

The proposed system in this paper assumes a complete knowledge of the depth map of the target body of water. We begin this system with a bathymetric map of the lake to be covered, as shown in Figure2.2. With knowledge of depth a priori, we construct an algorithm which is depth-sensitive. We propose to replace the navigable surface of the lake S with a surface $\sigma(S)$ which is locally-stretched in areas of high depth and is locally-scrunched in areas of low depth. This geometry $\sigma(S)$ is set up such that equally-spaced points pull back under σ^{-1} to sets which maximize coverage. After performing this sigma-transformation, followed by the inverse sigma-transformation, we will obtain a point set of sensor-cone locations. If the robot passes through each of these locations, it will view the portions of the lakebed covered by the sensor cone at the sensor-cone locations obtained from the inverse-sigma-

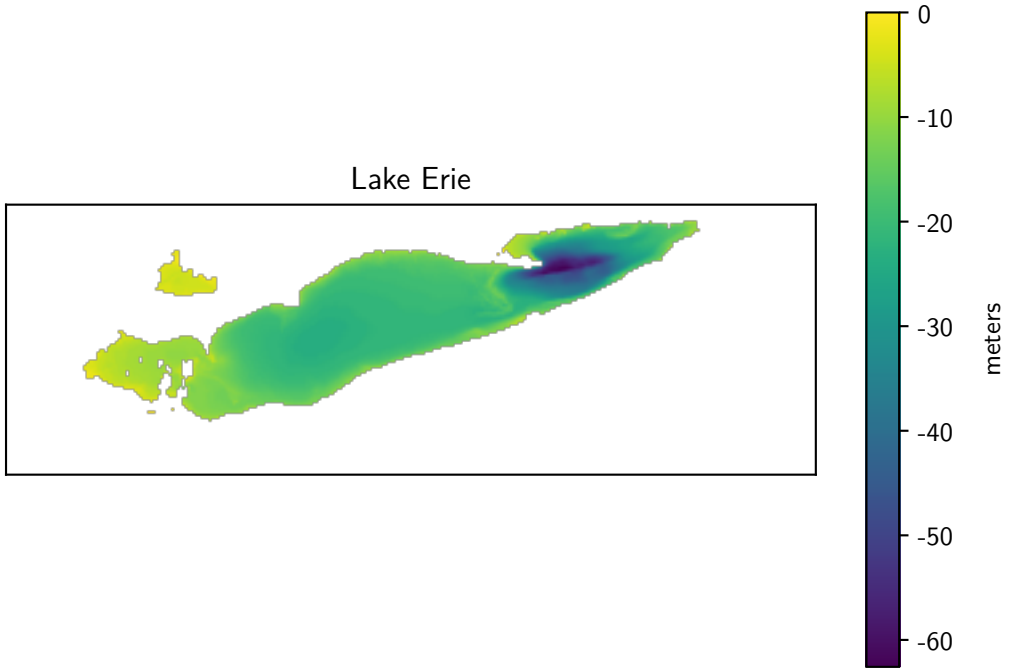


Figure 2.2 A sample depth map for Lake Erie

transformation. Thus, a heuristic TSP is used to find a robot path through the sensor cones to perform coverage.

2.1 1-Dimensional Sensor Coverage

In order to progress toward the desired geometric transformation in $\mathbb{R}^2$, we first develop a simplification in $\mathbb{R}$. Given a *lakebed* in $\mathbb{R}$ we aim to find a minimal set of points on the *surface* of the lake from which a corresponding set of *sensor cones* cover the entire *lakebed*. We say a *lakebed* is a differentiable function $f : \mathbb{R} \to \mathbb{R}$ on the interval $[0, 1]$ such that $f < 0$. An example of a lakebed is shown in blue in Figure 2.3.

We define the *surface* of the lake as the dotted line at $y = 0$, and we define the *sensor cones* as the cones which extend from the point set on the surface covering a *viewable area* on the *lakebed*, shown in Figure 2.5. Figure 2.5 is an example of

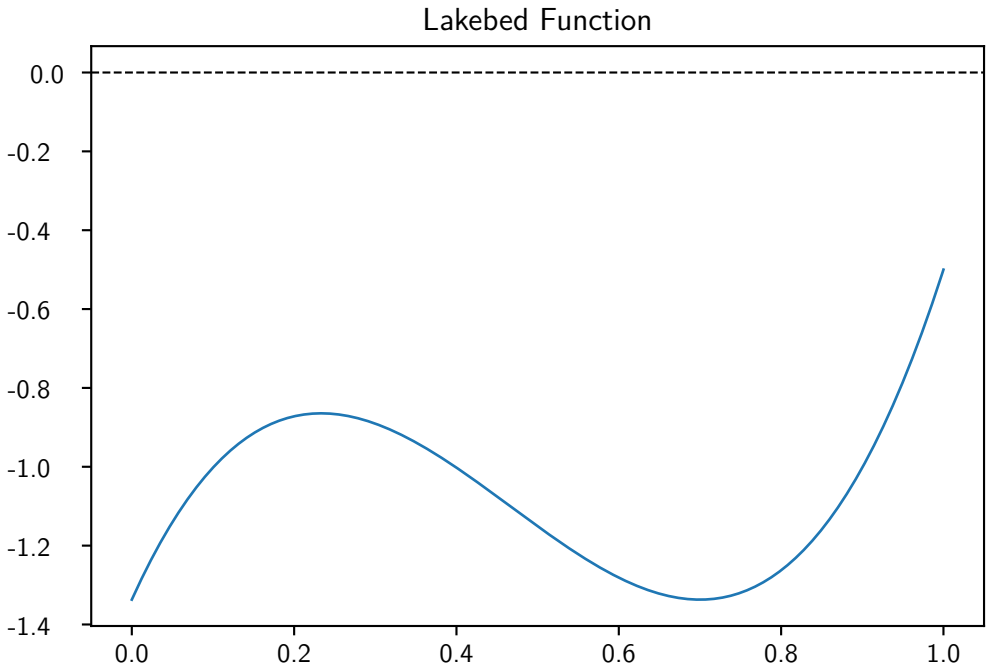


Figure 2.3 Idealized Lakebed in 1D

a fully-covered lakebed by 12 sensors which extend from a set of 12 points on the surface of the lake. A *minimal* point set from which sensor cones extend is one such that the removal of any point from the set no longer fully-covers the lakebed. We aim to develop a method to obtain a minimal point set which fully-covers a lakebed with knowledge of depth a priori.

Let S be the navigable surface of the lake, which is the line $y = 0$ in our representation. We aim to find a geometric transformation σ such that $\sigma(S)$ is a surface such that there exists an equally spaced point set T where the corresponding point set on the navigable surface $\sigma^{-1}(T)$ maximizes coverage of the lakebed. We craft this transformation σ exactly and approximately below.

Conjecture 1. *For any continuous* $f : [0, 1] \to [0, 1]$ *with* $f \leq 0$*, let*

$$\sigma(x) = \frac{1}{2n \tan \theta} \int_0^x \frac{dt}{f(t)}$$

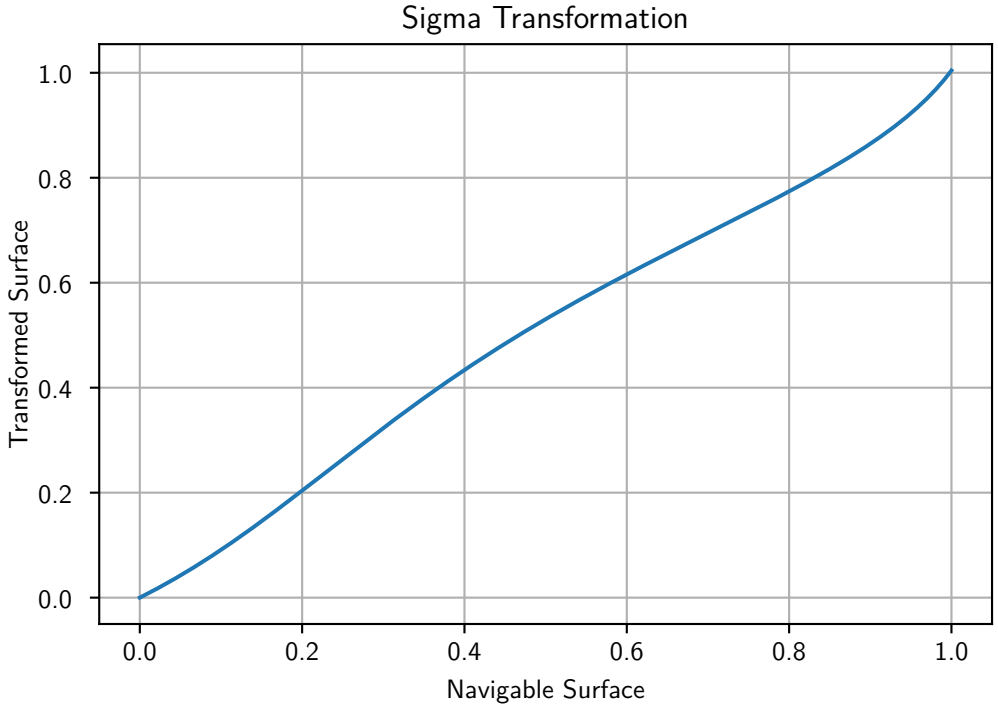


Figure 2.4 Transformation σ from the navigable surface to the transformed surface in 1D

and $n \geq \left\lceil \frac{1}{2\tan\theta} \int_0^1 \frac{dt}{f(t)} \right\rceil$ *be an integer. Then, for any set* $P \subset [-1/n, 1+1/n]$ *so that* $\min_{y>x} |x-y| \leq 1/n$ *for any* $x \in P \cup \{0,1\}$*, the set* $\{\sigma^{-1}(x) : x \in P\}$ *is covers the entire lakebed* f*.*

We now explain the conjectured geometric transformation from the beginning. For any point a in the navigable surface of the lake S, we assume that the observer has a downward-conical view of the lakebed of angular radius θ. Thus, we find that the condition that a point $P = (x, f(x))$ on the lakebed is observable from a point on the surface $a \in S$ is that the angle between the line descending vertically from the observer and the line connecting the observer and the point P is at most θ. This is equivalent to requiring that

$$\rho \leq |f(x)| \tan\theta \tag{2.1}$$

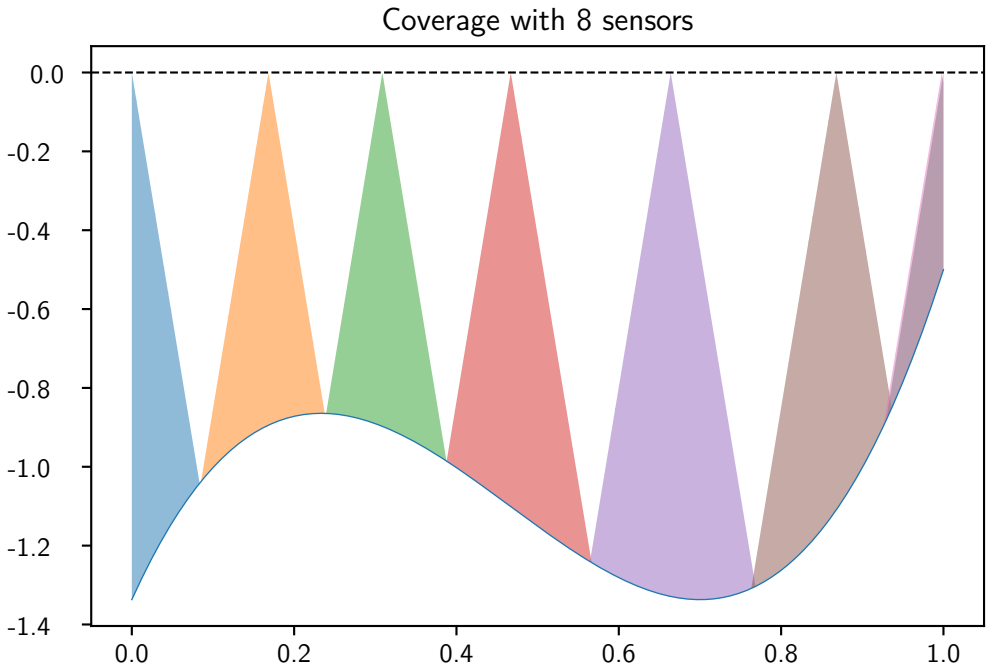


Figure 2.5 Fully covered lakebed with sensor cones shown.

where ρ denotes the distance $\sqrt{(x-a)^2}$ from the surface projection x of P to a. Now, applying 2.1, we seek to find the smallest point set $P \subset [0,1]$ such that

$$\bigcup_{a \in P} h_a^{-1}([0, \tan\theta]) = [0,1]$$

where

$$h_a(x) = \frac{|x-a|}{f(a)}.$$

One may search the space of possible point sets $P \subset [0,1]$; however, we propose an approximate solution. Consider taking a uniformly-spaced point set $P \subset \sigma([0,1])$ such that $|P| = n$ for some lakebed-transformation σ.

Thus,

$$\sigma\left(\sigma^{-1}(x) + \max\left\{f(x), f\left(x + \frac{1}{2n}\right)\right\}\tan\theta\right) - x \geq \frac{1}{2n}. \tag{2.2}$$

We can achieve equality in this expression asymptotically if

$$\sigma'(x) \geq \frac{1}{2nf(x)\tan\theta},$$

which is achieved exactly by

$$\sigma(x) = \frac{1}{2n\tan\theta}\int_0^x \frac{dt}{f(t)}.$$

Thus, we have the sigma-transformation for 1 dimension. Taking a uniformly-spaced point in the sigma geometry, T, we obtain sensor locations by taking the inverse of sigma, $\sigma^{-1}(T)$. Experimentally, the approximation for the number of sensor cones, n, seems to be too small; thus, we iterate the number of cones by 1 until complete coverage is obtained.

There are a few obvious generalizations from $\mathbb{R}$ to $\mathbb{R}^2$. First, the 1-D method can be used directly in slices of a 2-D geometry. For any 2-D navigable surface, partition the surface along a single axis into slices. For each of these slices, perform the 1-D sigma-transformation and coverage, which produces a point set along the slice on the navigable surface in 2 dimensions. Doing this for each slice, the method produces the same number of point sets as slices which can be a point set for the 2-D geometry. Increasing the number of slices increases the "resolution" of the surface.

Second, instead of embedding the 1-D method in a 2-D space, we can naturally extend the idea of the distortion map. In this case, we assume that the vehicle's sensor is a three-dimensional downward-facing cone which produces a round sensor shadow on the lakebed. This sensor cone is shown in Figure 2.6. In order to achieve local distortion proportional to depth in 2 dimensions, we replace integrals with squared double integrals as shown in the next section.

2.2 2-D Distortion Map with Volume Elements Proportional to Depth

In this section, we consider the more realistic situation of a 3D lakebed with sensors placed on the surface of the water. Given a depth map of a lakebed, our proposed method generates a set of points on the lake surface such that these sensors observe the entirety of the lake, assuming sensor visibility forms a cone growing downward from the sensor. Because this cone can extend wider when the surface of the lake is deeper, we aim to prioritize sensors in deeper waters. Figure 2.6 illustrates how the visibility cone of a sensor and the corresponding visible area on the lakebed.

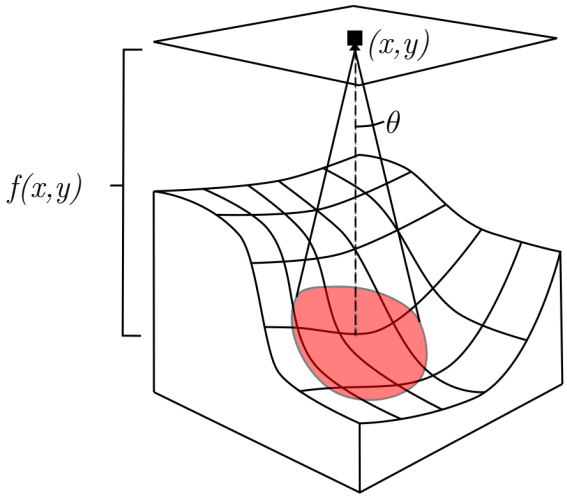


Figure 2.6 Illustration of 3D Sensor Visibility Cone

The lakebed is represented as a differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x, y)$ is the height of the lakebed at position (x, y) with the surface of the lake at height 0 and $f(x, y) < 0$ for all x, y. For simplicity, we will look at a square transect R of the lake and scale the parameters x, y such that the square has unit width. In order to avoid sensor shadows (i.e. the sensor cone intersecting the lake surface in more than one conic section), we assume the angle θ of the sensor cone

Similarly to Section 2.1, we aim to find a function $\sigma : \mathbb{R}^2 \to \mathbb{R}^2$ which distorts the

surface of the lake proportionally to the width of the sensor cone, or inversely proportionally to the depth $f(x, y)$. Specifically, this function must be a diffeomorphism, meaning it is a smooth transformation $R \to R$.

To achieve this, we use the fact that for a diffeomorphism $\sigma : R \to R$ defined by $\sigma(x, y) = (u(x, y), v(x, y))$ with Jacobian

$$\mathbf{J}_\sigma = \begin{bmatrix} u_x & u_y \\ v_x & v_y, \end{bmatrix}$$

the quantity $\det \mathbf{J}_\sigma$ measures the amount of expansion or contraction of σ at each point.

First, we define a normalized viewable area function

$$r(x, y) = \frac{f(x, y)^2}{\iint_R f(x, y)^2}$$

which we would like to be equal to $\det \mathbf{J}_\sigma$. Next, define

$$h_1(x) = \int_0^1 r(x, t)\,\mathrm{d}t \quad \text{and} \quad h_2(x, y) = \frac{r(x, y)}{h_1(x)}.$$

Finally, define

$$u(x, y) = \int_0^x h_1(s)\,\mathrm{d}s \quad \text{and} \quad v(x, y) = \int_0^y h_2(x, t)\,\mathrm{d}t.$$

Finally, $\sigma(x, y) = (u(x, y), v(x, y))$.

Now, since u does not depend on y, $u_y = 0$. Hence,

$$\det \mathbf{J}_\sigma = \begin{bmatrix} u_x & u_y \\ v_x & v_y \end{bmatrix} = \begin{bmatrix} u_x & 0 \\ v_x & v_y \end{bmatrix} = u_x \cdot v_y = h_1 \cdot h_2 = h_1 \cdot \frac{r}{h_1} = r.$$

Therefore, σ is defined such that it expands and contracts the space proportionally to r. Finally, we must check that the boundary of $\sigma(R)$ is identical to R. For any

$z \in [0,1]$, we get

$$\begin{aligned}
u(0,z) &= \int_0^0 h_1(s)\,\mathrm{d}s = 0 \\
u(1,z) &= \int_0^1 h_1(s)\,\mathrm{d}s = \int_0^1 \int_0^1 r(s,t)\,\mathrm{d}t\,\mathrm{d}s = 1 \\
v(z,0) &= \int_0^0 h_2(z,t)\,\mathrm{d}t = 0 \\
v(z,1) &= \int_0^1 h_2(z,t)\,\mathrm{d}t = \int_0^1 \frac{r(z,t)}{h_1(z)}\,\mathrm{d}t = \int_0^1 \frac{r(z,t)}{\int_0^1 r(z,t)\,\mathrm{d}t}\,\mathrm{d}t = 1.
\end{aligned}$$

Therefore, since u and v are nondecreasing in their parameters, σ maps R smoothly to itself and is, in fact, a diffeomorphism.

As a result, for any evenly spaced set Γ_n of n points in R, we expect $\sigma(\Gamma_n)$ to approximately cover the entire lakebed with minimal overlap, for an appropriately chosen n.

What we have presented above is the theoretical basis for this direction of our work, but we will also present our implementation of the above technique as well as an evaluation of how it should be improved going forward.

Rather than working with the lakebed depth map directly, we begin with a simpler lakebed modeled by

$$f(x,y) = \frac{1}{2}\sin(5x+8y) - 1.$$

Figure 2.7 shows the 3D plot of this function. The distortion map σ is illustrated in Figure 2.8 and Figure 2.9 shows how σ transforms an evenly spaced point set. Finally, we show the area covered by $\sigma(\Gamma)$ for a sample point set Γ in Figure 2.10 and calculate that the setup achieves 88.5% coverage with 53.4% overcoverage. In Figure 2.11, we apply an approximate travelling salesperson (TSP) solver to generate a path that an autonomous boat could take to make all of the necessary sensor readings.

Unfortunately, we found this proposed method to be insufficient for providing near-optimal coverage in 2D because the result on a sample lakebed yielded both areas which are uncovered and areas which are significantly overcovered. Our hypothesis is that the necessary condition is not just that $\det \mathbf{J}_\sigma = r$ but that $\operatorname{diag} \mathbf{J}_\sigma = [r_x \quad r_h$

or some similar condition which involves a consistency in both x and y dimensions. Once a suitable σ is found, it would be necessary to determine an approximation for the number of points required, and this would yield a point set which covers the space nearly completely with minimal overlap. One proposed solution would be to find σ_1 such that $\mathbf{J}_{\sigma_1} = \sqrt{r}$ in the way we have demonstrated and then rotate the process by 90° for σ_2 – the final transformation would then be $\sigma = \sigma_2 \circ \sigma_1$. From here, an exponential binary search can be used to find the minimum number of points required for complete coverage.

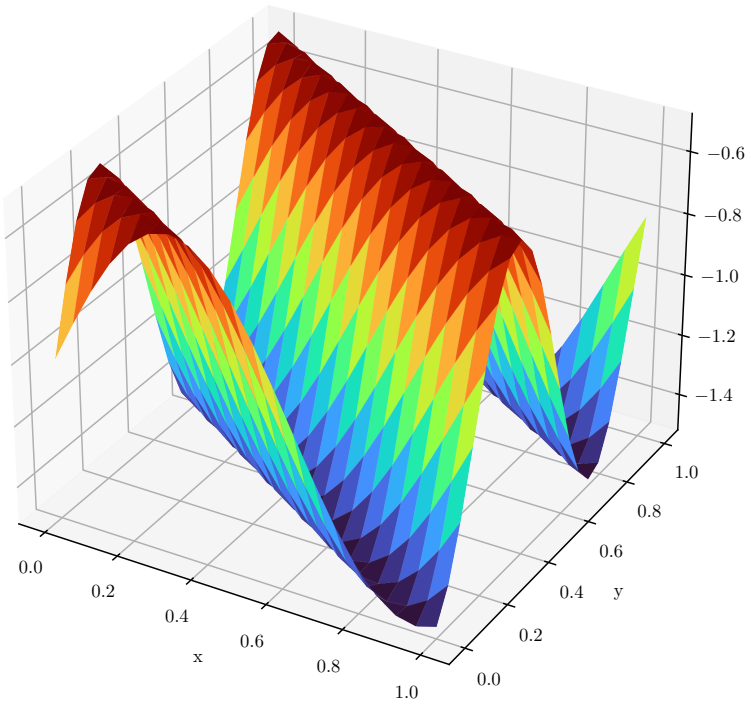


Figure 2.7 3D surface plot of sample lakebed

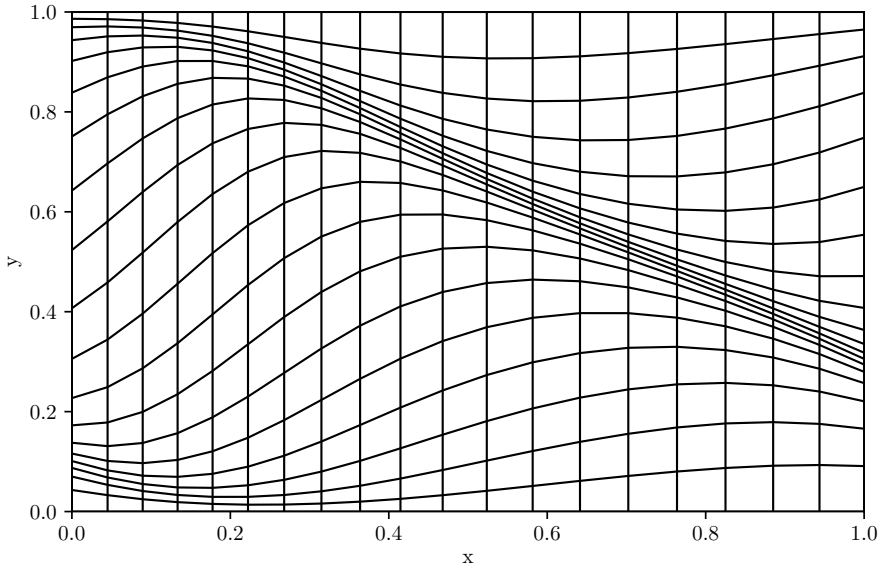
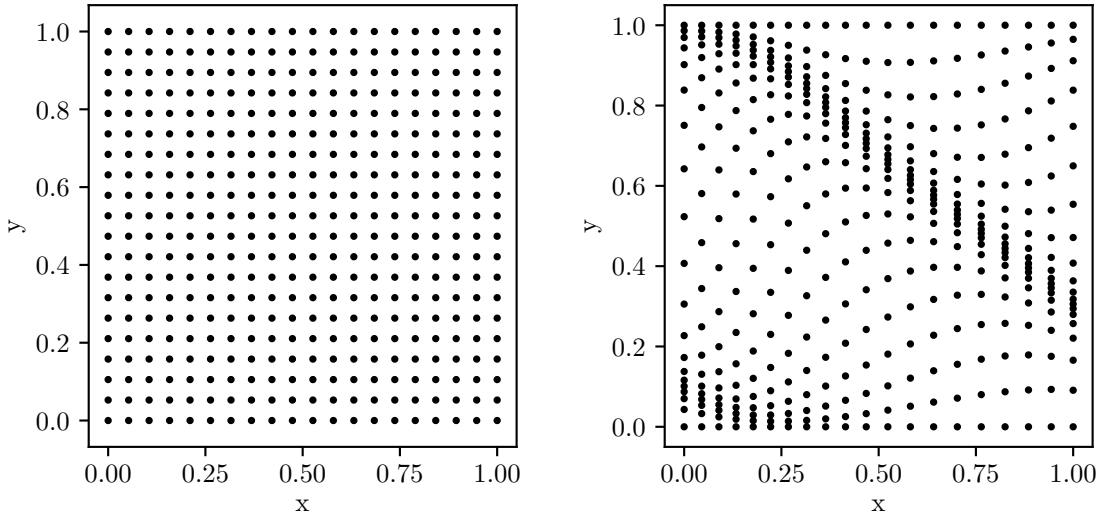


Figure 2.8 Transformation performed by σ



(a) Point set Γ (b) Transformed point set $\sigma(\Gamma)$

Figure 2.9 Transformation of an evenly spaced point set under σ.

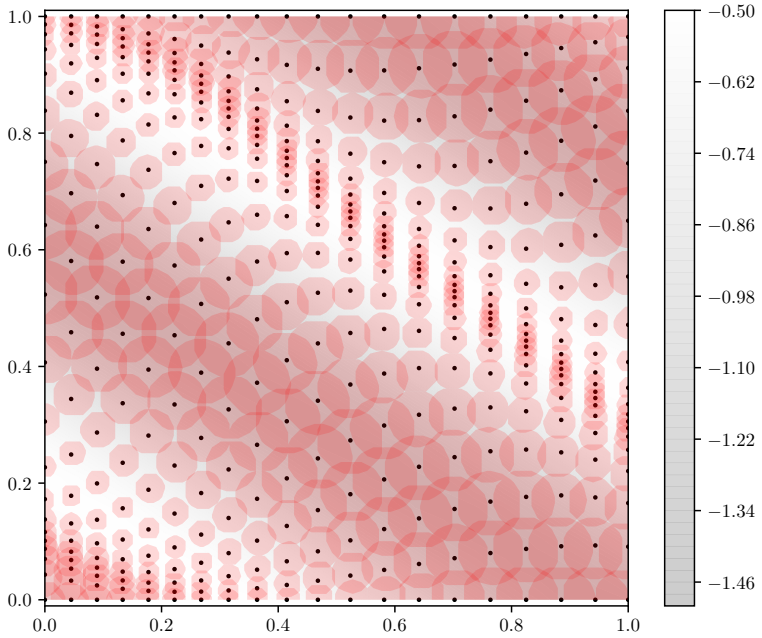


Figure 2.10 Map of area covered by $\sigma(\Gamma)$ with $\theta = 2.3°$. This configuration yields 88.5% coverage and 53.4% overcoverage.

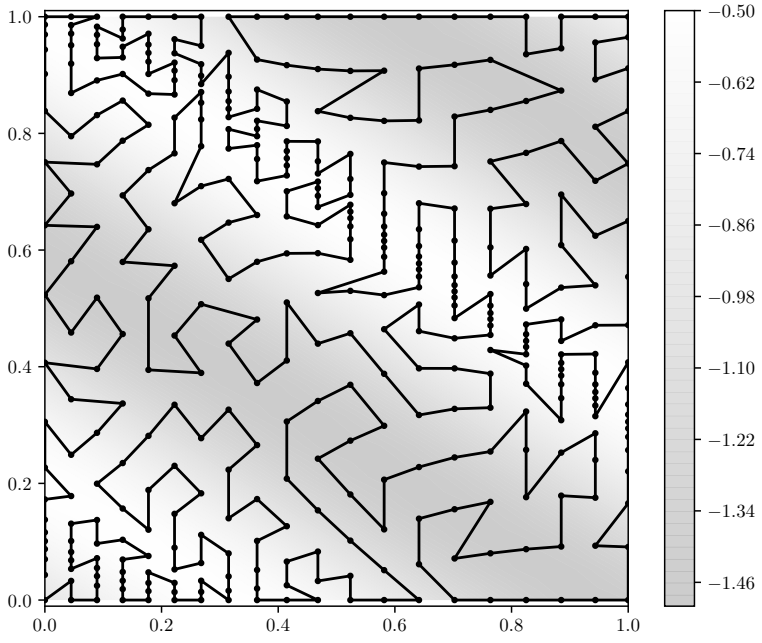


Figure 2.11 Robot path to perform coverage, generated using a heuristic TSP provided by the `python-tsp` package.

2.3 Optimization using Divergence-Free Transformations

Instead of the deterministic sigma-transformation, which transforms the geometry of a navigable surface of a lake into a space which is depth-sensitive, we consider the possibility of optimization over the space of all possible sigma-transformations. A *symplectomorphism* is a smooth map which preserves area and volume elements. An *automorphism* is a structure-preserving map from a space to itself. Thus, a *symplectic automorphism* is the transformation from a space to itself which preserves area and volume elements. We aim to find a symplectic automorphism which maximizes coverage. How can we find this symplectic automorphism? What space can we optimize over?

Let ψ, φ be two symplectic automorphisms which gives rise to the fixed measure which maximizes coverage. Due to J. Moser in 1965, $\psi \circ \varphi^{-1}$ is a symplectic automorphism which gives rise to the same fixed measure. Thus, given one symplectic automorphism which gives rise to such a fixed measure, you can obtain another such map by composing it with any symplectomorphism. Such a geometric transformation from the navigable surface to itself will produce a point set that maximizes coverage. Therefore, we propose to search the space of symplectomorphisms to find the desired map.

An *isotopy* is a smooth deformation from one map to another. The set of symplectomorphisms under the binary operation of composition is a group. Further, it is a *topological group*, meaning the structure of the group reflects the structure of the underlying manifold. Thus, you can obtain any other map by composing an initial measure-preserving map with another.

Each point on a given symplectomorphism has an initial direction of motion; thus, this gives rise to a vector field on the underlying manifold, known as a "sympelctic vector field" or a "Hamiltonian vector field". An isotopy in a Hamiltonian vector field is equivalent to a *flow*, which well-studied in fluid dynamics and physics. If there exists

smooth deformations between any two symplectomorphisms, then we can follow any given Hamiltonian flow, which arises from a symplectomorphism, and obtain any other symplectomorphism. Thus, the question is whether there are isotopies between all symplectomorphisms. Does every symplectomorphism arise from a Hamiltonian flow?

If we can begin with a symplectomorphism and computationally follow isotopies between maps, we can optimize over the coverage of each symplectomorphism. How can we follow an isotopy from a given symplectomorphism? Moreover, how can we obtain an initial symplectomorphism?

In general, every symplectomorphism does not arise from a Hamiltonian flow. However, due to Smale and Moser, if the underlying manifold of a symplectomorphism is compact, then every other symplectomorphism on that space does indeed arise from Hamiltonian flows (Smale 1959; Moser 1965). Since we are operating on the space of the unit square, $[0, 1] \times [0, 1]$, all symplectomorphisms on the space will arise from Hamiltonian flows.

In order to work with the algebra of symplectomorphisms, we first must generate a single symplectomorphism. We do this by observing that a divergence-free vector field arises from rotation the gradient of a scalar field by 90°. To get a random divergence-free vector field, we apply this process to random Perlin noise in the unit square. While this may not be perfectly random over all divergence-free vector fields, it gives us a nontrivial starting point for optimization. Once we have a divergence-free vector field, we can obtain a symplectomorphism by following the integral curves of the vector field for an arbitrary amount of time. In fact, the variability in the amount of time illustrates the continuous nature of the space of symplectomorphisms, which we hope to utilize for optimization. Figure 2.12 shows a sample unit square with Perlin noise used to give a symplectomorphism (any interval of the red line).

One issue we observed in this construction of a symplectomorphism is that it

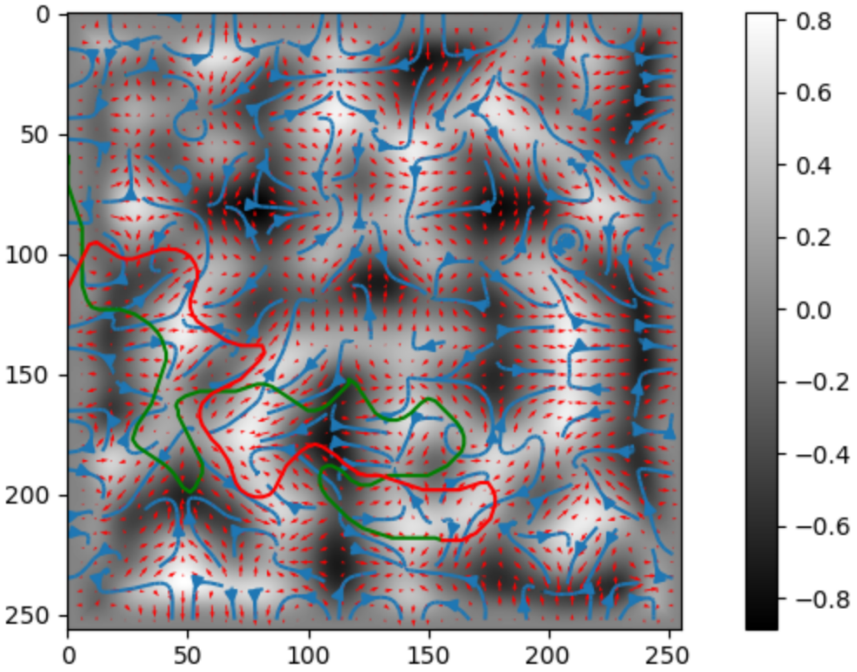


Figure 2.12 Computationally following flow of Hamiltonian vector field in python

tends to produce symplectomorphisms with repeating circular paths because the 90° rotation of the gradient gives paths which stay at a given height. To handle this issue, we propose using two scalar fields and alternating between the two over the course of an integral curve. This will prevent the path from falling into a repeating pattern.

Although we were not able to achieve this during the course of the project, our plan for optimization is to measure a gradient of the target criterion in the space of symplectomorphisms and use a modified form of gradient descent to find an optimal transformation. This symplectomorphism is then applied to the evenly spaced point set before transforming it under σ as described in Section 2.2. Because the symplectomorphism does not introduce any divergence, the resulting point set should provide the same quality of coverage as would be achieved with a grid of points. However, the benefit is that we have maximized some target value, such as maximum turning radius, in the process.

2.4 Hardware Overview

We had two hardware components we used in conjunction with the jetyaks. The first was a hand-held radar, the PereptIn DragonFly Millimeter-Wave Radar (77 GHz). The second was a GPS, the WIT JY-GPSIMU. It was surprisingly difficult to get both of these working. We used the Robot Operating System (ROS) framework for interoperability with other sensors on the boat. The radar came with a pre-compiled static archive file, `libPI_Radar.a`, which was not position independent and therefore did not compile on Ubuntu 20.04:

```
/usr/bin/ld: catkin_ws/src/afrl_radar/dependencies/libPI_Radar.a(Radar.cpp.o):
relocation R_X86_64_32 against symbol `_ZNSt12placeholders2_1E@@GLIBCXX_3.4.15'
can not be used when making a PIE object; recompile with -fPIE
```

As a result, we had to compile all parts of the radar on Ubuntu 16.04 instead. Once the radar compiled, we had to get it working. The format of the data was somewhat odd: each data read returned the angle, distance, and confidence of the measurement, but in an arbitrary order. We converted it to an in-order list of the distances, then published it to a ROS node. The ROS device driver for the radar is available online[1].

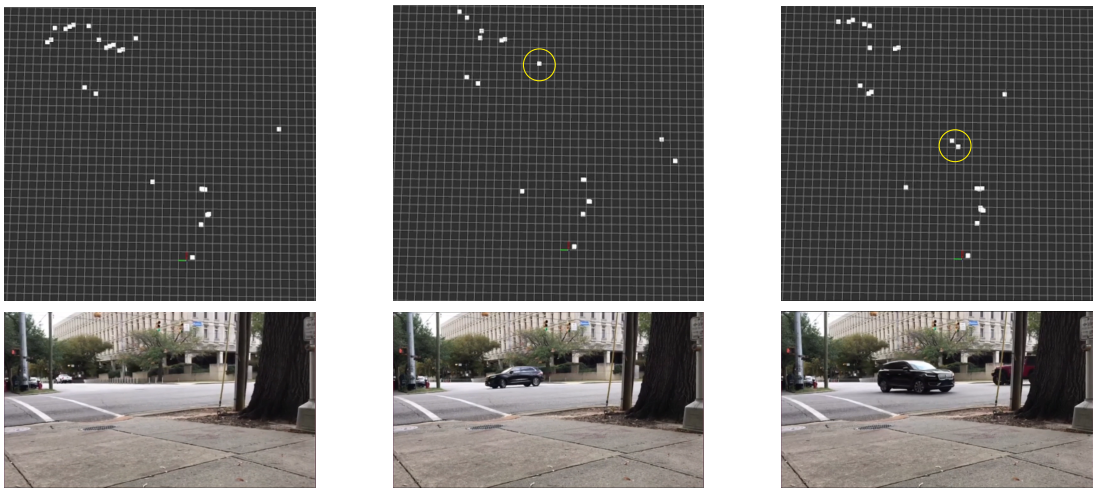
Next, we got the GPS working. The GPS came with software for measuring latitude and longitude and publishing it to a ROS node, but as it turned out, it didn't work. It measured the data correctly, but stored it in the wrong format: it reinterpreted the bits of an integer as a float. We fixed the bug and published the data to a ROS node. You can see the fixes[2] and video[3] on Github. Below are some examples generated from the data using RViz.

---

[1]Radar ROS driver available at `https://github.com/AutonomousFieldRoboticsLab/afrl_radar_ros`

[2]Fixed latitude and longitude calculations: `https://github.com/yowlings/wit_node/pull/4`

[3]The full video is available at `https://github.com/jyn514/Roaming-Robotics/raw/master/data/radar-cars2-trimmed.m4v`.



(a) A car approaches from outside the frame of the video.

(b) Car halfway through its u-turn. The radar data point is circled in yellow.

(c) The car after finishing its turn. Multiple points are logged because the car is moving quickly.

Figure 2.13 Moving car detected using radar.

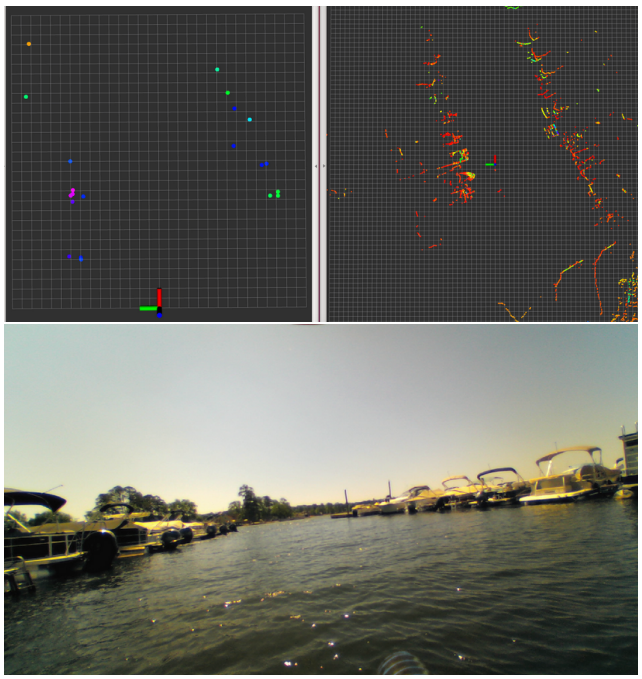


Figure 2.14 Jetyak leaving the dock. The top-left picture is the data collected by the radar. The top-right is reference data collected by a lidar sensor. The bottom picture is video from a mounted camera.

The car is difficult to see in still screenshots (Figure 2.13) because there is no relative motion visible. When watching the video[3], it is easier to make out which dots are the car and which are the ambient environment. Once the radar and GPS were working, we put them together on the Jetyak. Figure 2.14 shows the radar, lidar, and camera data from a deployment of the Jetyak. We used ROS and RViz to generate the visualization and align the radar data to the world frame using GPS[4].

[4]The code used to generate the visualization is available at `https://gist.github.com/jyn514/5f19114427dfff6cf7ae48906415176e`

Chapter 3

Discussion

3.1 Variations on 2D Lakebed Coverage

As we mentioned previously, there is still some work needed to determine a transformation σ which gives a point set with near-optimal coverage. In this section, we go a step further to discuss the future directions of such a technique. While such a σ would be guaranteed to give a near-optimal coverage, section 2.3 illustrates that there should be a whole algebra of suitable transformations. As seen in (Lewis et al. 2017), other factors may need to be considered when performing coverage by a robot, such as the maximum turn radius. Hence, we aim to perform some form of optimization over the space of symplectomorphisms to find the best point set with respect to another relevant quantity.

Although technically equivalent, another possible optimization technique is to vary the original evenly spaced point set transformed by σ. In this work we use a square grid, but there are other choices of evenly spaced or near evenly spaced point sets, such as a hexagonal grid or the Halton sequence. This technique, would likely be simpler to compute but possibly not as powerful.

Because the coverage is intended to be performed by a moving robot, another obvious direction is to generate a covering path directly rather than performing TSP on a covering point set. In such a formulation, we would consider the sensor footprint to be a line segment perpendicular to the robot's direction rather than a cone, and the coverage area of a full path is the integral of the scan lines over the length of the path.

Here, the σ transformation would need to be much more complex because the sensor footprint depends on the robot's direction in addition to position. Additionally, the "evenly spaced" criterion does not generalize well from a point set to a path. As a result, the choice of initial path would likely play a large role in determining the optimality of the path under σ.

3.2 Iterated Approach to Coverage of a Lakebed with Unknown Depth

Our proposed method currently assumes the lakebed contour is known, with the assumption that we are measuring something other than depth. However, if the goal is to measure the bathymetry of the lake, we propose an alternative method which uses the method iteratively until the full space is covered. In this technique, the robot performs an initial scan of the lakebed using boustrophedon coverage assuming the lake has constant depth equal to an estimate of the deepest part of the lake. From this, we get a bathymetry which is fully described at the deepest parts, but may have large gaps which are roughly estimated only by triangulation. Using this approximated depth map, we would use the previously proposed method to perform a more complete coverage of the thus far uncovered areas of the lake. After a sufficient number of iterations, the depth of the entire lakebed will have been recorded, and we expect this to be significantly more efficient than performing dense boustrophedon coverage over the whole lake. The primary drawback to this method is that the coverage plan must be generated in real time using the boat's limited computing power, rather than a more powerful computer offline as is currently required.

3.3 Obstacle Detection and Avoidance

The original goal of the radar and GPS was to have the Jetyak automatically detect and avoid obstacles along the coverage route. We have all the data necessary to do so

available, but not the code to process and react to the obstacles. Future work could have the Jetyak steer itself in a straight line as well as avoid obstacles it encounters along its route.

Chapter 4

Acknowledgements

We would like to thank Ibrahim Salman for his generous assistance both diagnosing hardware issues and deploying the radar and GPS on a Jetyak.

Bibliography

Cao, Z. L., Y. Huang, and E. L. Hall (1988). "Region filling operations with random obstacle avoidance for mobile robots". In: *Journal of Robotic Systems* 5.2, pp. 87–102. DOI: 10.1002/rob.4620050202.

Choset, H. (2000). "Coverage of Known Spaces: The Boustrophedon Cellular Decomposition". In: *Autonomous Robots* 9, pp. 247–253. DOI: 10.1023/A:1008958800904.

Farsi, M. et al. (1994). "Robot control system for window cleaning". In: *Proc. American Control Conference (ACC)*. Vol. 1, 994–995 vol.1. DOI: 10.1109/ACC.1994.751894.

Galceran, E. and M. Carreras (2012). "Efficient seabed coverage path planning for ASVs and AUVs". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 88–93. DOI: 10.1109/IROS.2012.6385553.

— (2013). "A survey on coverage path planning for robotics". In: *Robotics and Autonomous Systems* 61.12, pp. 1258–1276. DOI: 10.1016/j.robot.2013.09.004.

Karapetyan, N. et al. (2017). "Efficient multi-robot coverage of a known environment". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1846–1852. DOI: 10.1109/IROS.2017.8206000.

Lewis, J. S. et al. (2017). "Semi-boustrophedon coverage with a dubins vehicle". In: *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5630–5637. DOI: 10.1109/IROS.2017.8206451.

Lewitus, A. et al. (2003). "Harmful Algal Blooms in South Carolina Residential and Golf Course Ponds". In: *Population and Environment* 24, pp. 387–413. DOI: 10.1023/A:1023642908116.

Moser, J. (1965). "On the volume elements on a manifold". In: *Transactions of the American Mathematical Society* 120, pp. 286–294. DOI: 10.1090/S0002-9947-1965-0182927-5.

Smale, Stephen (1959). "Diffeomorphisms of the 2-Sphere". In: *Proceedings of the American Mathematical Society* 10.4, pp. 621–626.

Xu, A., C. Viriyasuthee, and I. Rekleitis (2014). "Efficient complete coverage of a known arbitrary environment with applications to aerial operations". In: *Autonomous Robots* 36, pp. 365–381. DOI: 10.1007/s10514-013-9364-x.

Yasutomi, F., M. Yamada, and K. Tsukamoto (1988). "Cleaning robot control". In: *Proc. IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3, pp. 1839–1841. DOI: 10.1109/ROBOT.1988.12333.