Karl Eirik Aasen

Coverage Path Planning for Seabed Mapping using Autonomous Surface Vehicles

Master's thesis in Simulation and Visualization
Supervisors: Robin T. Bye & Ottar L. Osen
December 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Abstract

The topic in this thesis is about path-planning for seabed mapping utilizing an autonomous surface vehicle with a sonar. In order for the path to be considered efficient, the depth and the characteristics of the sonar needs to be taken into consideration. Mapping the seabed is important for safe sea traffic, increasing safety in expeditions, geologists for uncovering minerals and resources, archaeologists for finding sunken treasures or old civilizations, and more. A proposed solution has been implemented, which is based on the approach by Manda et al. (2016). In this approach the complete survey path is gradually built up on a lap by lap basis. The proposed solution is able to gradually build up the complete survey path by placing straight line survey lines with constant or adaptive spacing. Or an adaptive survey path instead of straight lines. The proposed solution is applied to two different cases. The first case being in an area outside the coastline of Ørsta. The second case being in an area in the coastline between Hareid and Hjørungavaag.

The approach that places straight survey lines with constant spacing is used as the baseline, and is considered to be the naive solution. The spacing is decided by the shallowest point in the area to be surveyed. The rest of the approaches is compared to the naive solution. In the two presented cases, the approaches that utilizes adaptive spacing performs a lot better in terms of reduction in overlap and shortening the complete survey path, while keeping the same amount of coverage.

The current implementation of the proposed solution has some issues in terms of stability and robustness. Several areas for improvement and fixes is presented.

Acknowledgements

This year has been tough for me due to health issues. This resulted in the thesis taking a lot more time than originally planned. First of all, I want to give a special thanks to the professionals that helped me throughout the year, and my friends that encouraged me to keep working when times were hard. Special thanks to Robin T. Bye who initially proposed the topic and being my supervisor together with Otter L. Osen. Also thanks to Sindre Fossen from Maritime Robotics for insight in the topic.

Preface

This master thesis is submitted as the final work of Master of Science degree at the Simulation and Visualisation program at the Norwegian University of Science and Technology (NTNU). The research and report was started during the final semester, spring 2019, but delayed and delivered winter 2019. It was done with having Maritime Robotics as a light partner. The aim of the thesis is to propose an solution for planning a coverage path for seabed mapping utilizing a sonar. For the path to be efficient, the depth of the environment and the characteristics of the sonar is explored and taken into consideration.

Table of Contents

List of Figures

Abbreviations

ASV	=	Autonomous Surface Vehicle
USV	=	Unmanned Surface Vehicle
AUV	=	Autonomous Underwater vehicle
FOV	=	Field of View
VCS	=	Vehicle Control Station
CPP	=	Coverage Path Planning
DP	=	Dynamic Positioning
COLREGS	=	International Regulations for Preventing Collisions at Sea
DP	=	Dynamic Positioning
TSP	=	Travelling Salesman Problem
CSP	=	Covering Salesman Problem
2D	=	Two-dimensional
3D	=	Three-dimensional
EA	=	Evolutionary Algorithms
GA	=	Genetic Algorithm
DOF	=	Degree-Of-Freedom
NOAA	=	National Oceanic and Atmospheric Administration
ROI	=	Region of Interest

Chapter 1

Introduction

1.1 Problem

This project is a collaboration with Maritime Robotics in Trondheim. The topic is about using an Autonomous Surface Vessel (ASV) for seabed mapping using sonar. The ASV must follow path that is considered safe and efficient within the scope of the problem and avoid obstacles and hazards.

The vehicle operates on the surface, which means that the distance between the vehicle and the seabed will vary depending on e.g. the geometry and the structure of the seabed. Without considering this, a planned path will include unnecessary overlap.

For it to efficient, the vehicle must follow a path on the surface that efficiently maps the seabed without overlap. This means that the properties of the sonar and the dynamics of the ASV must be taken into consideration. The field of view (FOV) of the sonar changes depending on the depth. Which means that the sonar can reach a wider area when the depth is large, and a narrow area when it is small. This can be seen in 1.1.

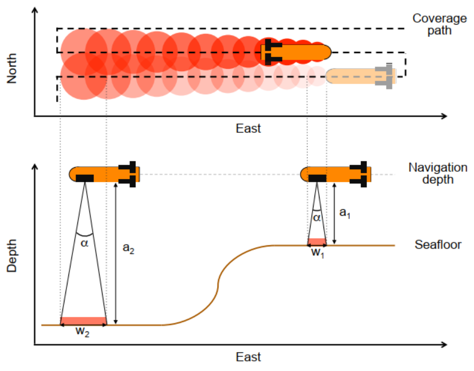


Figure 1.1: Overlap and changing FOV, retrieved from Galceran and Carreras (2012b)

To follow the planned path, the dynamics of the ASV must be taken into consideration, to ensure that the path is not counterproductive in the sense that it is impossible to follow or that it causes the ASV to do unnecessary and complex movements. A path that requires simple motion commands for the ASV to output would be the most practical.

Optimality and cost is dependent on the application and what it aims to accomplish. Knowing which factors to value over another and to optimize must be determined. In this case, several factors could be e.g. total distance, overlap, total coverage, fuel or energy consumption, time usage, safety, and more.

1.2 Motivation

Maritime Robotics is a company that believes it is important to follow the technological development. Solving problems like the one presented in this topic will contribute to better autonomous products. The company has something called the Vehicle Control Station (VCS), which is a software that allows planning and monitoring of missions and data acquisition. Implementing automatic path planning and better path planning algorithms will increase the value of the product by decreasing the amount of human labor and make it easier to plan missions. Seabed mapping is also a repetitive task that is ideal for automation.

More generally speaking, mapping the seabed is important for many reasons, Bernhard (2018); Frischkorn (2017); OER (2006). It is important for safe sea traffic, increasing safety in expeditions, geologists for uncovering minerals and resources, archaeologists for finding sunken treasures or old civilizations, and more. It can even help with revealing important information about the climate and determine weather patterns and ocean currents. By being able to do these mapping operations better and less expensive benefits many areas and sciences.

1.3 Scope

The scope of this thesis has been divided into three general main categories, as visualized in 1.2. These categories are Path Planning, Autonomous Surface Vehicle and Oceanography.

First category, and possibly the main component of this thesis is Coverage Path Planning (CPP). CPP is a variant of Path Planning, but it is more concerned about covering a surface given the agent's sensing abilities instead simply moving through it given a start and a goal position. In this category, the aspects of several CPP algorithms for seabed mapping will be reviewed to see if one or more fits the requirement to solve the problem. Changes to an already existing algorithm or method could also be proposed. The strategy will be to start with simple models, scenarios and cases and incrementally add complexity and details.

Second category is Autonomous Surface Vehicle. This is the type of agent that will be used together with a sonar for seabed mapping. In this category, the level of abstraction in terms of modelling the sonar and the dynamics of the ASV must be determined. The necessary specifications and properties required will be provided by Maritime Robotics.

Third category is Oceanography, which is science concerning the study of the ocean. Seabed or seafloor mapping falls under this. It could also very well fall under the topic of Hydrography, which specifically deals with surveying and charting bodies of water. Mapping is the process of creating a graphical representation of something, such as a landscape. In this case it is about mapping the seabed by measuring the water depth in a body of water. In other words, measuring the bathymetry.

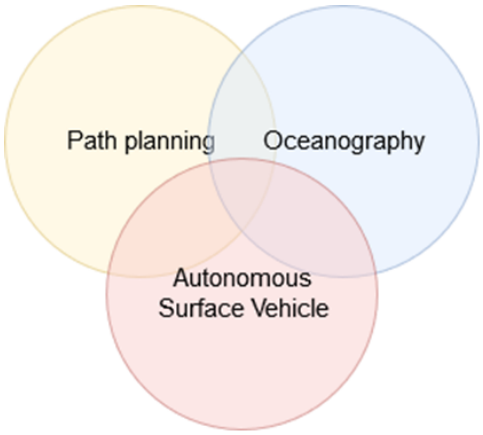


Figure 1.2: Scope of thesis

The research area of this thesis will be the middle piece, which is the intersection of all three main categories.

1.3.1 Levels of Abstraction

In this topic, the context of path planning is the one used in 2.1, which means the main focus is the process of constructing a path, and not motion planning, which is the process of creating the set of actions needed to follow the planned path. The dynamics of the ASV must be modelled, for the reasons mentioned in 2.1.1, but only go as far into the cybernetics as needed. The properties of the sonar will also be considered.

This means that topics such as localization, tracking, positioning (DP), external factors (wind, waves) and COLREGS safe sea traffic regulations is considered to be outside the scope. The main reason for this is because Maritime Robotics already has a working system, and at this point of time, it is assumed that these things have been considered in their system.

In this thesis, the objective is to propose a solution that could be used instead of manually planning a path by hand.

1.4 Objective

Create a solution for automatic path planning for efficient autonomous seabed mapping. More specifically, the solution should be modular in the sense that it is possible to e.g.

switch out the model of the ASV. Either to a different model or a more advanced one without having to do large changes. Other fitting components could also be modular as seen fit. The solution could also be flexible in way that different configurations for optimization could be chosen. E.g. prioritizing safety over distance and time usage or efficiency in terms of fuel consumption.

1.5 Research Questions

For this topic, the four following research questions has been established:

- Research Question 1: Can Coverage Path Planning be used to solve this problem?

The first question is about finding classic examples of CPP problems. Which applications have been solved as a CPP problem, and what kind of applications are they. Gain the knowledge of what a CPP problem is, what components it consists of and how to apply it to problem presented in this topic.

- Research Question 2: How can Coverage Path Planning be used to solve it?

In the second, it is more about finding similar work where CPP has been applied. Look at what they did, and what they used. Look into different CPP methods and algorithms for seabed mapping.

- Research Question 3: How can the proposed solution be compared to the current methods employed by Maritime Robotics?

For the third, it is about evaluation and validating whether the proposed solution is better. Finding out how can it be compared to the methods currently used by Maritime Robotics. Knowing what metrics to use, and if it can be measured in a quantitative way.

- Research Question 4: What are the benefits of using the proposed solutions? Advantages / disadvantages?

The fourth is about looking at the downsides and the upsides with the proposed solution. How successful is it? What are the gains of using it, are there any negatives at all? If there is, does the positive outweigh the negative?

1.6 Report Structure

The thesis is divided into the following chapters. Chapter 1 introduces the problem, the motivation, describes the scope and establishes the objective and the research questions. Chapter 2 includes the literature review and the theoretical background for each part of the scope. Specifically going more into the details of the main components. The chapter also includes related work within the area of research. Chapter 3 describes the methodology used. Chapter 4 is about case studies. Chapter 5 presents the results and discussion. Chapter 6 is the conclusion and future work.

Chapter 2

Literature Review

2.1 Path Planning

Path planning is an important component for autonomous mobile robotics. It gives the robot the ability to find a collision-free and an optimal path between a start and a goal position. Optimality of the path is dependent on the application. It could be in terms of distance, smoothness, energy consumption or any other metric present in the application Correll (2016).

Path planning, trajectory planning and motion planning are in many cases being used interchangeably. Lehtla (2008) defines path planning to be the constructed path from a point A to a point B, including stopping in defined path points. The planned path consists of numerous continuous motion trajectories that is determined by the trajectory planner. Motion planning refers to the steps required to follow the trajectory. This hierarchy can be seen in 2.1.

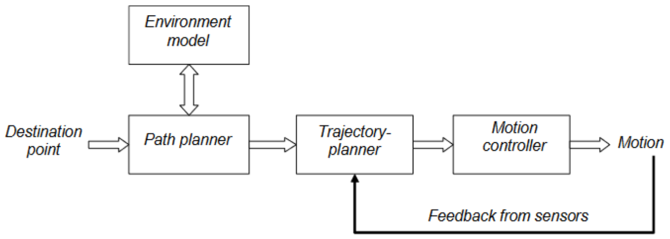


Figure 2.1: Hierarchical path planning, retrieved from Lehtla (2008)

2.1.1 Coverage Path Planning

Coverage Path Planning is a variant of Path Planning which aims to find a path that pass through all the points within an area of interest, Choset (2001). In other words, it is more concerned with covering a surface instead of simply finding the shortest path through it to reach a goal position. Choset believes that this is analogous to the covering salesperson problem. This problem is a variation of the travelling salesman problem, but instead of visiting each city, the salesman must visit the neighbourhood of each city. Nonetheless, for coverage, the salesman must pass over all points in the target environment, instead of simply just passing through all the neighbourhoods.

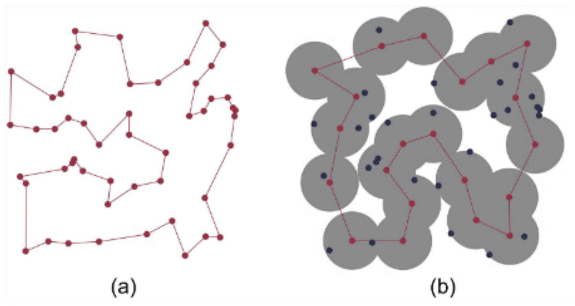


Figure 2.2: (a) TSP, (b) CSP. Red line and circles indicates optimal tours and visited nodes. Covered area is shown in grey, retrieved from Matsuura and Kimura (2017)

Galceran and Carreras (2012b) regards (Cao et al., 1998) to be one of the earliest works that defines the requirements for a robot to perform a coverage operation. These requirements defined by Cao et al. are as follows:

1. Must move through the whole area, and cover it completely.
2. Fill the region without overlapping paths.
3. Continuous and sequential operation without any repetition of paths.
4. Avoid all obstacles in the region.
5. Simple motion trajectories (e.g. straight lines and circles) should be used for simplicity.
6. An ”optimal” path is desired under the available conditions.

The author also mentions that it not always possible to satisfy all the requirements, so sometimes a priority must be considered.

Classification of Coverage Algorithms

Choset (2001) classified coverage path algorithms as either heuristic or complete dependant on whether the algorithm can generate a complete coverage of the work space. According to Choset, in early work, algorithms were dependant on heuristics which are rules-of-thumb that might work well, but does not guarantee a success. Complete algorithms is accomplished when the complete area is covered, and when there is no doubt whether it has been covered or not. This approach require more sensory and computational power.

Furthermore, the algorithms can be classified as either or off-line or on-line. According to Choset, this depends on how much prior information is available. Many classic approaches assumes that the environment is known in advance, but that assumption might be unrealistic in many cases. Instead, the agent is dependant on acquiring information about the environment through its sensors in real-time. In other words, in off-line planning, the environment is known in advance whereas for on-line it is not, and information is collected in real-time.

Criterion for CPPs

In Khan et al. (2017), three criterion for CPP is given attention, which are:

1. Environment decomposition technique.
2. Sweep direction.
3. Backtracking mechanism.
4. And to some extent, sweep motion and path smoothing.

According to Khan et al. these criteria are explained as: the decomposition method is the approach used to break down the environment into smaller regions for more efficient coverage. The sum of all sub-regions equals the total environment.

Sweep direction is used reduce to the amount of turning, which results in the trajectory mostly consisting of straight lines. The optimal sweeping direction is different for each sub-region, since the regions created from the decomposition method will have variations in size and shape.

The sequence in which order the agent should visit each sub-region is determined by the backtracking mechanism utilized.

Sweep motion determines which motions are used to perform coverage. The motions are basic with the notion that they are able to cover regions of any size and shape. These basic motions are also used as a base for more complex movements.

A path resulting path from a CPP algorithm is a linear piece-wise path that consist of only straight lines and sharp turns. In many cases, these paths are unrealistic to follow. To make them usable in applications, smoothness must be implemented. Smoothness is used to reduce turns and smooth sharp corners while keeping the path as short as possible.

Environment Decomposition Techniques

An environment consists of obstacles, free space and the robot itself. In many CPP problems, the first step is to break down the environment to separate the free space and the obstacles, Khan et al. (2017).

Random Coverage Method
Random path planning for coverage is a simple method for covering a surface. It is an common approach in commercial products, such as a robot vacuum cleaner or a lawnmover. In such an approach, the robot moves in a random direction until it meets an

obstacle, then it selects another direction and repeat the process. Sometimes it can be pre-programmed with a mechanism that allows for spiral motion or wall following behaviour Waanders (2011). According to Waanders, this method has its advantages: few sensors are needed and it do not require much computational power for path planning. Furthermore, there is no need for a map. Maximal coverage will be achieved regardless of environment shape, and whether it is dynamically changing or not. Nonetheless, the algorithm is inefficient and will contain a large amount of overlap.

Classic Exact Cellular Decomposition Methods
Cell-based decomposition methods decompose the free space in a region into non-overlapping cells. The cells can then be expressed as an adjacency graph, where the cells are the nodes, and edges represent a link between two adjacent cells, Galceran and Carreras (2013b). Coverage is proven to be complete if an algorithm make sure to visited every cell in the decomposition.

Trapezoidal decomposition is an approach that is an extension of exact cellular. As stated by Galceran and Carreras, this approach can generate a complete coverage path in planar polygonal spaces. In this approach, each cell is a trapezoid, which means that simple back-and-forth motion can be used to cover each cell.

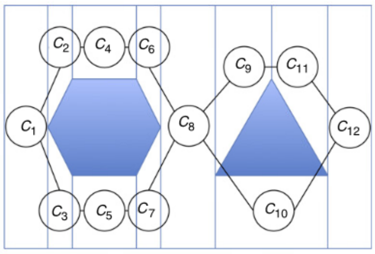


Figure 2.3: Trapezoid decomposition with adjacency graph, retrieved from Galceran and Carreras (2013b)

Boustrophedon decomposition is a further improvement of the trapezoid decomposition approach. According to Galceran and Carreras, one of the drawbacks of the trapezoid approach is that it generates many unnecessary cells that can be merged, since non-convex shapes can also be covered with simple motions, which can be seen in 2.4. This approach was proposed by Choset and Pignon (1997).

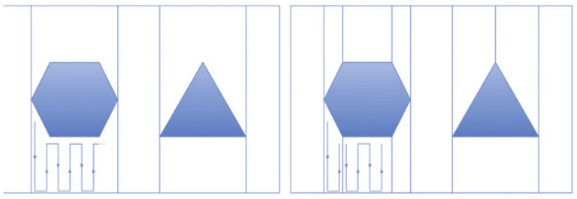


Figure 2.4: Trapezoid (left) versus Boustrophedon (right), retrieved from Galceran and Carreras (2013b)

Morse-based cellular decomposition
Morse decomposition is a generalization of the Boustrophedon approach by Acar et al. (2002). The Morse approach has the advantage of being able to handle any non-polygonal obstacles, and can be applied to any n-dimensional space. The different cell shapes can be obtained depending on which Morse function is chosen, which means that the Boustrophedon approach is a special case of the Morse approach, Galceran and Carreras (2013b).

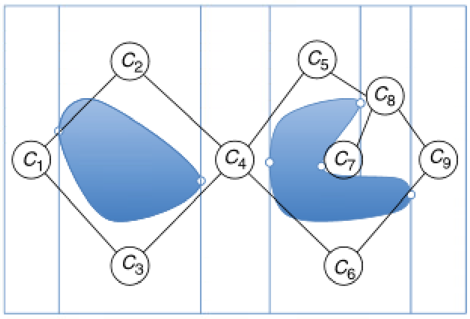


Figure 2.5: Morse decomposition with adjacency graph, retrieved from Galceran and Carreras (2013b)

Grid-Based Methods
In Grid-based methods, the environment is decomposed into a uniform grid cells, this was method proposed by Moravec and Elfes (1985). Each cell has a corresponding value that determines whether it is occupied or not. The shape of the cell is typically squares, but other shapes can be used as well. Completeness of the method is dependant on the resolution of the grid map. One downside with this method is the amount of memory required, Galceran and Carreras (2013b).

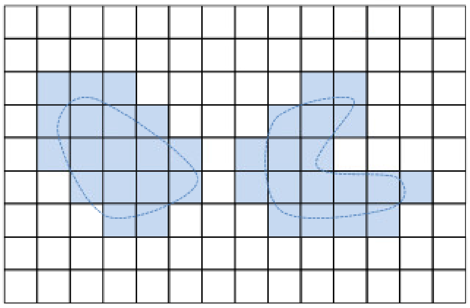


Figure 2.6: Grid map, retrieved from Galceran and Carreras (2013b)

Backtracking Methods

A common approach to determine the sequence of transits between the sub-regions in a decomposed environment, is by solving it as a TSP problem , Khan et al. (2017). This is because a decomposed environment can be expressed as an adjacency graph.

Greedy Algorithms
These algorithms builds up a solution to the problem by choosing the next choice that offers the most obvious immediate benefit, Khan et al. (2017). It is an algorithm that always make the most optimal local choice, in hope it will lead to the global optimal solution. Khan et al. proceeds to list frequently used search algorithms for sequence optimization.

Dijkstra's Algorithm, which is a graph based greedy algorithm that is commonly used to solve single source shortest path problems. It works by visiting nodes starting from a source, and then moves towards the goal by assigning distance cost to each neighbouring node. Then the node with the least cost is selected until the goal node has been reached.

A* Algorithm is an extension to the Dijkstra's algorithm. The algorithm uses a value that is the sum of the cost and heuristic function. It chooses neighbouring nodes dependant on the value, which will help avoid unnecessary exploration.

D* Algorithm is an optimal and efficient algorithm for path planning. It is able to handle dynamic unknown environments. It works by making an assumption that there are no obstacles, and start traversing from the source. When an obstacle is detected, the information of the environment is updated together with the calculation of the shortest path.

Theta* Algorithm is an variant of A* for any angle path planning. It that has the flexibility to propagate information along grid edges without constraining their paths to grid edges.

Evolutionary Algorithms
These algorithms are based on the principle of natural evolution. In a EA, a set of candidate solution is selected by applying a fitness function. An optimal solution is found by converging from a initial state to a global optimum by using a fitness function, (Khan et al., 2017). Khan et al. lists some of the most common used in CPP.

Genetic Algorithm is a heuristic based stochastic algorithm based on natural selection and mutation for solving optimization and search problems.

Ant Colony Optimization (ACO) is a probabilistic technique for solving computational problems. The algorithm is inspired from behaviour of ant colonies.

Particle Swarm Optimization (PSO) is a population based stochastic algorithm used for optimizing continuous nonlinear functions and perform parallel search. The algorithm is inspired from the social behaviour of organisms in swarms.

Sweep Motion Methods

In CPP, there are two standard motions that is commonly used. These motions are square spiral motion and the back-and-forth (boustrophedon) motion. These two basic motions are able to cover any region of any shape, Khan et al. (2017).

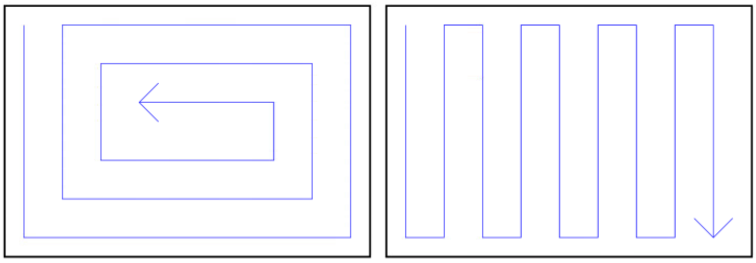


Figure 2.7: Square spiral (left), Back-and-forth (right), retrieved from Tommy (2017)

Path Smoothing Methods

Robots has constraints on properties such as: its curvature, velocity and acceleration. These constraints will restrict the movement of the robot while following a path created by a CPP algorithm. A path that is smooth enables the robot to follow the path without stopping, slowing down and reorienting itself on sharp turns. Many CPP algorithms generates paths that is in-efficient because it is unrealistic for a robot to follow. Maintaining smoothness throughout the coverage path increases efficiency, Khan et al. (2017). Khan et.al lists the two following classes for path smoothing that has been used in CPP.

The first class is the Graphical Method. In graphical methods, simple shapes like circles and lines are used to create a smooth path.

The second class is Function Based Methods. In function based methods, the path trajectory is represented by equations such as spirals, clothoids, spline based functions, and Bezier functions.

2.2 Autonomous Surface Vehicle

In literature, the term Autonomous Surface Vehicle and Unmanned Surface Vehicle is used interchangeable. In many cases a USV is defined to be a vehicle that does not have a operator to control it, but it can be remotely controlled. An ASV can also be without an operator, but it is able to operate without any intervention from a human. NOC (2018) defines ASVs as robotic vehicles that sit on the sea surface recording oceanographic data.

Areas of Use

The versatility of ASVs give them the ability to operate in a variety of missions, as listed by Vasconcelos (2015):

- Oceanographic measurements: bathymetry, water monitoring, salinity, currents, chemistry, earthquake prediction, hydrographic data, survey and structure inspection.
- Atmospheric measurements: surface winds, air temperature and humidity, sea surface temperature, hurricane path prediction and weather forecasting.
- Military missions: coastal/port surveillance, mine detection and reconnaissance, military training and security.

Advantages

Liu et al. (2016) states that USVs have a lot of advantages over manned systems in terms of the following reasons. Can perform longer and deal with hazardous missions, maintenance costs are lower and it is safer because of no personnel on-board, low weight gives increased maneuverability, greater potential in payload capacity and performs better in depth monitoring and sampling.

Underactuated

Most of the existing USVs are under-actuated because of the cost, and because full actuation is not practical, Liu et al. (2016). The author also states that the most common configuration used by USVs are mounting either two independant aft thrusters, or one main aft thruster and one rudder. This produces two inputs, propulsion force and yaw moment, while the USV is moving in an environment with three DOF.

2.2.1 OTTER

At this point of time, it is assumed that the solution or proposed methods are built towards the USV OTTER from Maritime Robotics. This USV is specially built for seabed mapping and monitoring of sheltered waters, MaritimeRobotics (2018).

2.3 Oceanography

Oceanography is the science that is about the study of the ocean. The field covers several topics, such as: marine life and ecosystems, ocean circulation, plate tectonics, geology of the seafloor, and the chemical and physical properties of the ocean, NOAA (2018b). There are many disciplines within the field, and according to NOAA's National Ocean Service, they are as follows.

- Biological oceanographers: studies plant and animals in the marine environment.
- Chemical oceanographers: studies the seawater, its processes and cycles, and the chemical interaction of the seawater.
- Geological oceanographers: explore the ocean floor and the processes that form its mountains, canyons and valleys.
- Physical oceanographers: study the conditions and the physical processes in the ocean, such as waves, currents, eddies, gyres and tides.

2.3.1 Hydrography

Hydrography is the science that measures and describes the physical features of bodies of water. Hydrographic data is collected by doing hydrographic surveys. Using multibeam echo sounders is the primary method of obtaining the data. This data is used for creating nautical maps and hydrographic models for marine geospatial products and services. By mapping out water depth, the shape of the seafloor and the coastline, the location of obstructions and physical feature of water bodies, is essential for keeping maritime traffic safe, NOAA (2018c).

2.3.2 Seabed Mapping

Seabed mapping and marine habitat mapping are loosely defined terms that can have different meaning depending on the application. Blyth-Skyrme (2011) defines the term seabed mapping to be as simple as bathymetry (depth) information or points representing observations of a single species or habitat across a defined area, and is used to describe activities involving the mapping of the seabed. Habitat mapping is used to refer to mapping of habitats which involves characterizing the benthic habitat across a given area of seafloor.

Blyth-Skyrme also states that mapping data has several uses, such as scientific investigation and understanding of ecosystems and monitoring of marine resources and habitats.

Reasons for producing maps could be; spatial management, selection of marine protected areas, management of marine resources, evaluation of management effective, survey planning, mapping or predicting species and habitat distribution.

Mapping Techniques

Pandian et al. (2009) states that mapping marine habitats is important for effective management of marine resources, which calls for more efficient methods. Pandian et al. list some of the more common techniques used, which are as follows.

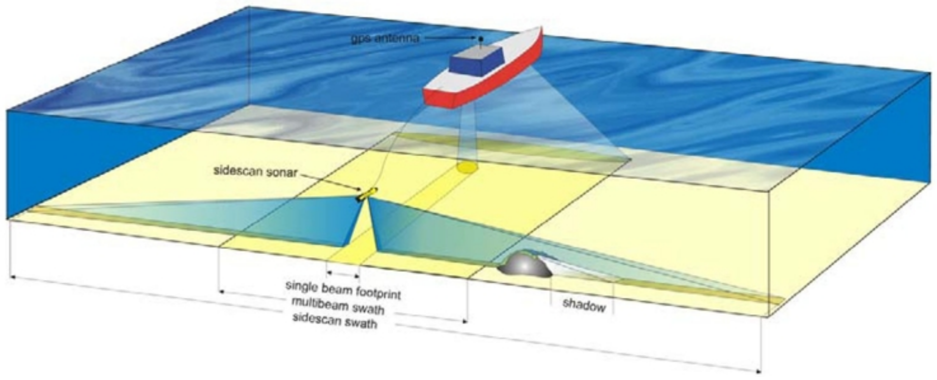


Figure 2.8: Differences between sonars, retrieved from[1]

Single-beam-echo-sounders (SBES) is used to obtain information about the reflective properties of the seabed. It sends a pulse of sound a given frequency that reflects from the seabed, which creates an echo that is picked up by the transducer.

Multi-beam-echo-sounders (MBES) functions in the same way, but sends several beams of sounds to cover a large fan-shaped area of the ocean floor instead of a small area.

Sidescan Sonar (SSS) is the leading tool for imaging the seafloor because of its good object detection and seafloor character discrimination. It provides high resolution, almost photographic quality images of the seafloor. The SS sonar has a dead-zone called the nadir.

Remote Sensing is a technique that is the combination of methods such as geomorphological segmentation, contextual editing and supervised classification.

In Boyd et al. (2006) many of the same techniques are listed, together with many more in different categories.

Bathymetric Charts

Bathymetric maps are topographic maps of the sea floor. Depth contours provide the size, shape and distribution of underwater features, NOAA (2018a).

2.4 Related Works

Galceran and Carreras (2012b)

In this work, the authors objective is to create a method that can create an efficient path to cover a 2.5D surface of interest on the seafloor. An ASV will be used follow the generated path by moving at a constant depth over the surface of interest. The vehicle carries a down-looking sensor that is used to image the seafloor from the navigating depth.

The method presented by Galceran and Carreras uses available environment information to create an efficient coverage path by minimizing overlap using variable-interlap spacing. The authors segments the target surface into regions of similar depth features using a K-means clustering algorithm for the initial segmentation. Furthermore they employ morphological operations, such as dilate and erode operations to smooth the border regions.

Furthermore, each region is solved as an individual CPP problem where the following steps are applied. First, Morse-based boustrophedon cell decomposition is applied to each region to divide them into sub-regions. Second, the sweep orientation is then set for each sub-region which is determined by the gradient of the underlying surface. Third, using the determined sweep direction, a boustrophedon path is generated to cover each sub-region where the inter-lap spacing is maximized on a lap-by-lap basis. Last, the individual paths in each region are combined to create the final coverage path. Result from the paper can be seen in 2.9.

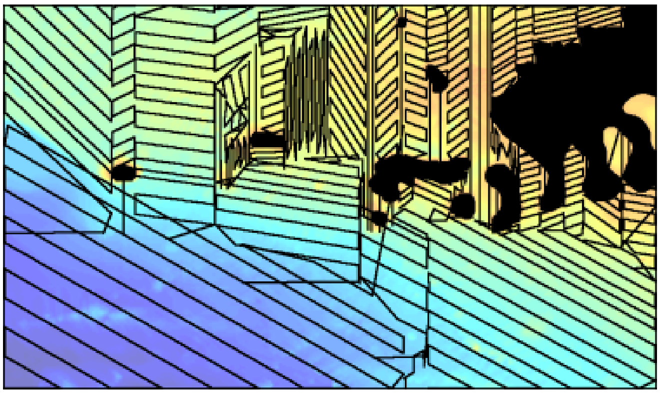


Figure 2.9: Resulting map with with coverage path, retrieved from Galceran and Carreras (2012b)

The proposed method is validated in simulation experiments where the authors uses a real-world bathymetric data set. The results showed an increase in path efficiency compared to a standard boustrophedon coverage path, with a path length of 10349.63m versus 15846.08m.

Manda et al. (2016)

In this work, the authors presents a method for comprehensive coverage path planning that does not require previous information about the survey area. It is able to adjust the survey lines dynamically based on the acquired data during the survey.

Manda et al. divides the path planning into two main steps: recording the swath data and planning the survey line. The second step consists of several sub-processes. The swath data is recorded while the ASV follows the survey line. It records the swath width from the sonar on both sides of the boat, together with vehicle position and heading at the time of the depth measurement. Methods for data refinement and decimation is employed to reduce processing times.

When a survey line is completed, the next one is planned by the following sub-processes. First, a offset path is formed from the subsequent swath. Second, the path is refined by removing areas where the path crosses over itself, to remove unnecessary turns and duplicate coverage. Third, sharp bends are removed from the path if they exceed a defined angle. Fourth, the path is fitted to the operation region, to ensure the path is not outside the region of interest. Last, eliminate planned points that falls within existing coverage, which is to prevent unnecessary duplication. The coverage of the operation is considered complete when all way points are eliminated. Result from the paper can be seen in 2.10.

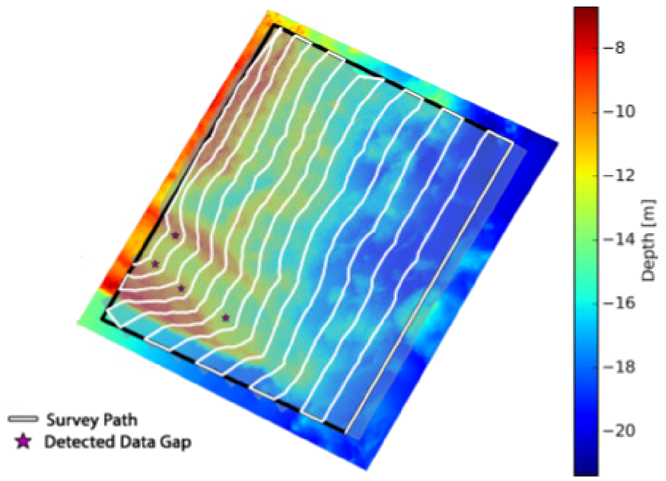


Figure 2.10: Simulation result on actual terrain, retrieved from Manda et al. (2016)

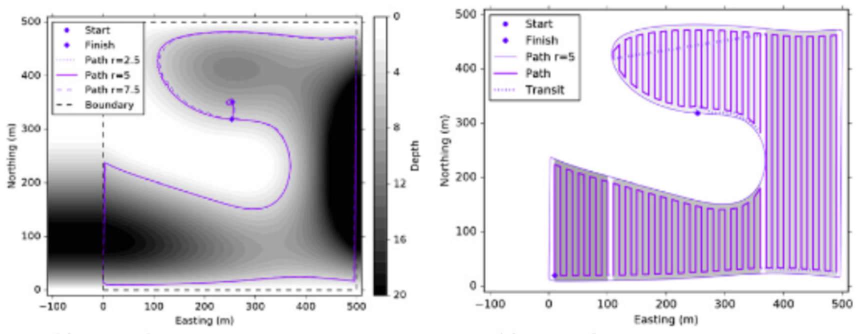
The proposed method for depth adaptive path planning shows promise in its ability to improve the hydrographic surveying workflow for ASVs. The algorithm works for both simple geometrically generated depth grids and existing survey data.

Williams and Wilson (2017)

In this work, the authors aim to provide a collection of algorithms which allows an ASV to operate in an unstructured environment with a minimal amount of information about the environment, and be able to autonomously explore the area and provide a map of the bathymetry.

Williams and Wilson divides the proposed algorithm into three components. The first component is the Gaussian Process (GP) which is used to build a model of the bathymetric contours during the mapping process. The GP is updated with data as it is collected by ASV. The second component is an algorithm that has been developed for following intersection of a bounding polygon and the depth contour predicted by the GP. For the third component, the authors has proposed an efficient Discrete Monotone Polygonal Partitioning (DMPP) of the resultant intersection polygon within which a lawn-mover pattern is planned for coverage.

The DMPP method was created to deal with the shortcomings of the Boustrophedon Cell Decomposition (BCD) method. According to the authors, these shortcomings was as follows: the BCD would at times create a track close to the cell boundary which is inefficient. It would also would make the transit paths between joining the cells inefficient, and it had lacking examples on how it could be applied to more complex boundaries.



(a) Contour/boundary path and bathymetry (b) Contour/boundary path and coverage

Figure 2.11: Simulation result of bathymetry and coverage path, retrieved from Williams and Wilson (2017)

The authors developed and implemented a collection of algorithms for autonomous path planning for coverage and creating a bathymetric map in an unknown area. The algorithms are tested both in simulation (2.11), and in the field.

Galceran and Carreras (2013a)

In this paper, the authors proposes a CPP method for inspection of 3D natural structures on the ocean floor charted as 2.5D bathymetric maps.

Galceran and Carreras explains that the proposed method is targeted for Autonomous Underwater Vehicles (AUVs). The planned path lay at a constant offset distance from the target surface, which allows for sensor imaging. The authors presents two problems with this kind of application. First, which is the frequent sudden depth changes that can result in inefficient motion. Second, that the vehicle requires time to adjust after a sudden change in relief, which can deteriorate the quality of the collected imaging data due to varying resolution. The proposed method accounts for these problems, and in high relief areas, it is able to create a coverage path which follows the constant-depth horizontal contours on the target surface.

For high-slope regions, the authors presents a slicing algorithm that plans coverage paths following constant-depth horizontal bathymetric contours at sequentially increasing depths. For planar regions, a rectilinear decomposition approach is proposed. This approach decomposes the space outside the high-slope areas into rectilinear cells. Then individual coverage paths are created within each cell. Each individual path within a cell consists of a lawnmower-type motion at a fixed offset distance above the target surface.

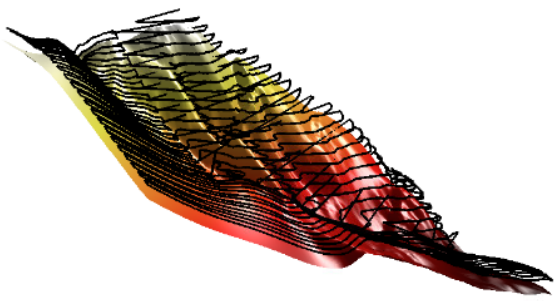


Figure 2.12: Simulation result on a high-slope region, retrieved from Galceran and Carreras (2013a)

The proposed method was tested on a real-world bathymetric data set of a lava tongue, which can be seen in 2.12.

Fang and Anstee (2010)

One of the main contribution in this paper by Fang and Anstee is the application of the generalized Voronoi diagram method for decomposing the survey area into sub-areas, and then calculating paths within these areas for incomplete sonar coverage. Furthermore, the transit between the sub-areas was solved as a TSP utilizing a genetic algorithm. A complete survey path generated can be seen in 2.13.

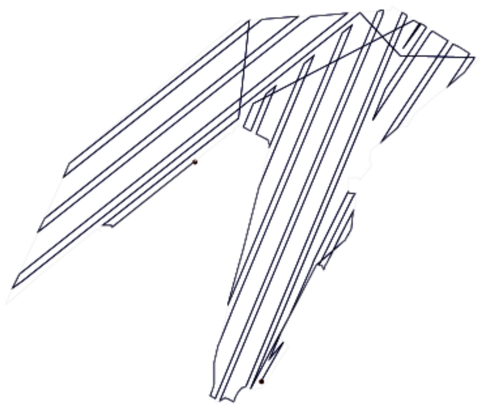


Figure 2.13: A complete survey path generated, retrieved from Fang and Anstee (2010)

Galceran and Carreras (2012a)

In this paper, Galceran and Carreras proposes a framework for generating coverage paths for marine habitat mapping. The framework utilizes the algorithms presented in previous work Galceran and Carreras (2012b) and Wong and MacDonald (2003). According to the authors, the framework consists of the following stages:

First-time mapping: a survey map of the area is created by using prior information, such as a bathymetry map or a nautical chart of the targeted area.

Determining region of interests (ROIs) and selecting starting locations: once a survey of the area is available, regions of habitat presence is selected, and regions with none are discarded. Starting points of the ROIs are determined.

Visiting all the ROIs: when the starting points for the ROIs has been selected, determine the order in which they should be visited.

On-line Coverage of ROIs: to cover the extent of each ROI on-line using imagery data, the sensor-based method previously referenced is used. It is used to detect the external and ”hole” boundaries in the ROI.

Moving to the next ROI: when the coverage of a ROI is completed, the vehicle moves to the next ROI.

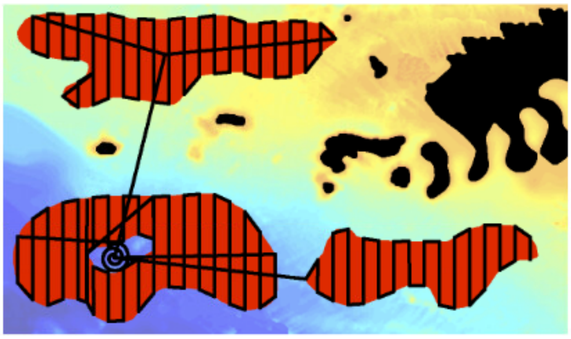


Figure 2.14: Coverage path covering ROIs, retrieved from Galceran and Carreras (2012a)

The proposed approach is tested on real-world bathymetric data sets. A result from the paper can be seen in 2.14.

Galceran et al. (2013)

The aim of the work is to propose a survey path planning method which minimizes uncertainty in terms of the robot’s position while also considering the area coverage performance.

Galceran et al. states that the algorithm avoid turns in the target area to maximize the quality of the sonar readings. It then operates on a parallel track basis by constructing a graph representing the parallel tracks required to cover the area, which is called the coverage graph. Then the survey path is planned in the two following steps:

1. The best possible order in which to cover the parallel track edges is found, which minimize the overall uncertainty along the path.

2. Crossing track edges are inserted in the path, if after tracking a parallel track, the uncertainty surpasses a given threshold.

To reduce uncertainty, for every point in in the bathymetry, the salience is computed, which is a Computer Vision tool. The proposed method was tested in simulations with real-world bathymetric data. The mapping performance of the proposed method was significantly better compared to a standard lawnmover-type method.

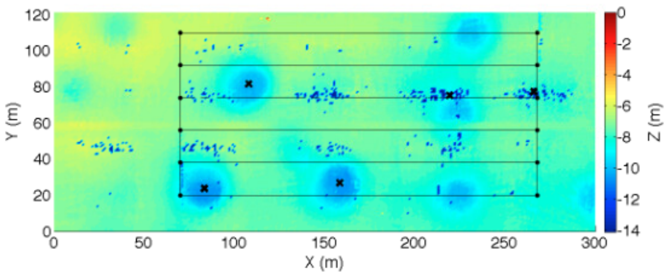


Figure 2.15: Coverage path, and key salient points (marked as "x"), retrieved from Galceran et al. (2013)

Jung et al. (2009)

Jung et al. presents an online coverage method for the exploration of unknown oceanic terrains using multiple AUVs. The authors proposes a new method for efficient coverage. The proposed method is the improvement of the Hert method that utilizes approximate cell decomposition as one of the exploration methods. The proposed method improves the shortcomings of the Hert method by addressing complex characteristics of unknown oceanic topography. The method is tested in simulations.

Zhu et al. (2018)

Zhu et al. proposes a method for complete coverage path planning of a AUV based on GBNN algorithm. The authors describes the proposed in the following steps. First, a 2D grid map is constructed by discretizing the work space of the UAV, and then a neural network that corresponds to the grid map is built. Second, dependent on the state of the grid map, the neutral network activity is constantly updated. Last, complete coverage path of the AUV is planned through the path planning strategy. The method is tested in simulations, and the results show that the method is quick, can handle both dynamic and static environments and avoid deadlocks.

Englot and Hover (2012)

Englot and Hover presents several contributions within sampling-based coverage path planning. The proposed algorithm is used in autonomous in-water ship hull inspection.

Englot and Hover (2016)

Englot and Hover presents a framework of optimal search that was developed by the US Navy to optimize operations, such as: search-and-rescue and mapping tasks. The framework provides path plans that takes the spatial and kinematic aspects of sensor performance into account

Kapctanovic et al. (2018)

Kaptchanovic et al. proposes the Basic Accordion Coverage Path Planner (BA-CPP) algorithm. The BA-CPP is an online side-scan sonar data-driven complete coverage path planning algorithm for large scale marine areas. It is tested extensively in various simulations.

More

The work in the following work has a small amount of relevancy and overlap Paull et al. (2013); Morin et al. (2013); Kim et al. (2014); Pratama et al. (2015), but is worth mentioning.

2.4.1 Critical Review of Main Authors

In the section of related work, I have mostly focused on work that in some form or another utilizes CPP in a marine environment. The six or so first related work listed has the most overlap in terms of the research area of this topic. The ones that comes after do not have as much overlap, but contain some elements that is related, but maybe not directly. Galceran and Carreras are two authors that seem to have contributed much within this area in the past. The work in Manda et al. has a lot of overlap in terms of this thesis, and seems to an approach that is worth taking inspiration or looking more into. William and Wilson also has some overlap, but that work is more concerned with building up a model of the mapped seabed, and improving the BCD method. All the other authors listed shares some overlap as well, but they are more for underwater applications.

2.4.2 Connecting Research Questions

In the first question, I asked whether CPP can be used to solve this problem, which is a yes. There are many applications utilizing CPP in the domains of ground, air and water. Several examples of related work that has used CPP and directly overlaps within the research area of this topic. I also have a fairly decent idea about how a CPP is defined and what components it consists of.

In the second question, I asked whether how I can use it, which is still a on-going process. There is numerous amount of different methods and algorithms, and the process of selection and reviewing is still in-progress.

The third and fourth question is still too early to answer, but looking at the third, there is defined several methods and metrics for measuring efficiency for CPP in the literature.

Chapter 3

Methodology

3.1 Workflow Methodology

For the implementation/practical part of the thesis, a workflow methodology similar to Scrum[1] was used. In Scrum, the work is segmented into sprints, and for each sprint you plan, design, build, test and review until a satisfactory result has been met. If it is not satisfactory, you go back to planning and refactor/iterate.

I proceeded to segment the practical part into loosely coupled modules that needs to be implemented to achieve the necessary results.

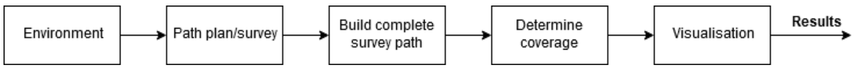


Figure 3.1: Main components as modules

Each general module is treated as a sprint.

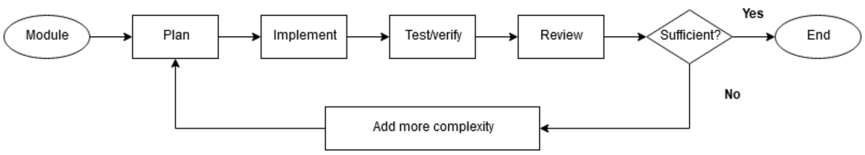


Figure 3.2: Sprint process for each module

First, the module is treated as a problem that is further simplified and divided into more manageable problems. Second, the smaller problems are implemented and put together. Third, it is tested and verified. Fourth, it is reviewed. Fifth, a conclusion is made whether if it is good enough, if not more complexity is added, otherwise the process is ended. When

[1] https://www.atlassian.com/agile/scrum

a process has ended for a module, it do not necessarily mean it is perfect or completed. Throughout the thesis a sprint could be re-opened.

3.1.1 Backup

For backup, DropBox is used. Dropbox synchronized every time a change is applied and saved on a file. A history for a file can also be accessed. Dropbox is sufficient for a single user, because there will not be any synchronization conflicts between users.

3.2 Tools

The programming language used for implementation of the algorithm is Python. In Python several notable libraries is used, such as numpy, matplotlib, pandas, cocos, scipy, sklearn and most important shapely.

Shapely is the main library which is a Python wrapper for the widely deployed GEOS[2] (Geometry Engien Open Source) library. This library makes it simple to work with set-theory, geometry, transformations and more. For the implementation I will mostly be working with simple Shapely Points, LineStrings and Polygons most of the time.

The programming environment is going to be JetBrains PyCharm for debugging and Jupyter Notebooks to run isolated sections of code and for fast prototyping and testing.

3.3 Choosing Algorithm Approach

Given the literature review performed in 2.4. The majority of the work reviewed is for land-based applications, where air and water being a minority. Furthermore a lot of water-based solutions is for underwater agents. Even though that is the case, a lot of it is generalized and can be applied any where. That said, I decided to take inspiration and base my approach on "Depth adaptive hydrographic survey behaviour for autonomous surface vessels" Manda et al. (2016). The reason for this is because this is the most relevant work I have reviewed and has a lot of overlap in terms of what I want to accomplish.

In the work by Manda et al., given an environment bounded by a simple polygon. An agent is used to gradually survey the area bounded by the polygon. The survey path is gradually being built by using the data recorded in the previous traversed path. In essence, the agent starts at one side of the boundary and then gradually sweeps the environment until the other side has been reached.

I will be following this approach as I interpret it. I would also possibly like to introduce some more complexity in the environment by inserting simple obstacles, and then proceed to solve that issue by simplifying the environment by using a method for environment decomposition.

[2]https://trac.osgeo.org/geos/

3.3.1 Further Abstractions - Implementation wise

It is assumed that sufficient information and data is known beforehand, so the main focus is offline path planning. Following this assumption, because information is know beforehand, it is also assumed that the chosen and mounted sonar is able to reach any depth of the area to be surveyed. This means the sonar range is always chosen to reach the deepest depths.

The main focus is to determine the theoretical coverage, so that the only concern is finding out where the outer sonar swath lands on the seabed, which is visualized in 3.3.

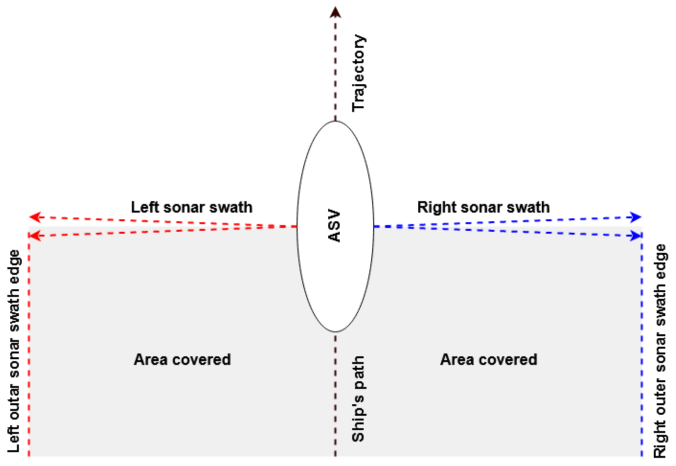


Figure 3.3: Theoretical coverage between outer sonar swath edges

The sonar model will be based on a MBES sonar, 2.3.2, so there will be multiple beams, but as mentioned above, the only concern is the outer beams. On a MBES sonar, there is also no dead-zone to consider compared to a SS sonar 2.3.2.

3.4 Data Set

The data set used in this thesis, is depth data that is public available at Geonorge[3]. The data consist of a terrain model that has a resolution of a 5m grid, and encompasses the area of Southern Sunnmore. It contains depth data from seabed surveying in areas such as: Fiskaa, Larsnes, Eiksund, Volda, Hareid and more.

3.4.1 Extracting and Reading the Data

When downloaded, the format of the data set is a .tiff which is an image format used to store raster images. This means the data set is more or less an image that cannot be displayed through normal image viewing software. Initially I attempted to load, read,

[3]https://kartkatalog.geonorge.no/metadata/eb2933a9-1699-40bf-b6b7-2adb66ccd714

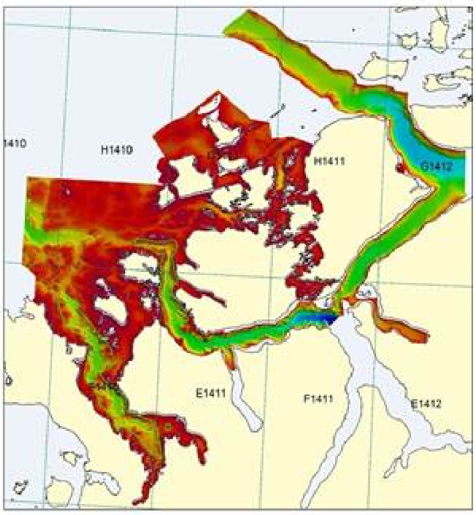


Figure 3.4: Data of area in data set - retrieved from GeoNorge

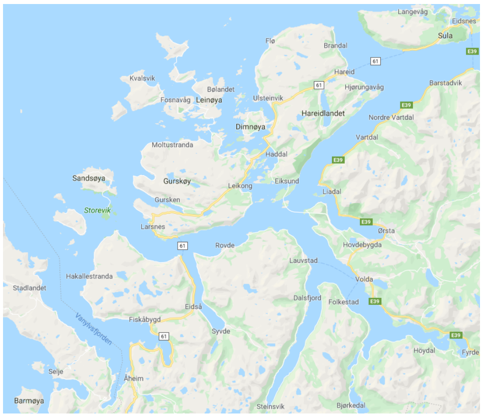


Figure 3.5: Data of the equivalent area in Google Maps - retrieved from Google Maps

display and convert the data set through code, but I ended with having issues either with libraries or memory running out because of the sheer size of the data. The solution was to do these things through ArcGIS[4], which is a tool for working with maps. With ArcGIS it was quite trivial to load, re-sample and export the data to another format, such as a xyz format that can be easily loaded into a Pandas DataFrame. With the data loaded into a data frame, it has the shape of 33209660x3, which is massive. In order to have something in a more workable state, snippets from the data is retrieved, which can then be further down sampled when necessary.

[4]https://www.arcgis.com/index.html

Converting data into grids

Snippets from the data set is furthermore made into grids. A grid is a structure that makes it a lot more trivial to work with, process and access the data in code. This is achieved with NumPy and SciPy in the following steps.

```
# 1-dimensional arrays extracted from data frame
x_data, y_data, z_data

# create linear spaced arrays from x_data and y_data
x_space = numpy.linspace(minimum_value_of_x, maximum_value_of_x,
length_of_x)
y_space = numpy.linspace(minimum_value_of_y, maximum_value_of_y,
length_of_y)

# create xy meshgrids
x_mesh, y_mesh = numpy.meshgrid(x_space, y_space)

# create z-meshgrid
z_mesh = scipy.interpolate.griddata((x_data, y_data), z_data, (x_mesh,
 y_mesh), interpolation_method)
```

Listing 3.1: Establishing z-grid

In order for the z-grid to be established. The lengths of the arrays need to be the same, and there needs to be sufficient data for the interpolation to work properly. Afterwards, it will result in x, y, z grids that are m x n grids where both m and n are the same length (according to the size of data). For the interpolation method, there are several choices, such as: 'nearest', 'linear' and 'cubic'. At this point of time, 'linear' is used.

```
x_mesh = [[0, 1, 2],     y_mesh = [[0, 0, 0],     z_mesh = [[-1, -1, -2],
          [0, 1, 2],               [1, 1, 1],               [-3, -2, -1],
          [0, 1, 2]]               [2, 2, 2]]               [-4, -3, -3]]
```

Listing 3.2: Example shape

3.4.2 Determining Survey Area Boundaries

In order to determine the survey area, the region of interest needs to selected. This is resolved by first filtering out region of interest from the z-grid given a depth threshold, e.g the agent cannot traverse the environment if it too shallow. This is achieved in the following way.

$$\text{survey\_area}(x, y)\begin{cases} z_{\text{mesh}(x,y)} & \text{if} \quad z_{\text{mesh}(x,y)} < \text{threshold} \\ 0 & \text{if} \quad z_{\text{mesh}(x,y)} > \text{threshold} \end{cases}$$

Figure 3.6: Filtering data set by threshold

Then secondly, represent the resulting z-grid as a polygon. This polygon is further simplified as a path covering a simple polygon is more trivial and easier to find compared to a complex shape. Which is done in the following way.

A matplotlib method is used to triangulate, and group together data. The data is drawn as filled contours.

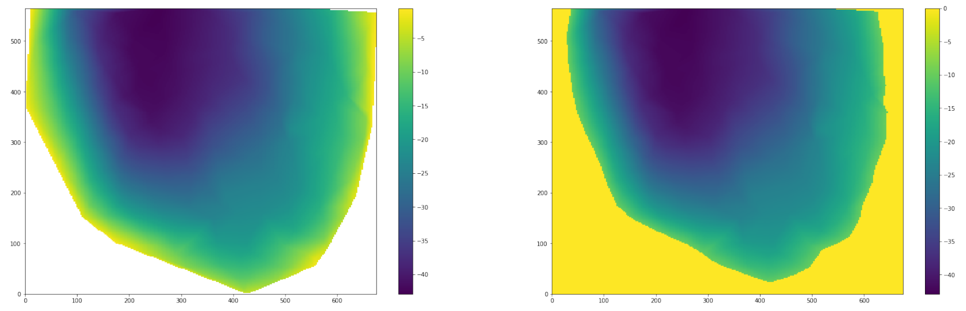


Figure 3.7: Unfiltered (left) versus filtered (right), threshold = -10

```
co = matplotlib.tricontourf(x, y z)
# this method returns a Triangulation object
contours = co.collections
# contours are retrieved from the object and converted into shapely
polygons
polygon = shapely.ops.unary_union(contours_as_shapely_polygons)
# the polygon is found by taking the union of all contours
simplified_polygon = polygon.simplify(threshold, preserve_topology)
# polygon is then simplified further given an appropriate threshold,
which then becomes the survey boundary
```

Listing 3.3: Establishing survey boundary

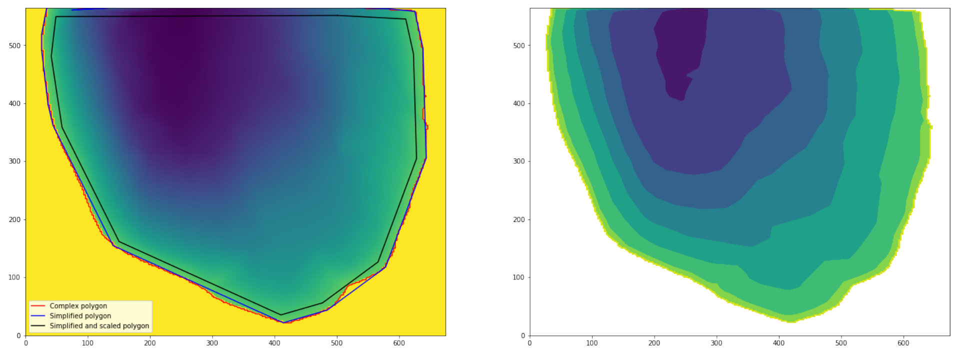


Figure 3.8: Survey boundary (left), depth data as filled contours (right)

The threshold has for simplifying the survey boundary to be chosen accordingly. The threshold determines how close vertices in the polygon has to be for them to be merged. The preserve topology parameter determines whether the general shape of the polygon should be kept.

3.4.3 Retrieving Depth Data from Grid

To retrieve depth data from the z-grid, approximation is used. The dimensions of the x and y-grid is limited, which means that the x and y-grid has do not have a corresponding value for every possible value. If that was the case, both would grids would be infinite long. For any given (x, y) position, the x, y-components are approximated to fit their corresponding grids. The approximated x will return a row, and the approximated y will return a column in the shape of indices. These indices are used to access the corresponding depth value of the approximated (x, y) value. This is trivial and achieved with numpy. The approach of using approximation could results in issues if the resolution of the data is not sufficient enough.

3.5 Surveying an Area

In the following sections, the algorithm is described in steps. It covers the process and steps for path-planning using straight survey lines with adaptive spacing, and the version that uses adaptive survey paths using average heading.

3.5.1 Path Planning using Straight Survey Lines

Determine direction line

Before anything happens, a direction line is determined. The direction line gives an indication about the general direction in which the depth increases from a given point. The direction in which the depth travels has to be considered to reduce turns and increase efficiency, Galceran and Carreras (2012b). When the direction line is found, it will be used to place the survey lines perpendicular to it.

In this case, the general direction in which the depth increases from a starting position is established by first doing a straightforward A* search on the z-grid. Which uses the depth as cost, and the euclidean distance as the heuristic. This results in a path that chooses to go where the depth is deepest until the goal state is found. The goal state (position) can either be input manually or be retrieved automatically from the z-grid.

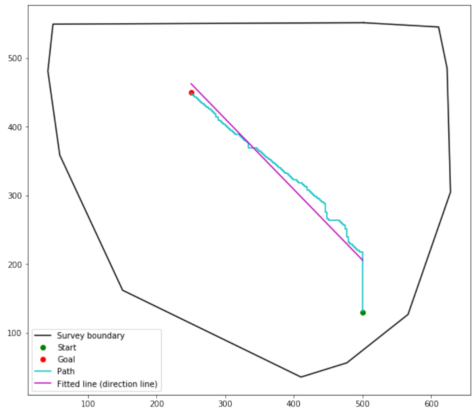


Figure 3.9: Path resulting search, and resulting direction line

In order to convert this path into a general direction line, an linear regression is applied to the path. The linear regression will attempt to fit a (prediction) line onto the path. The direction line is then furthermore bounded by the survey area, and the two points of intersection will be the two potential starting positions of the survey. In this case the potential starting positions will be directly on top of the survey boundary. To make sure they are both inside, a scaled down version of the survey boundary is used. So, in this case there is an outer and inner survey boundary. The direction line is then instead bounded by the inner survey boundary. The length of the survey lines is chosen based on the size of the outer survey boundary. The survey lines needs to be long enough to stretch over the outer survey boundary at any given position.

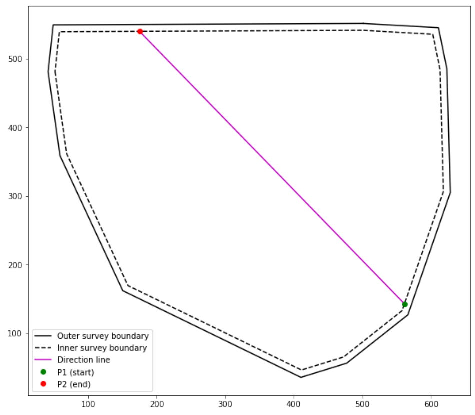


Figure 3.10: Extended and bounded direction line

Determine initial survey line

The two potential starting positions P1, P2 for the survey is determined by the direction line bounded by the inner survey boundary. If P1 is selected, the initial survey line will be positioned at P1, and all the succeeding lines will be positioned towards P2. If P2 is chosen as start position, it will go from P2 to P1. As mentioned previously, the direction line also determines the orientation of the survey lines. The placed survey lines will be placed perpendicular to the direction line.

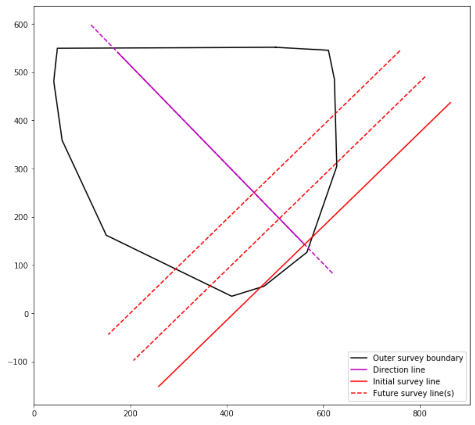


Figure 3.11: Establishing initial survey line

Determine stopping conditions

Several stopping conditions has been established, and one of either has to pass for the algorithm to end.

1. Next survey line is positioned outside a defined goal boundary.

2. Next survey line cannot be created due to not sufficient data available.

3. Swath lines cannot be approximated due to being outside survey boundary

First a defined goal boundary is established. Due to the fact that the shape of the survey boundary can vary a lot, it is hard to determine a boundary that fits any given shape. The initial goal boundary was to fit a simple rectangle between P1, P2 in 3.10, but an issue with this approach is that it could end prematurely because there could be pieces of the survey boundary not being included, which is visualized in 3.12.

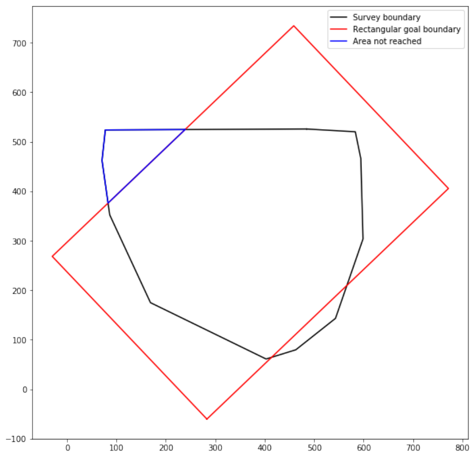


Figure 3.12: Rectangular goal boundaries

The other approach was to fit an circle around the survey boundary. The origin of the circle is placed in the center of gravity of the survey boundary. The radius is either determined by he vertex furthest away from the origin, or the weighted average of all the lengths from the origin to each vertex. An issue with this approach is that depending on which method is used to determine the radius of the circle, and the position of the initial survey line. The algorithm could end prematurely because the initial survey line is positioned outside the goal boundary. It is the center of gravity of the survey line that is checked to see whether it is inside or not.

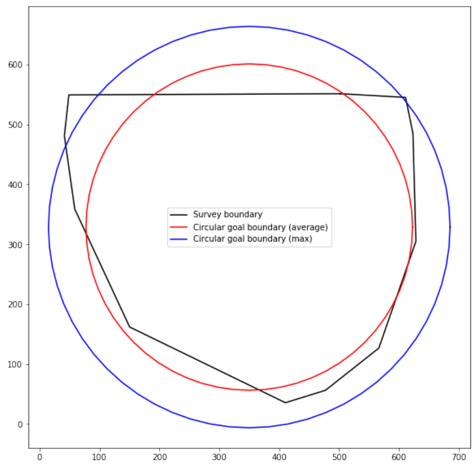


Figure 3.13: Circular goal boundaries

Due to it not being trivial to define a general goal boundary that consistently works, another stopping condition is used. Together with the algorithm stopping when the next survey line is outside the goal boundary, it also stops when the next survey line cannot be planned and positioned. Which occurs when the next survey line is completely on land, or is outside the data set boundary. It is the same case, if it stops due to not being able to approximate swath lines.

Determine sonar swath length

Due to the method chosen to retrieve depth data from the z-grid, the sonar swath length plays an important role. In this case, the swath length is defined as follows. The maximum swath width across the seabed from the center of the agent to one of the outer edges of the swath. This metric is used to determine how much depth data that needs to be retrieved from z-grid for finding out where the outer swath edges lands on the seabed. This is to prevent retrieving data that is outside the sonar range.

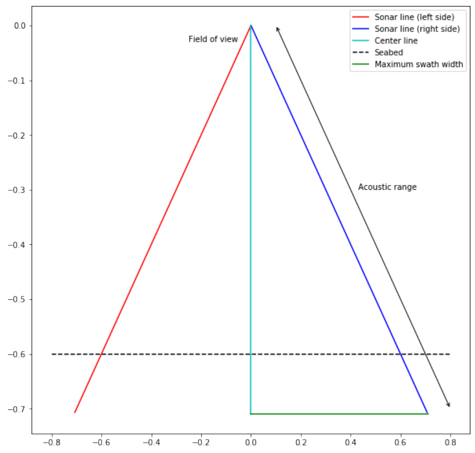


Figure 3.14: Definition of sonar swath length

The maximum sonar swath is determined by by the FOV (Field of View) and the acoustic range of the sonar. Because of the assumption in 3.3.1, the acoustic range is not that important because with all the environmental data available beforehand. It is assumed that the selected sonar can reach any depth, so some arbitrary fitting range is selected. Even with that assumption, it is still being used to prevent retrieving unnecessary data.

Algorithm process - straight survey lines

The process of the algorithm can be summarized with the steps in the following chart.

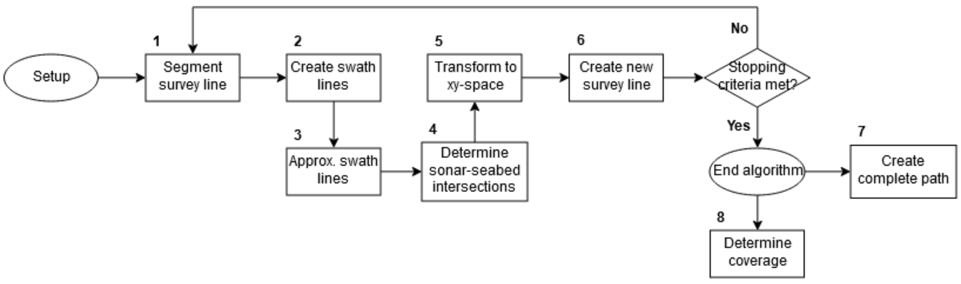


Figure 3.15: Algorithm process in steps

In setup the following steps are performed. Loading data set, setting up survey area, determining direction line, initial survey line, stopping conditions and sonar swath length.

1) Segment survey line

The survey line is segmented into points. The line is segmented into evenly spaced points that is determined by a step-size. This step size could represent how often the sonar does a reading, e.g. a reading every 5 meters. In this case, an appropriate step-size is chosen based on resolution and size of the environment.

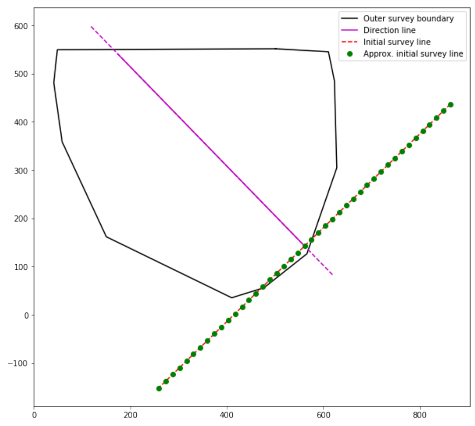


Figure 3.16: Segmented survey line

2) Create swath lines
From each point of the segmented line, a pair of perpendicular lines is extruded, one from each side. These lines are the swath lines, and each pair represents the maximum swath width of the sonar from an above point of view. Along each of these lines, an intersection between the sonar and the seabed will occur.

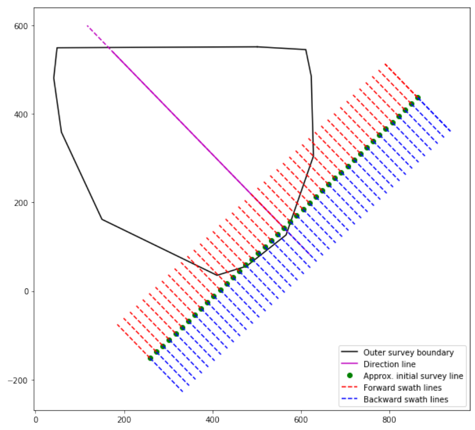


Figure 3.17: Creating swath lines

3) Approximate swath lines
In order to retrieve the appropriate depth data, each survey line is segmented into points, and approximated to fit the x and y-grid. Each swath line is segmented into evenly spaced points in the same way as the survey line. The spacing is determined by a step-size that

is also chosen according to the resolution and size of the environment. The indices of the approximated x, y values is used to retrieve the appropriate depth values from the z-grid. The depth values retrieved from each approximated swath line is used to determine a depth line. Which is then used to establish the intersection that occur between the sonar and the seabed.

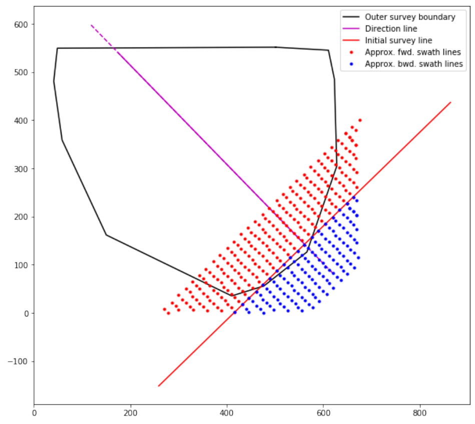


Figure 3.18: Approximated swath lines

4) <u>Determine sonar-seabed intersections</u>
Each pair of approximated swath lines in x, y-space is looked upon as a slice in x, z-space. The depth line (seabed) for each slice is defined as follows.

First, when a swath line is segmented into evenly spaced points, the distance of each point along the line is stored. For instance, if a 1 meter line is segmented into four evenly spaced points, the first point is at 0 meter, the second at 0.25m and so on. The distance is paired up with each retrieved depth value.

Second, finding the equivalent position of retrieved depth points in x, z-space is not straight forward considering rotation and orientation has to considered. To prevent that, interpolation is used. In Shapely, LineString objects has a interpolation method that returns a point at a specified distance along a line. With this method, is it trivial to find a specific point a long a line when the distance to that point from the origin of the line is known. Which is used to find the equivalent position in x, z-space without having to consider rotation.

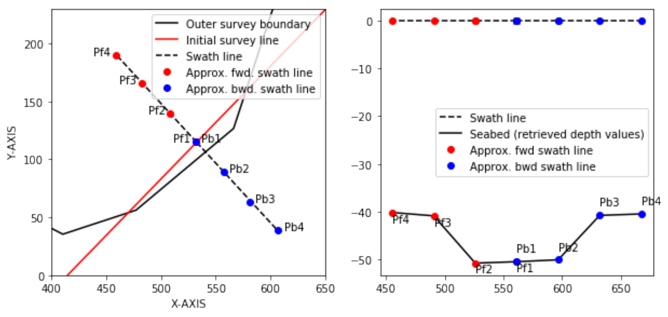


Figure 3.19: Establishing depth line, and switching between x, z and x, y-space

In the example scenario in 3.19. The acoustic range is 150m, and the FOV of the sonar is 90 degrees. This equals to a maximum swath width of 106m. The swath lines are segmented into 4 points due to step-size being 20m, where the distance of the Pfn (point-forward), and Pbn (point-backward) is equal to 106m/4 * n, where n = 0, 1, 2, 3. To find the equivalent position of the depth point in x, y-space. A swath line is defined, which is a flat line with the length of 106m. The length that is paired with each depth point (z-component) is used to determine the x-component. The length is used to interpolate and establish it's x-component along the flat swath line.

Third, when the depth line has been determined, it is paired up with the sonar model established in 3.17. Then it checks for intersections between the sonar and the depth line in x, z-space. These intersections represents where the outer swath edges of the sonar lands on the seabed.

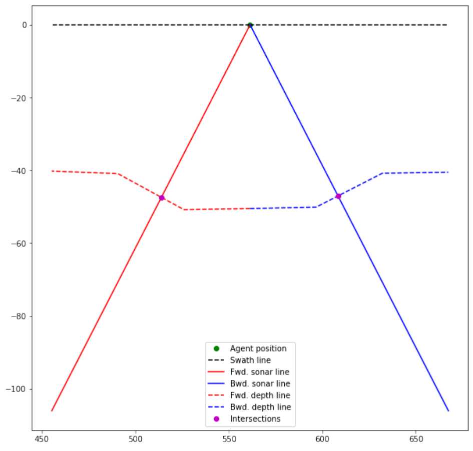


Figure 3.20: Intersections in x, z-space

5) Transform to intersections to x, y-space

As mentioned previously, for each pair of swath lines, the intersections will occur somewhere along the forward and backward swath line. In the previous step, the intersections was found in x, z-space. These intersections has to transformed to x, y-space to be used further. The same method as established in the previous step using interpolation is used here as well.

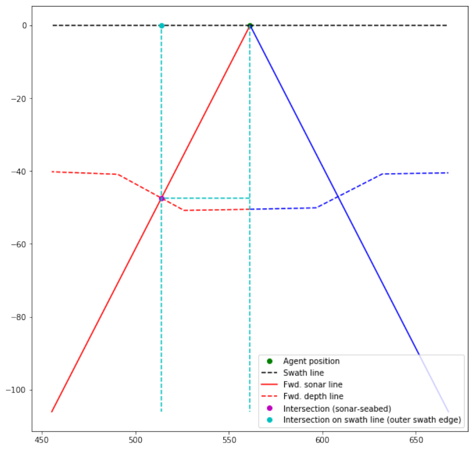


Figure 3.21: Intersection on forward swath line (x, z-space)

In order to determine where the equivalent of the intersection between the sonar and the seabed occurs along the swath line. The x-distance between the agent and the intersection is found. Then that distance is used in an interpolation to determine where the point is on the swath line. Each intersection is paired with its corresponding x-distance.

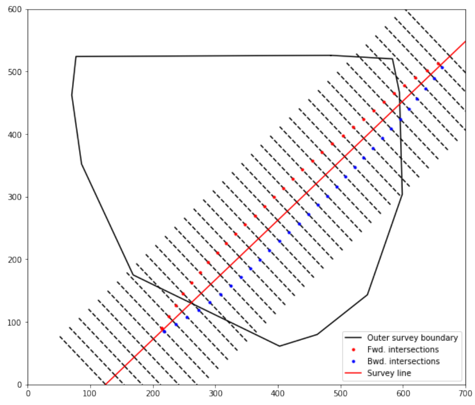


Figure 3.22: Intersection on forward swath lines (x, y-space)

In result, the forward and backward intersections along each swath line is determined, which is the outer swath edge of the sonar that establishes the area of coverage.

6) Create new survey line

The next survey line is created from the forward swath intersections in x, y-space. For each forward point, the distance between the point along the swath line and the survey line is stored. The next position of the survey line is determined one of the following strategies.

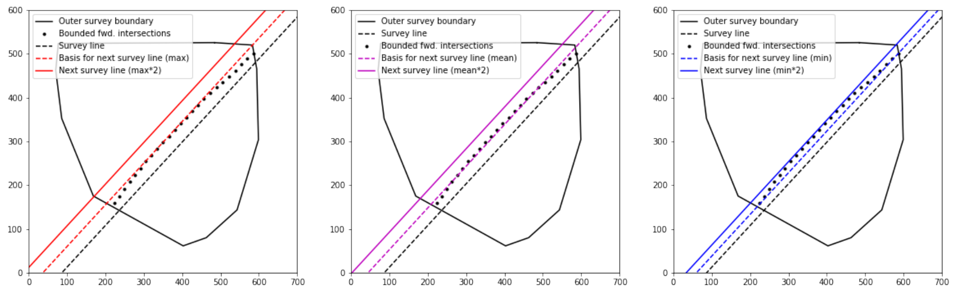


Figure 3.23: Next survey line (maximum, mean and minimum)

The position is selected by either choosing the point, shortest, furthest or the average distance away inside the survey boundary. The backward swath is also taken into consideration by placing the next survey line e.g. 2 times the maximum distance.

When the new position for the next survey line is chosen. The next survey line is created, moved and aligned according to the direction line, so that the next survey line is parallel placed to the previous one with a distance between which is determined by the strategy used.

The center point of the next survey line is checked if it is inside the goal boundary. If it is not, the algorithm ends. Also, if the next survey line cannot be established due to insufficient data, the algorithm will end as well.

7) Create complete path

To determine the complete path, the transition between each survey path has to be created. Which is accomplished by simply following the survey boundary when transitioning between survey lines. This is done in the following steps using set theory.

First, pairs of survey lines are merged together to form a survey polygon, so that each survey polygon consists of survey line[n-1] and survey line[n] where n = 1, 2, 3.., n. The rim of the survey polygon is also converted into a set of line segments. The reason why the survey lines are much longer than they need to be, is because of set theory being utilized. Each created survey polygon has to be large enough to encapsulate or in other words, intersect (slice out) a section of the survey boundary rim. These sections are called transition segments, and are scaled down so they do not overlap with the survey boundary.

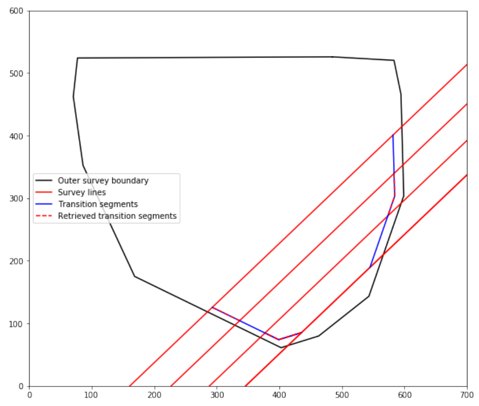


Figure 3.24: Transition segments

The transition segments are stored in a particular order. Which makes it trivial to retrieve the relevant transition segments following a pattern.

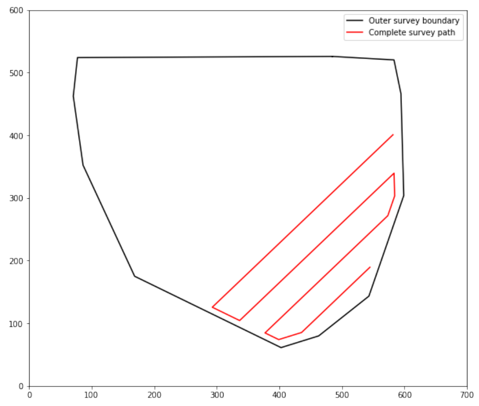


Figure 3.25: Complete survey path

Furthermore, the survey lines is then bounded by the outer survey boundary and then merged together with the retrieved transition segments to form a complete survey path

8) Determine coverage

Using the forward and backward outer swath edges retrieved in step 5), the amount of coverage and overlap can be determined. Using the same approach as in step 4), create coverage polygons from the resulting swath edges from each survey line, and use set theory.

First, the coverage polygons are established, which consist of the forward and backward outer swath edge from each survey line. These coverage polygons are bounded by the outer survey boundary as well.

Second, to determine the overlap polygons, the union between every polygon is checked (except itself) which will result in polygons representing overlap.

Third, to establish the total coverage, the all the coverage polygons are merged by taking the unary union of all the polygons.

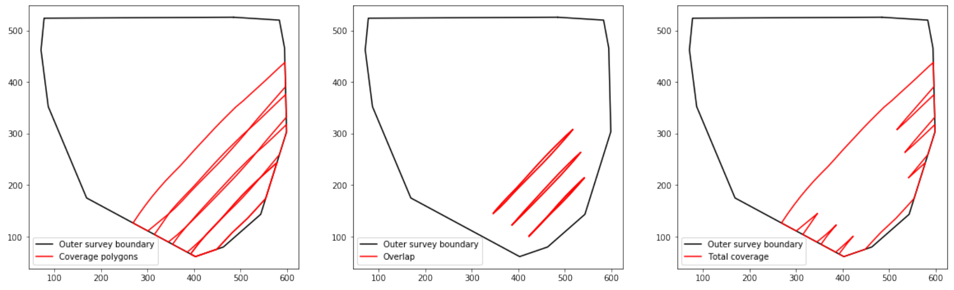


Figure 3.26: Coverage, (coverage polygons, overlap polygons and total coverage)

Fourth, because these are Shapely polygons. It is trivial to retrieve the area and do simple statistics such as calculating percentage of overlap and coverage. Which is the main metric used to determine whether the results are good or not.

3.5.2 Path Planning using Survey Paths

Compared to using straight survey lines, there is a number of notable differences. First, the direction line play a lesser role. In this case it is only used to determine the position and orientation for the initial survey line, it is also no longer used to align future survey lines.

Second, the same stopping conditions are used, but the goal boundary is more inconsistent because the placement of future lines are not as predictable.

Third, the idea is that each individual survey path is not a straight line anymore, but a series of smaller line segments that forms a survey path instead. There are more steps involved in processing the survey path, in terms of: segmentation, preparing, processing and planning the next survey lines. The strategy for planning new survey lines are different as well. Otherwise, it is mostly the same steps, which are summarized in the following chart.

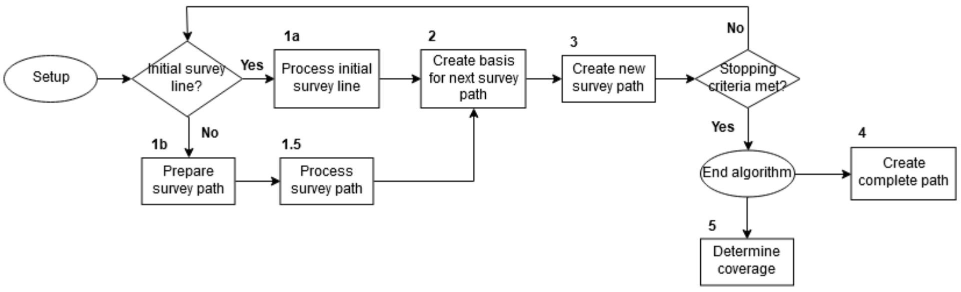


Figure 3.27: Algorithm process in steps

In the chart 3.27, the steps 1, 2, 3, 4 and 5 from 3.15 are encapsulated in the steps 1A, 1B, 2 and 3. The setup part is the same as the previous process, in terms of loading data set, setting up survey area, determining direction line and so on.

1a) Process initial survey line

The initial survey line determined by the direction line is processed using the exact method in 3.5.1. It will be the first and last straight survey line in the process. The process then deviates from the previous one after the forward and backward outer swath edges has been determined which is seen in 3.17.

2) Create basis for the next survey path

The steps 1b and 2 share a lot of same tasks. Tasks such as cleanup and simplification of the survey paths, so that the following steps can work without any errors. Another important task is to make sure that the survey path is continuous, and fixing it, if it is not. Step 2 mainly deals with creating the basis for the next survey path and cleaning it up, so that from the basis, the new path can be created without issues.

First, the survey path is cleaned up by removing unnecessary vertices and merging small line segments, but keep as much as original topology as possible. Furthermore, the number of trailing decimals for the coordinates of each vertex is reduced, which is to prevent losing continuity.

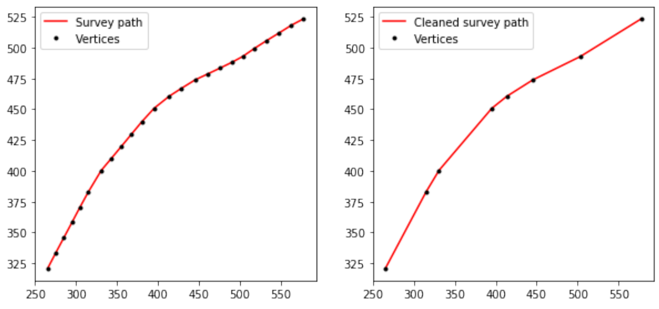


Figure 3.28: Cleaning basis for next survey path

Second, check whether line segments consists of more than 2 vertices, if it does, the line segment is segmented into sub segments. Each line segment should only consist of 2 vertices for simplicity.

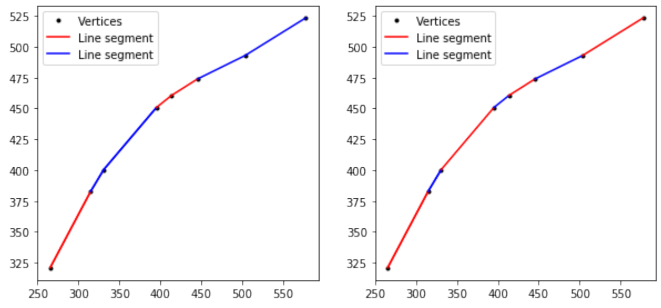


Figure 3.29: Sub splitting line segments

The goal is to simplify the basis for the next survey path as much as possible, without losing too much of its original topology. To accomplish this, the tolerance (distance between vertices) for the simplification function has to be chosen accordingly to the environment resolution, and step size chosen for further segmentation to establish swath lines. This is made possible through the methods available in the Shapely library.

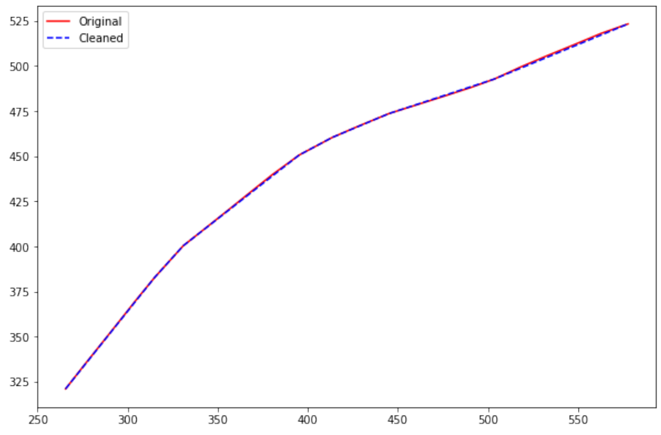


Figure 3.30: Difference between original and cleaned

3) Create new survey path
The new survey path is created from the basis for the next survey path by calculating the average heading for each vertex between pairs of line segments in the survey path. This method was interpreted from the work in Manda et al. (2016).

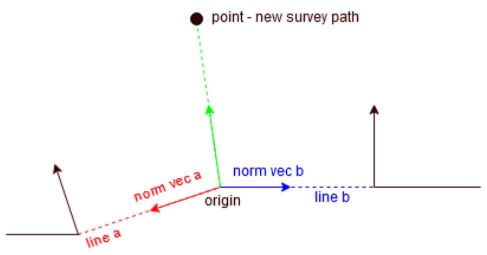


Figure 3.31: Average heading

As I interpret it, it follows a simple idea. Each vertex in the survey path has 2 line segments connected to it (unless it is an endpoint). A line segment that ends at the vertex, and another that starts at it. The normalized heading for each line segment is calculated, from which the the normalized average heading is calculated. This normalized average heading determines the direction of a point that will be further used to create the next survey path. How far out this point will be is determined by the distance of origin (length between intersection and survey line).

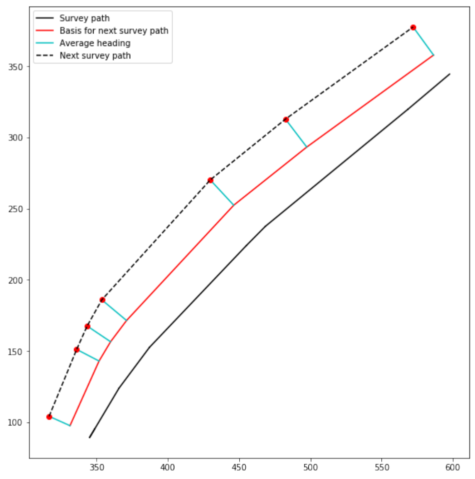


Figure 3.32: New survey path established

An observation that was made using this approach, was that the direction of the average heading vectors has a tendency to flip flop a lot. This depends a lot on the orientation of the pair of line segments used to calculate it. The get the specific behavior in 3.32, the x, y components of the vector is set to be always be negative, positive, or a mix of it. It becomes a big problem when the path curves in many different ways, so having a solution to manually set the sign of the components in a specific way for every scenario is not trivial at all.

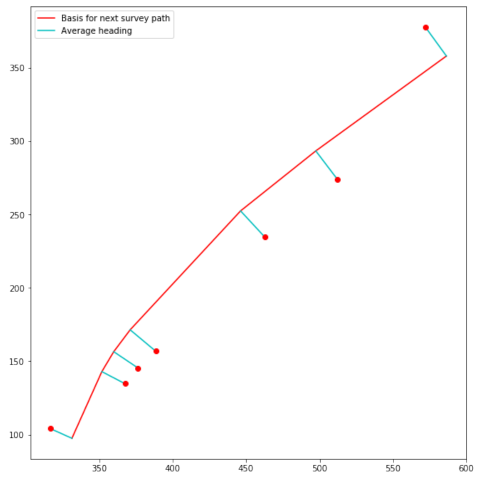
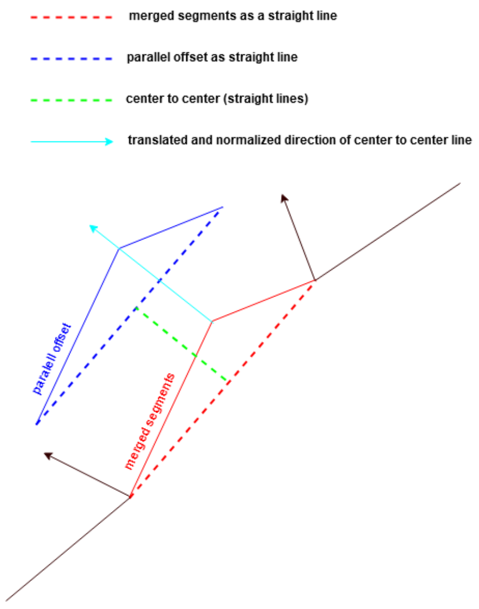


Figure 3.33: Vectors flip flopping

In 3.33, the average heading is calculated without adjusting signs of the x, y-components at all. I decided to try an approach that achieves the same results, but that the vectors faces the correct way (all pointing outward from the same side), and is mostly solved using methods in Shapely.



merged segments as a straight line
parallel offset as straight line
center to center (straight lines)
translated and normalized direction of center to center line
paralell offset
merged segments

Figure 3.34: Alternative method for calculating average heading

The alternative approach is quite straightforward as well. Pairs of line segments are merged together, and from the merged line segment a parallel offset line is created. The parallel offset line will always be created on the same side as long as the vertices of the survey paths are in the same order (e.g. clockwise). This offset line is used to align every vector in the same direction. The equivalent average heading vector is then created by first extruding a line between the center of the merged line segment and the parallel offset line. Secondly, move the line to the origin between the two merged line segments. Third, calculate the normalized direction of the line, and last, determine the length of the vector which establishes a point which is used to create the new survey path.

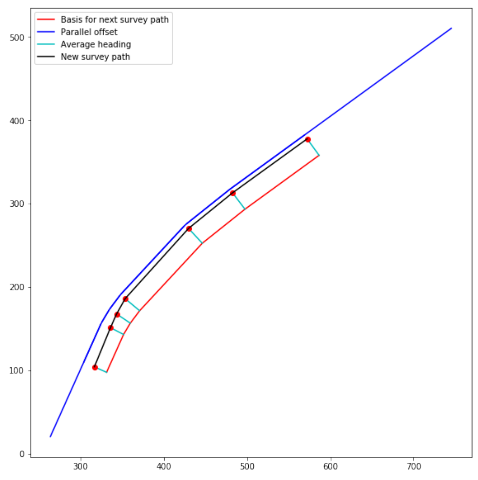


Figure 3.35: New survey path establishing using alternative method

In hindsight, the alternative method simply creates an parallel offset of the basis for the next survey path. Then for each vertex, the distance between the offset and the basis is adjusted.

4) Create complete path

The complete path is almost created in the exact method as previous, but because the shape of the created paths can vary a lot, some precautions is made. The coverage polygons needs to be checked whether they encapsulate and slices out the correct area from the outer survey polygon. If they do not, it is made sure they do by stretching out the survey paths.

5) Determine coverage

Coverage is determined in the same exact way as previously.

1b) Prepare survey path

As previously mentioned in step 2, steps 1b and 2 shares a lot of the same responsibilities. Which is doing cleanup so that the next steps can proceed without errors. Together with doing cleanup and making sure that the path consist of straight line segments, another responsibility is to make sure that both sides of the survey path reaches all the way to the survey boundary. This is done by simply stretching out the end segments of the survey path. The idea is to prevent gaps in coverage.

1.5) Process survey path

Compared to processing a straight survey line, the only major difference is how the survey path is segmented into points compared to a straight line, because the survey path is a collection of straight line segments. Otherwise creating swath lines, approximation of swath lines, determining sonar-seabed intersections and transforming them to x, y-space use the same methods as previously.

For a straight line segment, is it trivial to segment it into points because it is linear. To determine the points, x, y just needs to gradually increase or decrease (depending on orientation) between a min and maximum x, y-value. That is not the case for a non-linear survey path.

The method used to segment the survey path into points do not require it to be segmented into straight line segments. It still needs to be done because the method used to create perpendicular lines (swath lines) requires a base line to extrude them from. The survey path is segmented into points by using the length and interpolation.

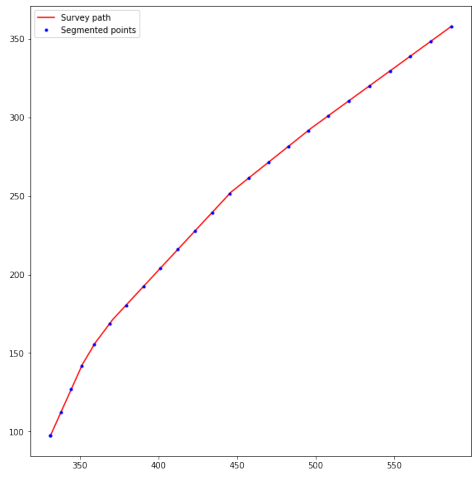


Figure 3.36: Survey path segmented into points

The length of the survey path is segmented into pieces, which as then further used in interpolation to retrieve the point for each length along the survey path. Then it is determined which line segment of the survey path each interpolated point belongs to. From there on, the swath lines created.

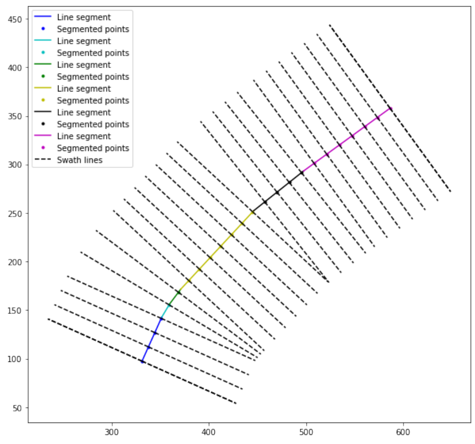


Figure 3.37: Segmented line segments with points

After the swath lines has been created, it follows the same process as previously for approximation and determining sonar-seabed intersections.

Chapter 4

Cases

4.1 Planning Path for Seabed Mapping Outside Ørsta

For the first case, an area outside the coastline of Ørsta has been selected. The data for this area has been extracted from the data set in 3.4. In this case the survey boundary is bounded by the coastline and automatically retrieved.

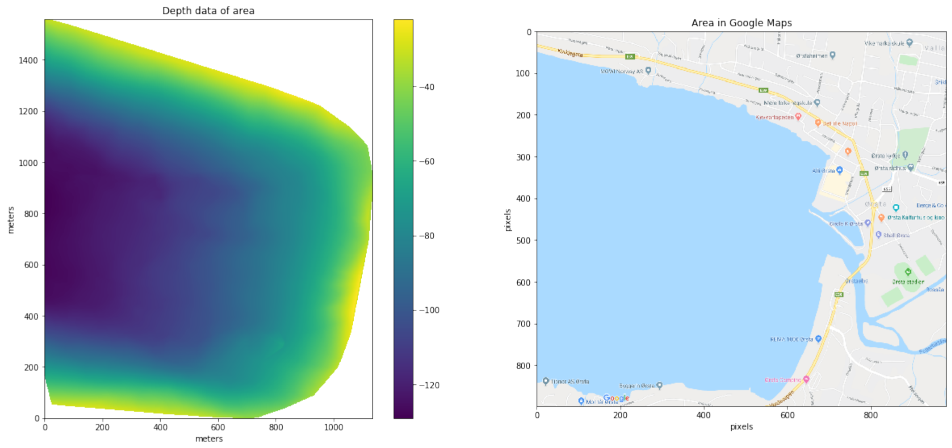


Figure 4.1: Depth data and equivalent area in Google Maps

The depth data encompasses an area that is around 1000x1200 meters according to the data set. Which is confirmed to be correct according to Google Maps.

4.1.1 Applying Algorithm

Determine direction line and initial survey line

Before the path can be planned, the direction line and the initial survey line has to be determined. This is done on a down sampled version of the data set. The dimensions of the original is 2000x2000x3, but for this I am using a 250x250x3 version to speeds things up a bit.

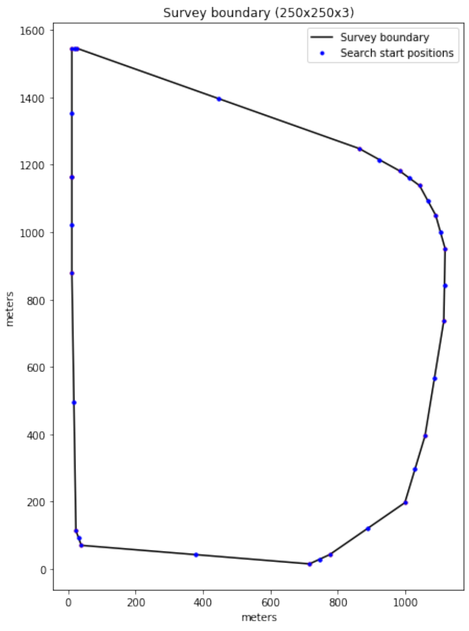


Figure 4.2: Search start positions

Vertices and the center-point for each line segment in the survey boundary is used as a search start position. For each search start position, a search to determine the direction line is made, which will results in a potential position for the initial survey line.

For each resulting direction line and initial survey line, the algorithm is ran using the straight survey line method utilizing the mean-strategy. The sonar FOV is set to be 90 degrees and has an acoustic range of 200m. The step sizes for swath lines and retrieval of depth line is set to be 10-15m. The pair has the best results in terms of overlap and coverage percentage is selected for further use.

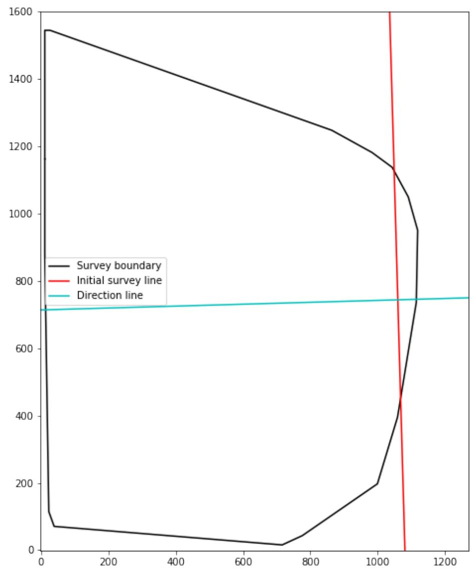


Figure 4.3: Selected direction line and initial survey line

The configuration in 4.3 was selected due to having the best results in terms of coverage and overlap.

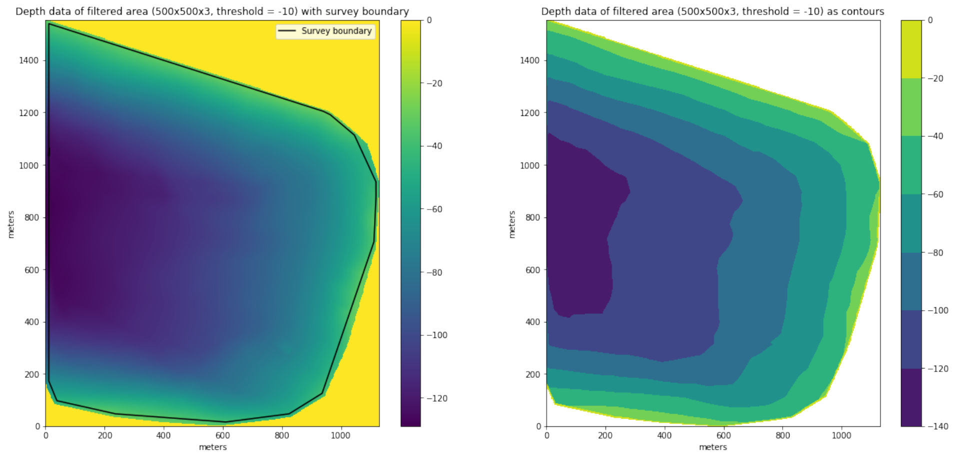


Figure 4.4: Depth area, survey boundary and area as contour

In 4.4, is the depth area and boundary that will used further for the following steps. The dimension of the depth area is 500x500x3, and the filter threshold is -10m. Which means that any depth shallower than -10m is considered land or depth 0 (surface).

Path planning with straight survey lines - static

In order to have something to compare to, a run using a constant spacing between the survey lines is used. The amount of spacing is determined by shallowest point in the survey area. This is to guarantee that every point in the area is reached.

In depth area in 4.1, the shallowest point is -23,69m. The spacing is determined in the following figure.

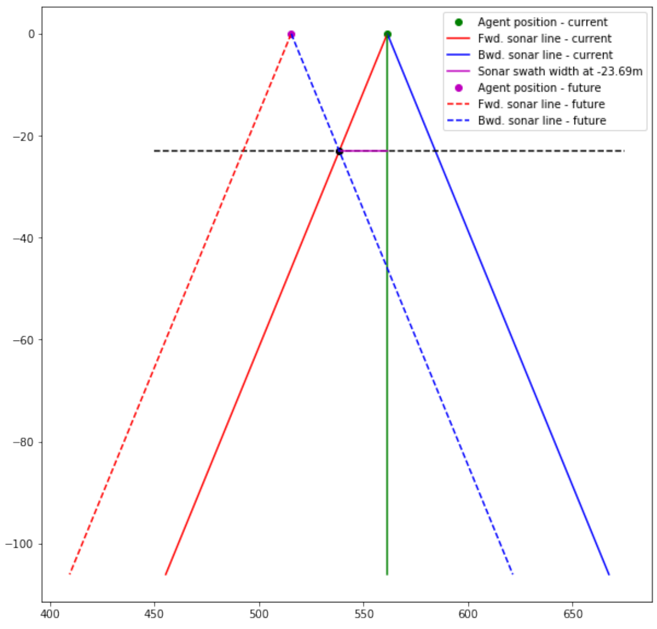


Figure 4.5: Determining value for constant spacing

In 4.5, the calculated spacing is to be 23m, to prevent any gaps at the shallowest point. The sonar FOV is set to be 90 degrees and has an acoustic range of 200m. The step sizes for both swath and depth is 10m, which follows the idea that every 10m a sonar reading is performed, and for every 10m along the swath line, depth data is retrieved to establish the depth line.

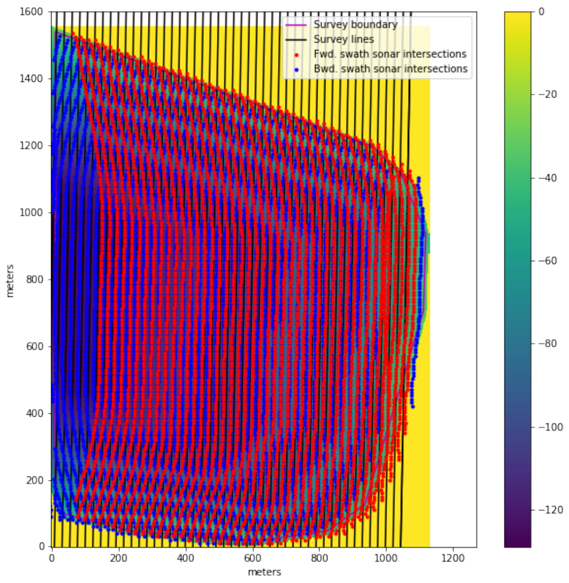


Figure 4.6: Results, straight survey lines - static

The run had an elapsed time of 16 minutes.

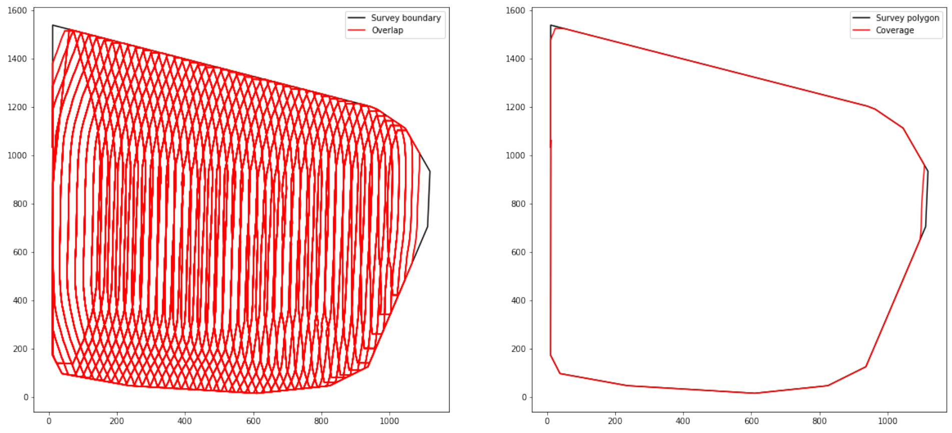


Figure 4.7: Overlap and coverage, straight survey lines - static

Of the whole survey boundary, 99.7% was covered, but there was 5015% overlap.

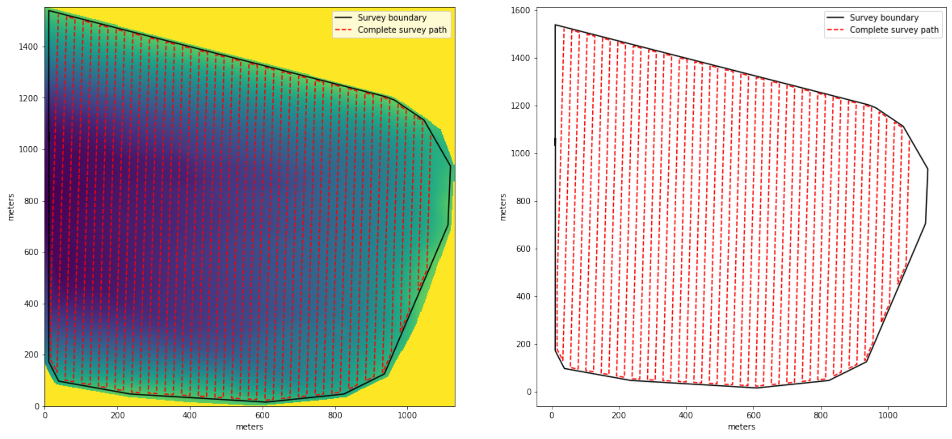


Figure 4.8: Complete coverage path, straight survey lines - static

The complete survey path had a length of 58353 meters.

Path planning with straight survey lines - adaptive

Using the exact same depth data, and configuration as previously. Three runs with the adaptive straight survey lines method established is used. One run for each strategy, with the overlap factor not used.

Maximum-strategy

Using the max-strategy resulted a total run-time of 2 minutes. Of the whole survey boundary, 84.06% was covered where there were 0.01& overlap. The complete path had a total length of 8190.7m.

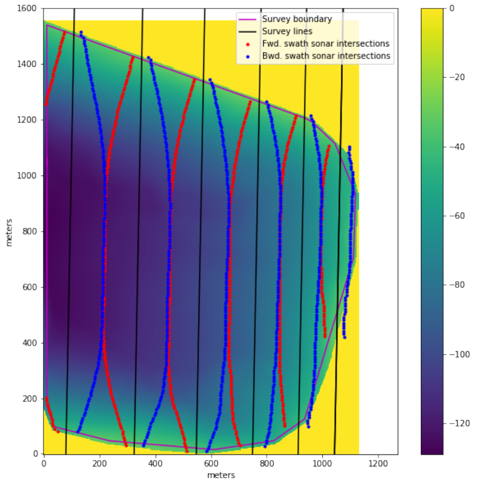


Figure 4.9: Result, straight survey lines - adaptive (max)

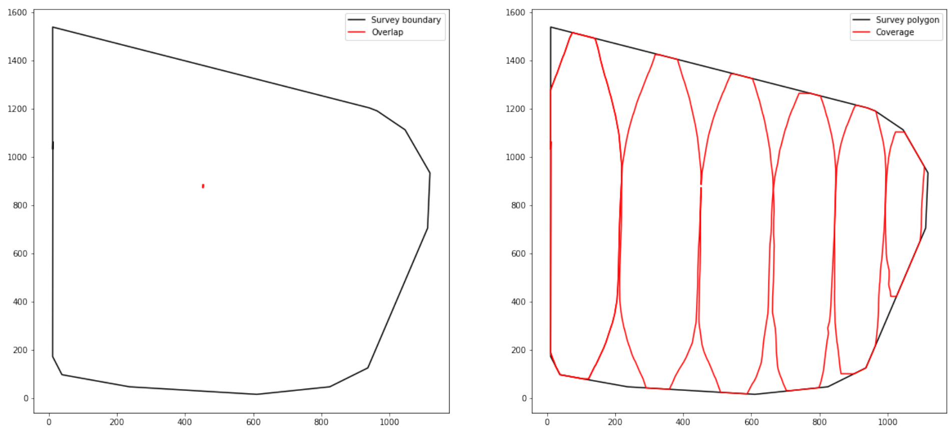


Figure 4.10: Overlap and coverage, straight survey lines - adaptive (maximum)

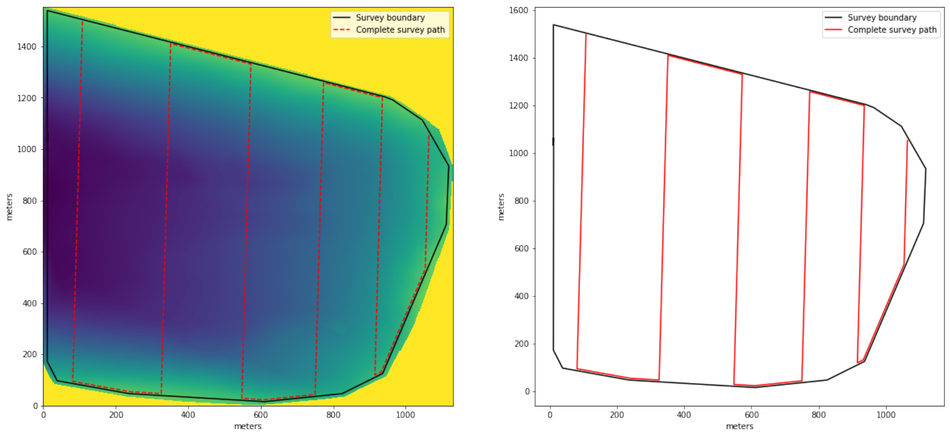


Figure 4.11: Complete path, straight survey lines - adaptive (max)

Mean-strategy

Using the mean-strategy resulted a total run-time of 2.14 minutes. Of the whole survey boundary, 90.42% was covered, there was a total overlap of 17.2%. The complete path had a total length of 9516.2m.

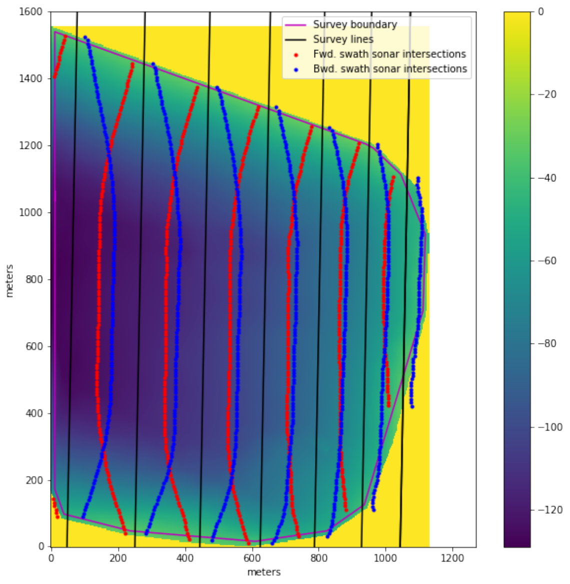


Figure 4.12: Result, straight survey lines - adaptive (mean)

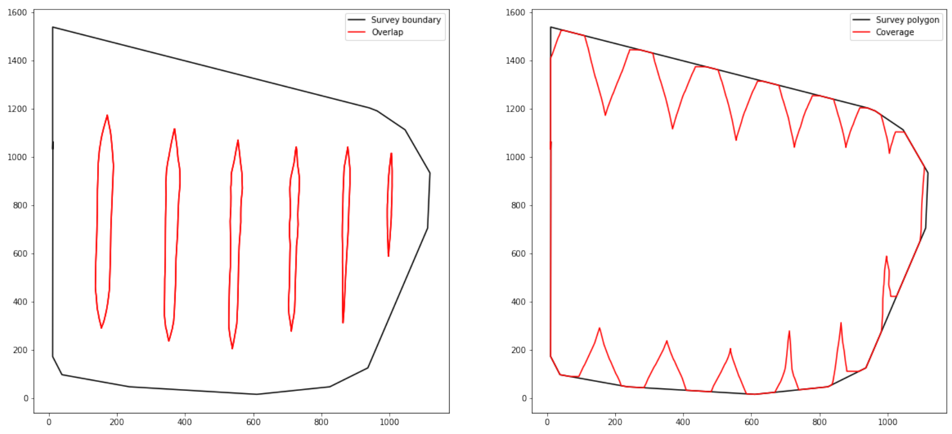


Figure 4.13: Overlap and coverage, straight survey lines - adaptive (mean)

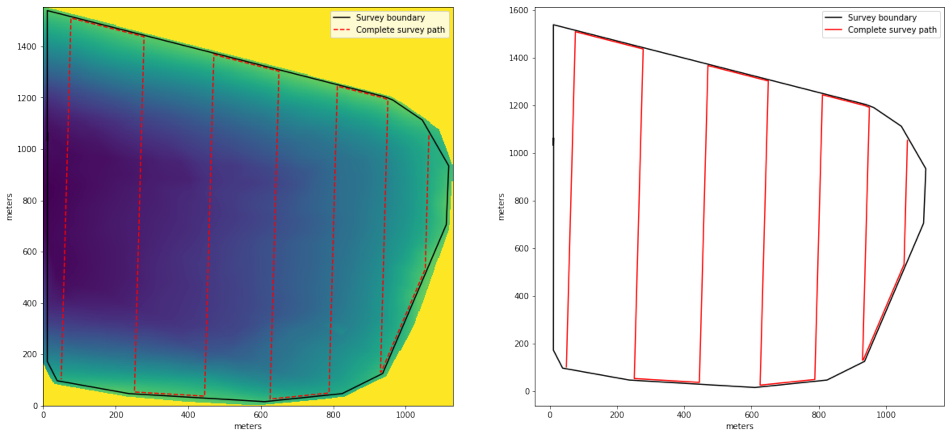


Figure 4.14: Complete path, straight survey lines - adaptive (mean)

Min-strategy

Using the min-strategy resulted a total run-time of 6 minutes. Of the whole survey boundary, 99.53% was covered, there was a total overlap of 440.1%. The complete path had a total length of 20994.1m.

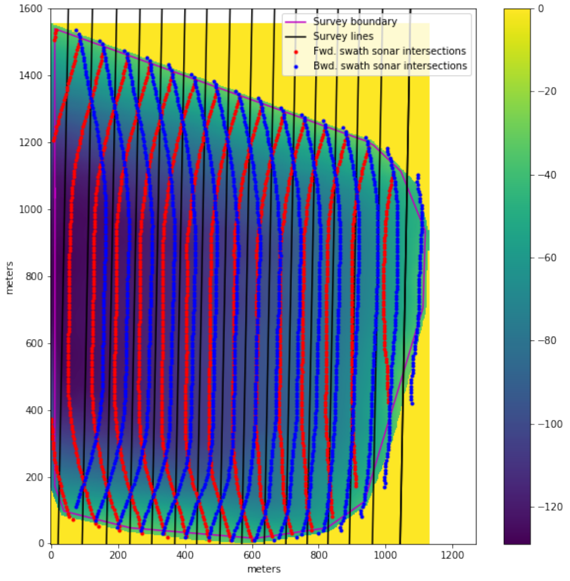


Figure 4.15: Result, straight survey lines - adaptive (min)

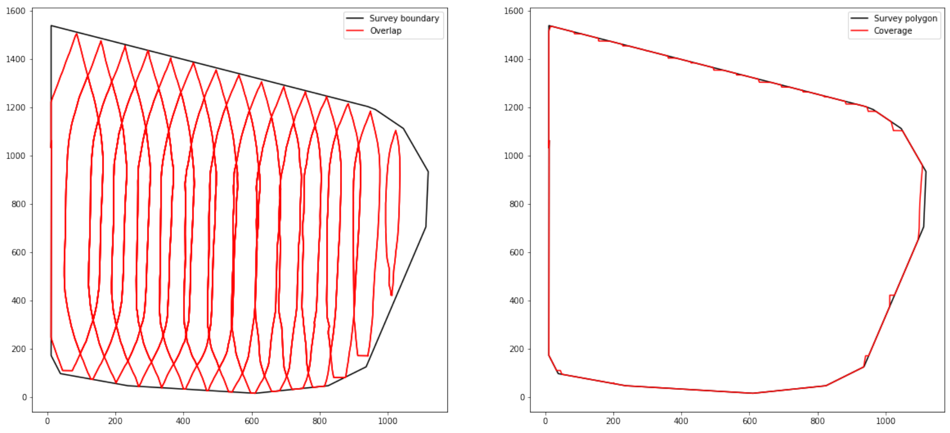


Figure 4.16: Overlap and coverage, straight survey lines (min)

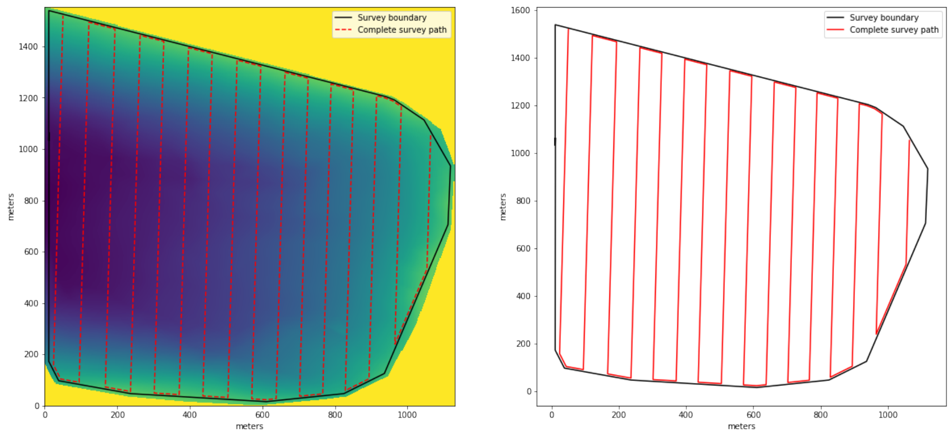


Figure 4.17: Complete path, straight survey lines - adaptive (min)

Path planning with survey path lines - adaptive

Due to inconsistencies in this method, the same dimension as the one used for determining direction line and initial survey line was used. The configuration for direction and initial survey line was changed to 4.18 as well, so that is starts from the deeper side.

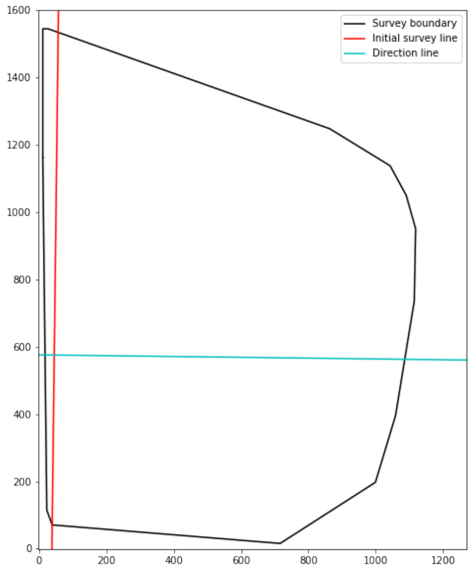


Figure 4.18: Switched direction and initial survey line configuration

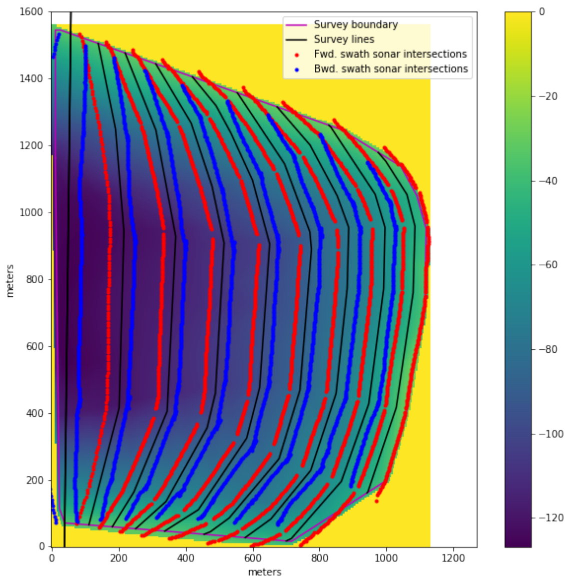


Figure 4.19: Result, survey paths - adaptive

The run had a total run-time of 2.14 minutes.

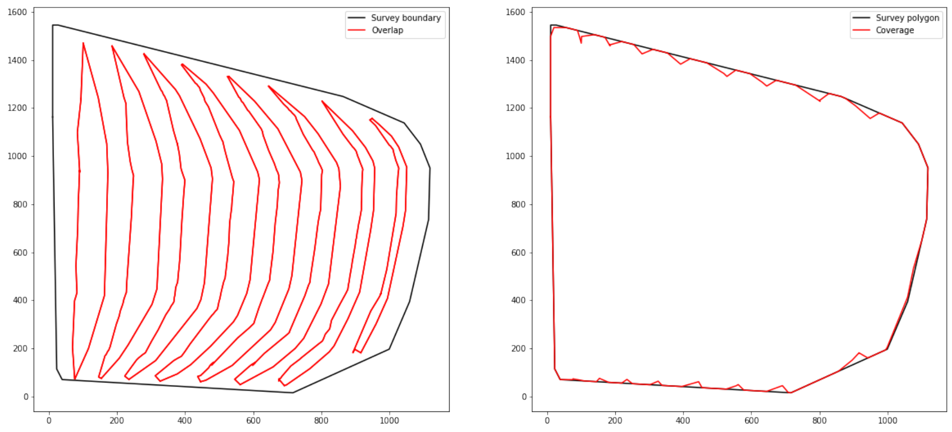


Figure 4.20: Overlap and coverage, survey paths - adaptive

This resulted in a total coverage of 98.79% coverage, and a total of 82.54% overlap.

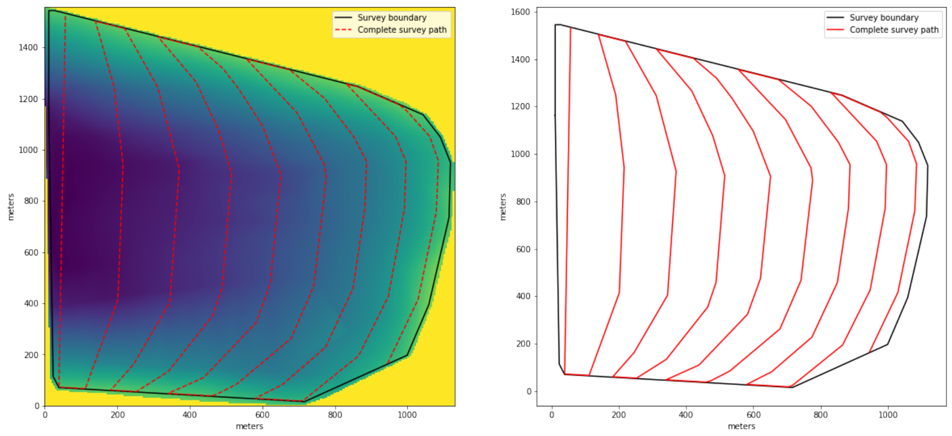


Figure 4.21: Complete path, survey paths - adaptive

The complete path had a total length of 13633.8m.

4.2 Planning Path for Seabed Mapping Outside Hareid

For the second case, an area outside the coastline of Hareid has been selected. The data for this area has been extracted from the data set in 3.4. In this case the survey boundary is manually defined, which is an rectangle.

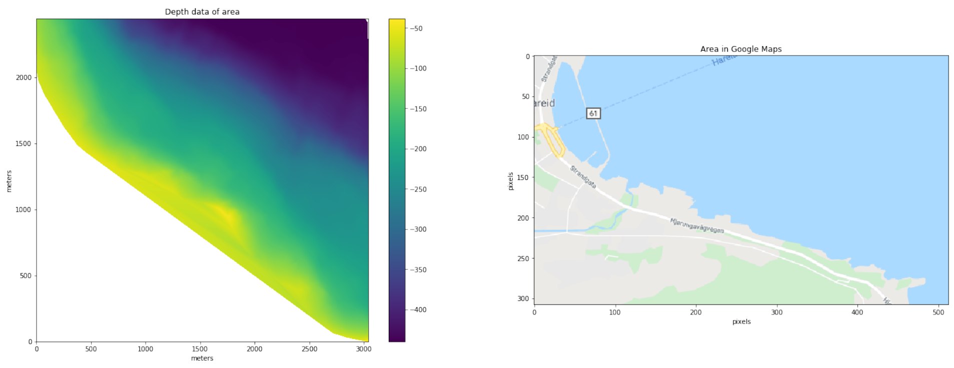


Figure 4.22: Depth data and equivalent area in Google Maps

The depth data encompasses an area that is around 3000x2000 meters divided by 2 (because of triangular shape) according to the data set.

4.2.1 Applying Algorithm

Determine direction line and initial survey line

As before, the direction line and the initial survey line has to be determined. This is done on a down sampled version of the data set as well. The dimensions of the original is a bit larger at 3000x3000x3, but for this I am using a 300x300x3 version to speed things up a bit.

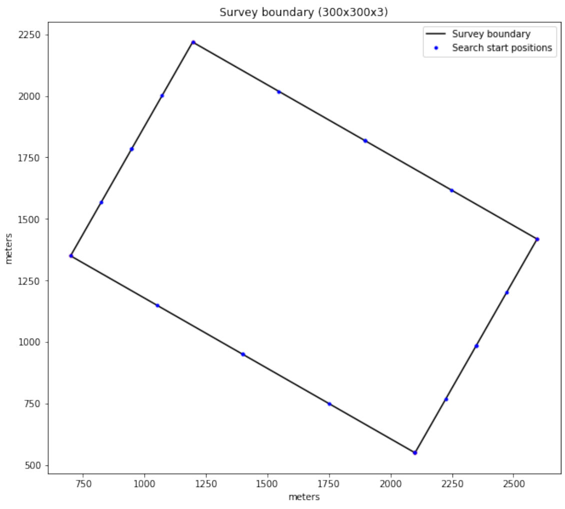


Figure 4.23: Search start positions

The same method as used previously is used to determine the direction line and the initial survey line.

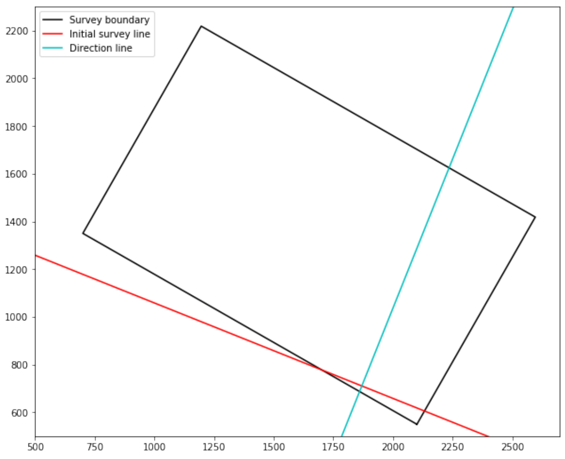


Figure 4.24: Selected direction line and initial survey line

The selected direction line and initial survey line in 4.24 gave the best results in terms of coverage and overlap

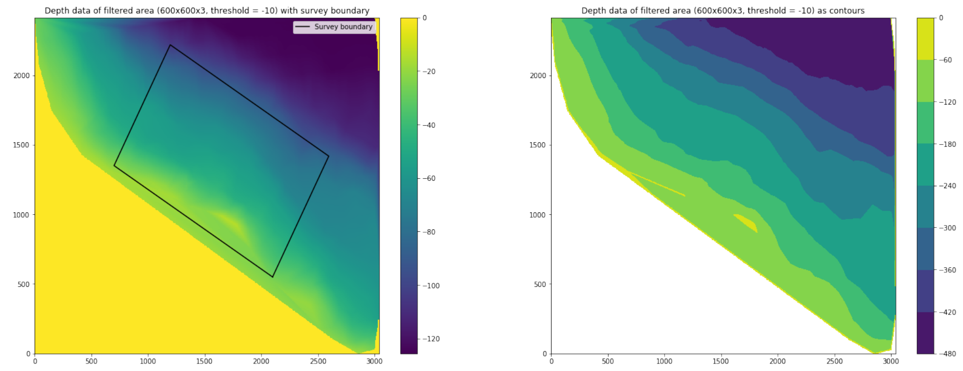


Figure 4.25: Depth area, survey boundary and area as contours

In 4.25 is the depth area and the boundary that will be used further in the following steps. Dimension of the depth area is 600x600x3, and uses the same filter threshold as previous. Since the dimensions are higher than previous I decided to increase the step size to 15m instead of the previously 10m. Due to very deep depth levels, for the sake of reducing massive run times, the depth is divided by a factor of 3.5, so that the real depth is times 3.5. In 4.22 the depth is 0-400m, and after reduction it is 0-120m.

Path planning with straight survey lines - static

A survey using constant spacing between survey lines is done here as well to use as a reference. Using the same method as previously to determine the spacing used. In the depth area in 4.25, the shallowest point is 55.1m. This results in a spacing of 55m, but after reduction, it is 13.8m.

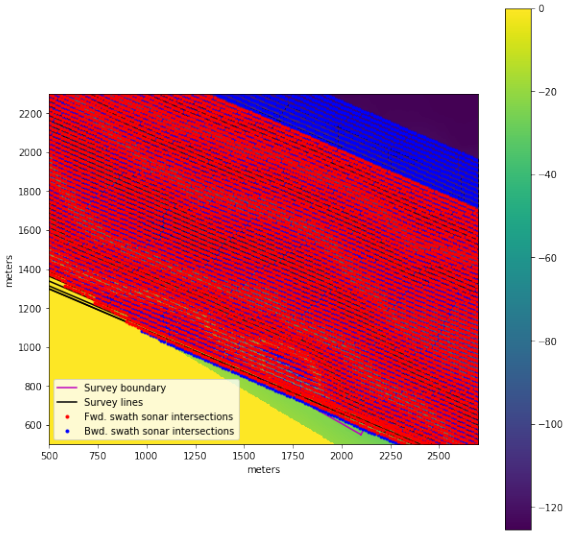


Figure 4.26: Results, straight survey lines - static

The total run-time was 37 minutes.

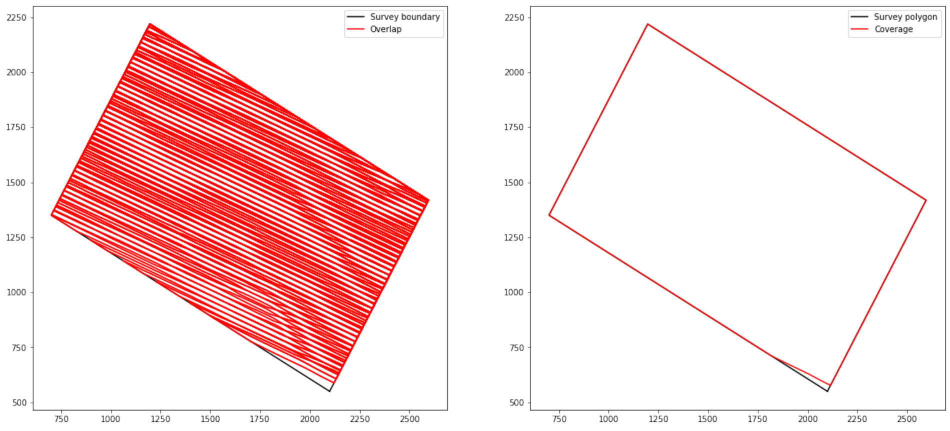


Figure 4.27: Overlap and coverage, straight survey lines - static

The total coverage was 99.69%, with a total of 5014.5% overlap,

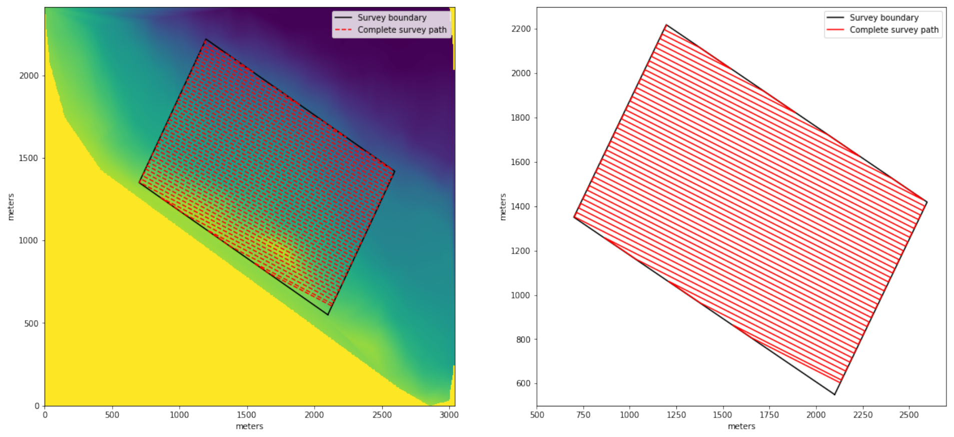


Figure 4.28: Complete coverage path, straight survey lines - static

Length of the complete survey path was 71853.1m.

Path planning with straight survey lines - adaptive

For the adaptive method, three following runs using each strategy is done as well.

Maximum-strategy

The run-time was 4.28minutes. Of the whole survey boundary, 70.05% was covered, and there was a total overlap of 0.05&. The complete path had a length of 12817.1m

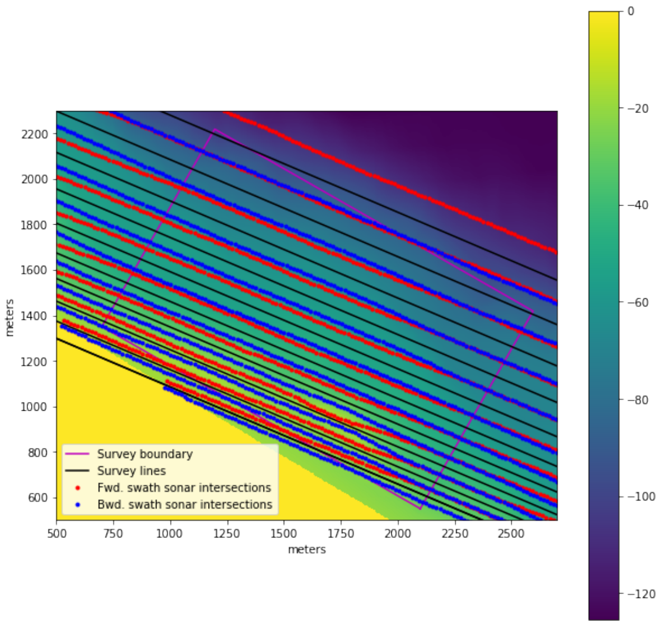


Figure 4.29: Result, straight survey lines - adaptive (max)

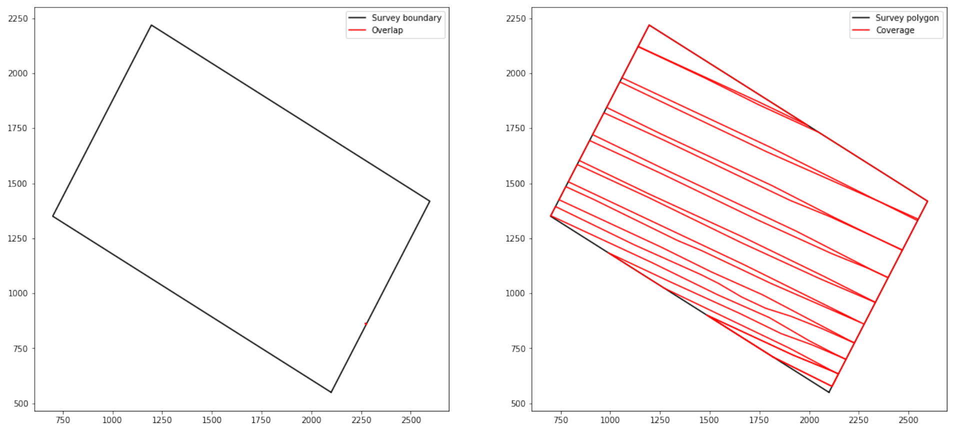


Figure 4.30: Overlap and coverage, straight survey lines - adaptive (max)

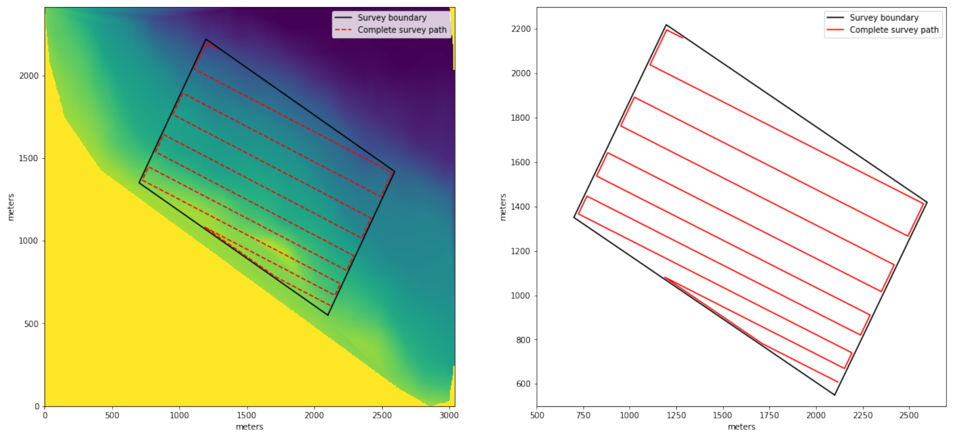


Figure 4.31: Complete path, straight survey lines - adaptive (max)

Mean-strategy

The run-time was 10minutes. Of the whole survey boundary, 95.78% was covered, and there was a total overlap of 8.08%. The complete path had a total length of 18463.5m.

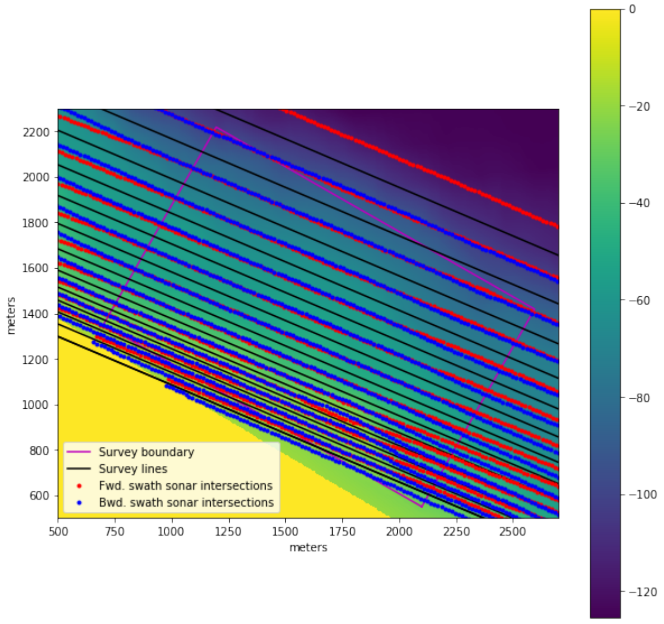


Figure 4.32: Result, straight survey lines - adaptive (mean)

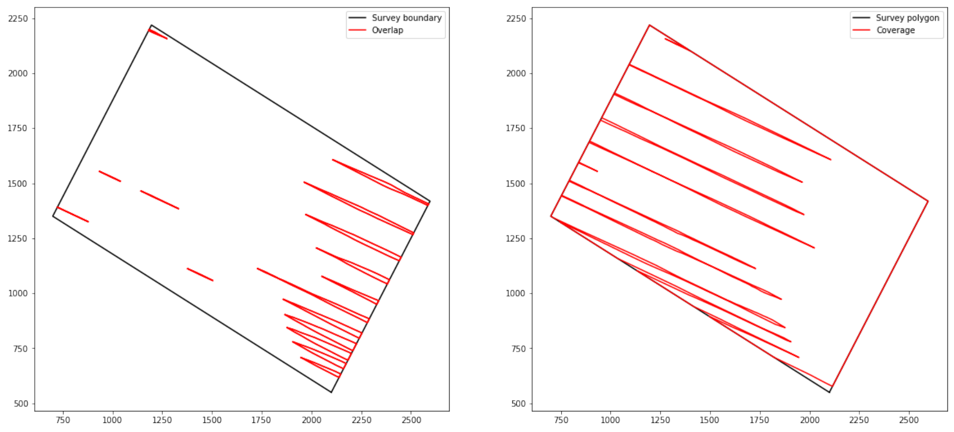


Figure 4.33: Overlap and coverage, straight survey lines - adaptive (mean)

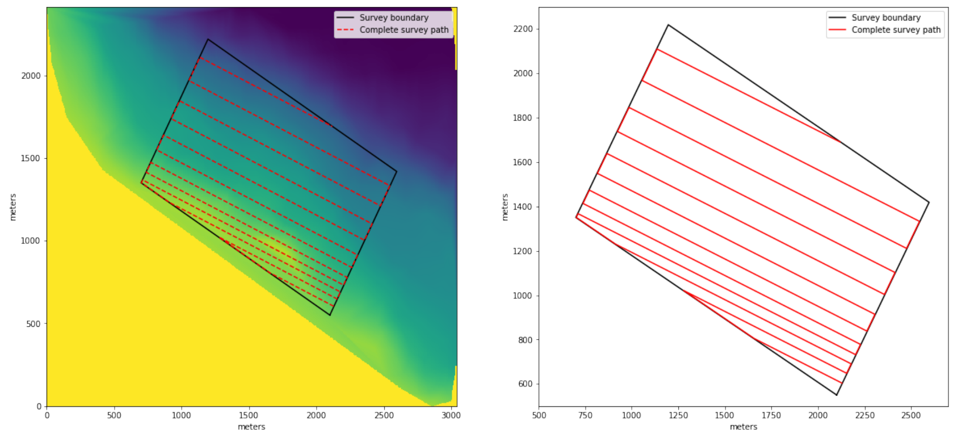


Figure 4.34: Complete path, straight survey lines - adaptive (mean)

Min-strategy

The run-time was 13minutes. Of the whole survey boundary, 99.67% was covered, and there was a total overlap of 30.65%. The complete path had a total length of 23288.7m

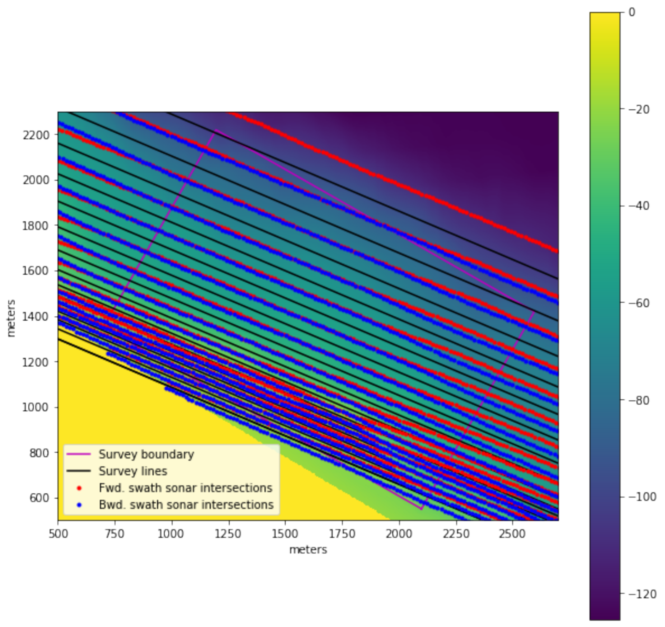


Figure 4.35: Result, straight survey lines - adaptive (min)

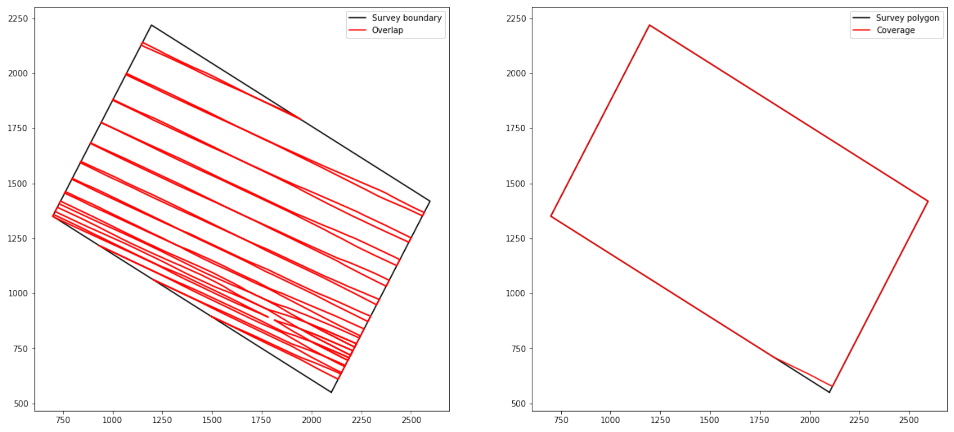


Figure 4.36: Overlap and coverage, straight survey lines - adaptive (min)

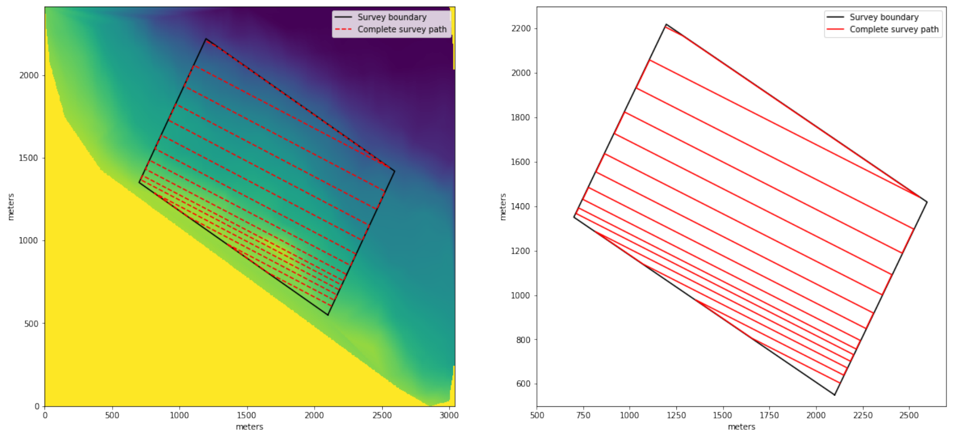


Figure 4.37: Complete path, straight survey lines - adaptive (min)

Path planning with survey path lines - adaptive

Due to inconsistencies in this method, the same dimension as the one used for determining direction line and initial survey line was used. The configuration for the direction and initial survey line had to be changed as well, 4.38.

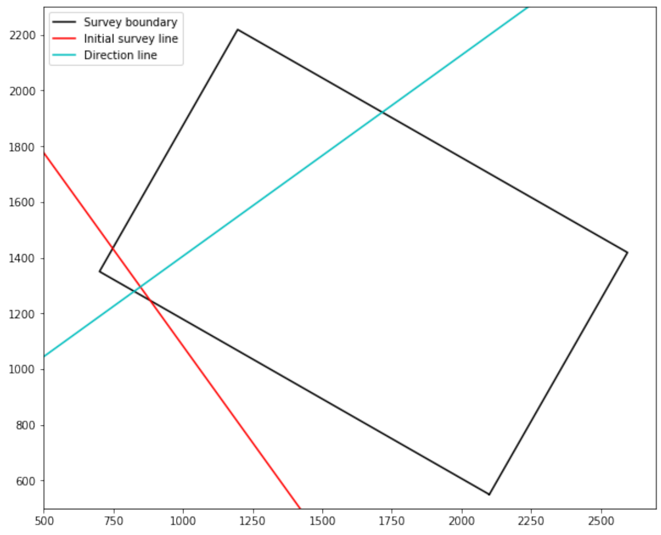


Figure 4.38: Switched direction and initial survey line configuration

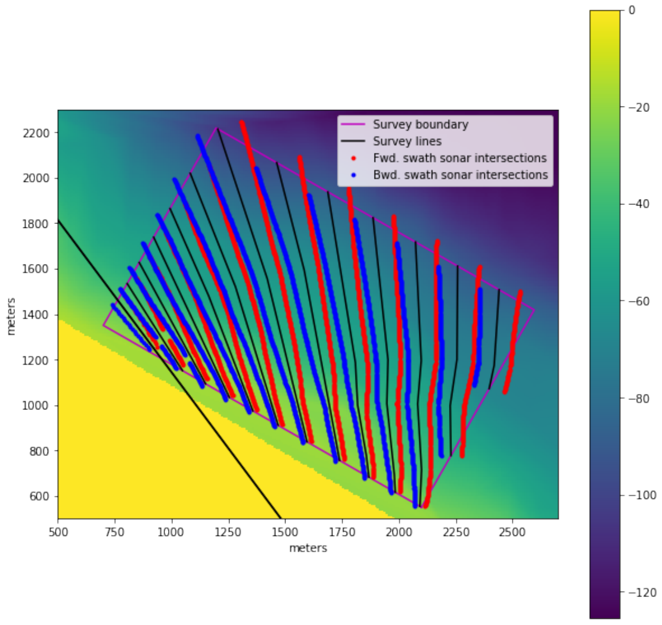
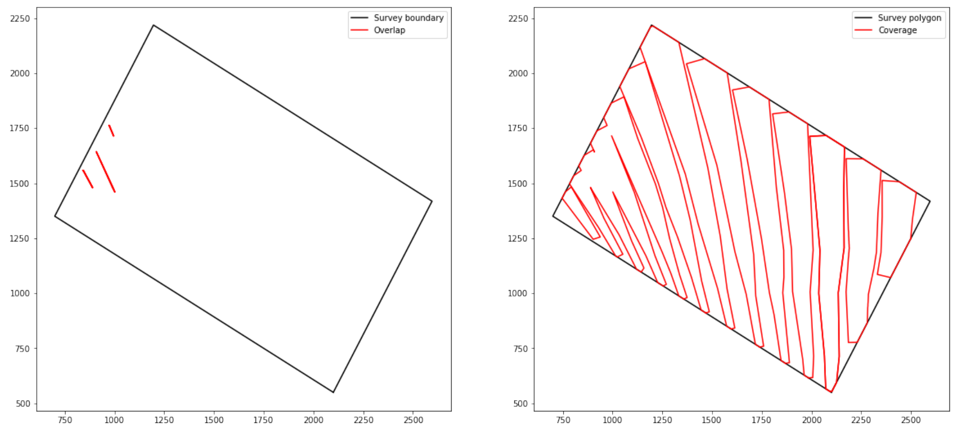


Figure 4.39: Result, survey paths - adaptive

A total run-time of 1.4 minutes.



This resulted in total coverage of 73.52% and a total overlap of 0.05%.

Figure 4.40: Overlap and coverage, survey paths - adaptive

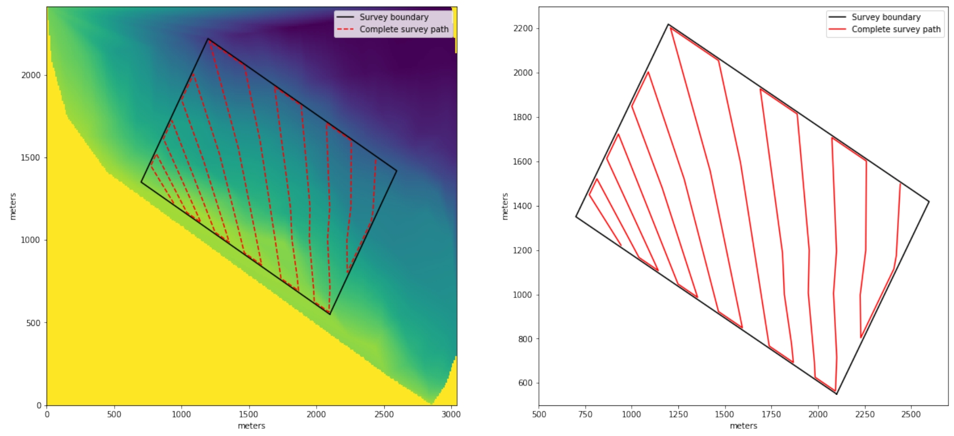


Figure 4.41: Coverage path, survey paths - adaptive

The complete path had a length of 13750.3.8m.

Chapter 5

Results and Discussion

5.1 Case 1: Ørsta

For the first case I selected a region that is bounded by the surrounding coastline. The region selected is an area outside the coastline of ørsta.

5.1.1 Configuration

Experiments 1-4 used the same 500x500x3 version of the data set, direction line, initial survey line and same parameters. Experiment 5 uses a more down-sampled 250x250x3 version, but had to use another direction line and initial survey line entirely. Due to complications, this experiment had to start from the deeper side and move towards the shallow side otherwise it would not complete. The reason for this is because of the topology of the depth. Going from shallow to deeper end, the survey lines would start curving a lot, and lead to issues determining the complete survey path. The sudden cut-off also causes issues.

I did not setup a good structure to determine either the parameters for the data set or the algorithm, which I should have done in hindsight. The resolution of the data set and the chosen algorithm was selected based on what seemingly works well. I should had more structure and tests to determine good parameters. Because I think that what I selected was a overkill in some cases.

Overkill in terms of the resolution of the data set being larger than necessary. I should also have done tests to choose the best configuration in terms of both the data set and algorithm parameters and stuck to it. I also think I should not have used different sampled versions of the same data set, e.g. survey boundary of a 500x500x3 is going to be different from a 250x250x3. I should have stuck with the 250x250x3 version to prevent any inconsistencies.

5.1.2 Results

In this case I did the five following experiments.

1. Planning using straight survey lines - constant spacing: This experiment had a run-time of 16 minutes, a total coverage of 99.7%, a total overlap of 5015% and a total complete path length of 58353m.

2. Planning using straight survey lines - adaptive spacing (maximum): This experiment had a run-time of 2 minutes, a total coverage of 84.06%, a total overlap of 0.01% and a total complete path length of 8190.7m.

3. Planning using straight survey lines - adaptive spacing (average): This experiment had a run-time of 2.14 minutes, a total coverage of 90.42%, a total overlap of 17.2% and a total complete path length of 9516.2.m.

4. Planning using straight survey lines - adaptive spacing (minimum): This experiment had a run-time of 6 minutes, a total coverage of 99.53%, a total overlap of 440.1% and a total complete path length of 20994.1m.

5. Planning using survey paths - adaptive spacing (average heading): This experiment had a run-time of 2 minutes, a total coverage of 98.79%, a total overlap of 82.54% and a total complete path length of 13633.8m.

Looking at the results above, experiment 3-5 yielded the best results, but it depends on what is prioritized. Experiment 4 is a straight upgrade from 1, but experiment 3 is also great if overlap and path length is a priority. Experiment 1 is good if quick run with barely any overlap and with the shortest path length is needed, but coverage is least prioritized. Experiment 5 also yielded good results, but it is an outlier considering it had to be ran on a different configuration, together with switching the pair of direction line and initial survey line due to complications.

By looking at 4.11, using the maximum-strategy, the spacing gradually increases because the deepest point from each survey line is used. It is the same case in 4.14 using mean-strategy because the average increases with each line. For the minimum-strategy 4.17, the spacing seem to be around the same throughout because of the shallow points along the coastline. Using average heading 4.20, it was often a hit or miss because it would break, or the coverage or complete path could not be determined. In this case, it seems to work okay when starting from the deeper side. I can imagine it being a lot better if each survey path was less simplified, e.g. each survey path consisting of several smaller segments, but doing so would likely make the algorithm more unstable.

5.2 Case 2: Hareid

For the second case I selected a region that is bounded by a manually defined survey boundary. The region selected is an area in the coastline between Hareid and Hjørungavaag.

5.2.1 Configuration

Following the same approach as previously. Experiments 1-4 used the same version 600x600x3 of the data set, direction line, initial survey line and same parameters. Experiment 5 uses a more down-sampled 300x300x3 version and a different pair of direction and initial survey line. The data set used in this case encompasses a much larger area, so the dimensions are larger compared to the previous case. In terms of structure and parameters, follows the same approach as previously.

The topology of this area is different compared to the previous case. In this area, the depth gradually goes from shallow to deep, seen in 4.25. I think that the way the direction line is defined here, do not work well with this type of environment. There is no singular deepest point to move towards, since the deepest point is a whole area.

5.2.2 Results

In this case I did the five following experiments as previous.

1. Planning using straight survey lines - constant spacing: This experiment had a run-time of 37 minutes, a total coverage of 99.69%, a total overlap of 5014.5% and a total complete path length of 71853.1m.

2. Planning using straight survey lines - adaptive spacing (maximum): This experiment had a run-time of 4.28 minutes, a total coverage of 70.05%, a total overlap of 0.05% and a total complete path length of 12817.1m.

3. Planning using straight survey lines - adaptive spacing (average): This experiment had a run-time of 10 minutes, a total coverage of 95.78%, a total overlap of 8.08% and a total complete path length of 18463.5m.

4. Planning using straight survey lines - adaptive spacing (minimum): This experiment had a run-time of 13 minutes, a total coverage of 99.67%, a total overlap of 30.65% and a total complete path length of 23288.7m.

5. Planning using survey paths - adaptive spacing (average heading): This experiment had a run-time of 1.4 minutes, a total coverage of 73.52%, a total overlap of 0.05% and a total complete path length of 13750.8m.

Looking at the results above, in this case, experiment 3-4 yielded the best results. Once again experiment 4 is a straight upgrade compared to experiment 1 in terms of everything. Experiment 3 also did well in that same sense. Experiment 2 and 5 performed a lot worse in this scenario. From observation, overlap across the board is a lot less except for experiment 1. In this case, experiment 5 uses a different configuration as well.

This is a result of choosing a not ideal direction and initial survey line for this kind of environment. The approach for it has to reworked to be able to properly deal with this kind of environment, or it could be manually defined, same as the survey boundary.

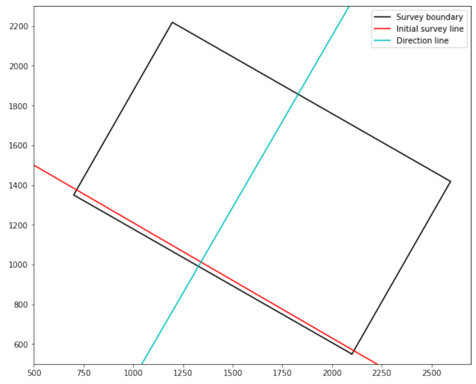


Figure 5.1: Ideal direction and initial survey line for this environment

With a proper direction and initial survey line in this type of environment, e.g. 5.1. Where the initial survey line is parallel to the coastline, and the direction line is perpendicular to the shoreline. I would expect experiments 2-5 to achieve over 90% coverage and a small amount of overlap. I think that experiments 2 and 5 would achieve the best results in terms of coverage, overlap and path length. Using this config, I would also expect that the spacing for experiments 2-5 will gradually increase as the depth gradually increases.

5.3 Algorithm Performance

Looking at the run-times in the results for both of the cases, I do not think they are consistent times considering the runs were done in a Jupyter Notebook on a computer that had a lot of different background processes going on.

It is hard to say something about the time complexity or the completeness of the algorithm because of how often it breaks because of the issues mentioned in 5.5.5. It was difficult to try and measure it.

The run-time of the algorithm drastically increases when either the resolution of the data set increases, step sizes for approximating survey lines and creating depth lines decreases, or when the depth is very high which increases the maximum sonar swath length 3.5.1, which directly increases how much depth data is retrieved.

In its current state, the algorithm is not robust enough to say anything about completeness. It requires a lot of re-factoring and rework to make it work properly. At this point of time, a lot of temporarily band-aids fixed has been applied to make it function to some degree. That said, it works alright when using straight survey lines - both static and adaptive, but with adaptive survey paths it breaks a lot.

The module that determines coverage works well most of the time, but the module that deals with creating the complete survey path needs to be remade. It is the part of the algorithm that fails the most. The way the process works is outdated and flawed as well, but there was not time to rework it.

5.4 Answering Research Questions

1) Can CPP be used to solve this problem?

The short answer is yes, according to the literature review and what was written previously in 2.4.2, 3.3. CPP is an umbrella term that encapsulates a lot categories, topics and methods. There is guaranteed to be techniques and methods that will be able to solve this problem, without knowing too much beforehand. Just as Artificial Intelligence can be applied in some form or another to likely solve any problem.

2) How can Coverage Path Planning be used to solve it?

As covered in the literature review, and mentioned in 3.3. There is a lot of applications for different domains. Even though water-based applications is in the minority with air. A lot of it can be generalized and used in any domain. For this thesis, I decided to take inspiration from the work by Galceran and Carreras (2012b), but do my own spin and interpretation of it, which is covered throughout Chapter 3. The approach used in this work has a lot of overlap, and the method used is what I imagined initially how I would do it before the literature review. It only felt natural to use the same approach.

3) How can the proposed solution be compared to the current methods employed by Maritime Robotics?

At the point of time of the beginning of this thesis. The current method employed was to either manually plot the path on a map, or insert a template of a path that could be scaled. I tried to recreate this approach by simply planning a path that had a constant spacing between the survey lines. The spacing would either be chosen randomly or be decided based on some data. This would be the base-line that I would compare my solution to. The metric used for comparison is coverage, overlap and length of complete path, where coverage is the priority.

Ideally it would the best to compare it to a previous survey performed by Maritime Robotics, but it was never followed up on.

4) What are the benefits of using the proposed solution? Advantages / disadvantages?

By only looking at the proposed solution and not the implementation or execution of it.

Advantages
1) The complete survey path is planned automatically, no manual labor involved.
2) While keeping the same amount (or close to) theoretical coverage. The amount of overlap is less, together with length of the complete path being shorter.
3) The solution uses simple and straightforward ideas.

Disadvantages
1) It is considered offline at this point of time, data of the environment is needed beforehand.
2) Complex environments and obstacles is not considered.

3) ASV dynamics is not considered.

5.5 Deviations and Issues

Throughout the thesis, several topics were cut due to time constraints. This was either the result of being overambitious or spending a lot more time than initially planned on other topics or issues. Some topics were also cut because of issues not being fixed in time.

5.5.1 Autonomous Surface Vehicle

This was one of the main topics in the scope in 1.3, and it was cut due to time restraints. The agent that supposedly is going to traverse the path is a surface vehicle, but it is considered to be a perfect agent. The dynamics of the ASV is not included at all in implementation.

The initial idea was to include the dynamics of the agent in some form or another, so the agent has the ability to actually traverse the planned path. The dynamics would either be included during path-planning or in post-processing. During planning by adjusting sharp angles during a survey path, or in the transition between survey paths. Or included during post-processing in terms of path smoothing.

Including dynamics could have also opened the ability to use energy/fuel consumption as a metric for comparison. By establishing a model of the ASV, so that an estimated consumption could be retrieved for a given path.

5.5.2 Cellular Environmental Decomposition

This was something I initially wanted to include in order to introduce a bit more complexity by adding obstacles. In the beginning I managed to implement a method for boustrophedon cellular decomposition, by simply implementing the pseudo-code from Choset and Pignon (1997). I had a lot of issues trying to make the proposed algorithm work in an area without obstacles, so I thought it was not appropriate to introduce more complexity, Even if I wanted to attempt to integrate it, it would require a lot more testing and likely to deal with a lot of unforeseen issues.

5.5.3 Density Map of Overlap

The method I use to visualize overlap in 3.26 only gives an indication on where overlap is and not how often it occurs. I wanted to visualize the occurrence of overlap by using a density map of sorts because I thought it would give more value. I initially implemented something, but I ran into some issues I were not able to solve in time.

5.5.4 3D Visualisation

Creating a 3D visualisation and simulation in Unity3D was not a major priority, but I attempted to implement it. The reason I wanted to do a 3D visualisation was because I wanted to have another layer of validation by simulating an agent following the planned path, and gradually coloured the seabed when the sonar covered it. In Unity3D I could also easily apply a more realistic behaviour to the agent, and see how good it would follow the path.

In Unity3D I initially loaded in the appropriate depth and path data stored in .csv files. I would then manually generate a mesh from depth data. The agent would follow the path, and when the sonar attached to it, intersected with the mesh. I would access the appropriate triangles in the mesh and paint them. This approach was not scalable at all, and it had issues creating the mesh from large environments. The method used for painting the mesh had serious performance issues as well.

I had another approach I wanted to attempt, but I never managed try it and test it properly. For the other approach I decided to convert the depth data into a grey-scale height map that could be used for automatic terrain generation. I then planned to apply a texture to the generated terrain, and when the sonar intersected with the texture, the appropriate pixels in the texture would be coloured.

5.5.5 Common Issues

During implementation and running of the algorithm. I ran into a lot of common and unforeseen issues. I think that a lot of these issues could have easily been detected early if I did not do the majority of the testing and validation with generated or super down-sampled data sets.

Stopping Conditions

The stopping conditions were very inconsistent and most of the time they were never met. To bypass this, I ended up encapsulating parts of the code in try-catch statements, and store relevant data on a loop-by-loop basis. When an exception were thrown, the run is interrupted and all data stored up to that point is returned. In most cases when an exception is thrown, it is because is should have ended earlier, but a stopping condition were never met. Due to it not stopping earlier, in some cases, unnecessary data is stored that needs to be removed. It not removed, it has a tendency to break the next processes, such as determining coverage and complete survey path. This issue results in some manual labor to determine whether the data is good to go.

Losing Continuity

As mentioned in 3.2, I am mostly working with Shapely Points, LineStrings and Polygons. In most of the time the LineStrings and Polygons objects needs to be continuous, whole and clean. If a LineString becomes a MultiLineString because it is not continuous, a lot of methods tied to LineString objects become unavailable. Merging of LineString objects become impossible as well, which I do a lot.

Losing continuity has probably been one of the most common issues I have encountered, and a lot of steps were introduced to prevent it. At any point during the run, if a LineString lose continuity or a merge fail because of it, the run will crash. To prevent it, regular checks are put in place, together with simplifying, cleaning and rounding down trailing decimals.

This problem usually occurs when running the algorithm for path-planning with adaptive survey paths using average heading, in 4.19, 4.39, or when creating the complete survey path following the method in 3.5.1 for all versions of the algorithm.

When I did the initial testing on generated and super down-sampled data sets I rarely had this issue, but it started to often occur when I increased the resolution and dimensions. In hindsight, a better solution for this issue would probably be put in place if I initially used more proper data. I also believe that there are some steps that does not require continuity, but that has not been properly investigated.

Self-Intersections

LineStrings or Polygons crossing over itself, or in other words self-intersection was a issue that resulted in a lot of problems, it also took a while to realise how big of an issue it was. In the implementation, I use a lot of set theory operations, such as union, intersect and difference. These issues mostly occurred when using these operations. Such as during path-planning with adaptive survey paths in 4.19, 4.39, but it would also appear in any version of the algorithm because the survey boundary could have an self-intersection. During when determining coverage and overlap, and creating the complete survey path. In some cases it was barely noticeable by eye because of the scale.

I managed to put in some temporary fixes to prevent self-intersections when dealing with Polygons, but not for LineStrings, so the problem is still mostly present. I also realised that in the work of Manda et al. they implemented a method to remove self-intersections from the path, but I did not bother to do it initially because of the following reason. The data they worked with has a lot more variety in topology and depth compared to what I used. So in my case, the depth topology is predictable, and I expected a more clean path. I also did not know that the set operations could not be applied to Polygons or LineStrings that crossed it self. A big slip up on me side that caused a lot of grief, obvious in hindsight.

Chapter 6

Conclusion

6.1 Results

In conclusion, from the two environments presented in Chapter 4. The proposed algorithm performs much better compared to an algorithm that utilizes constant spacing between survey lines. The results suggests that the same amount of coverage is kept, but a reduction in overlap and path length is achieved dependant on which version is chosen. The environment plays a large role in performance, and there needs to put a lot of more emphasis on determining survey strategies, starting position and which direction to sweep or move towards to.

In the first case 4.1, a direction and initial survey line that managed to yield good results was found. It was not the same case in 4.2. Due to the topology of the environment, my solution found a direction line and initial survey line that was not ideal for the environment at all.

There also needs to be put a lot more emphasis on deciding parameters for both the data set and the algorithm. Parameters such as data set resolution, step size for approximation, step size for retrieving depth data, and so on. These parameters are vital for both performance the algorithm running to completion, together with finding good general stopping conditions.

In terms of time complexity and completeness. It is hard to determine these factors due to the current implementation not being robust enough. The current iteration has a lot of issues, covered in 5.5.5. Several temporarily fixes needs to become well thought out good permanent fixes.

The lack of robustness is the result of over-ambitiousness and a lack of focus, and due to this the execution suffered a lot. In hindsight, just focusing on specific version of the algorithm, such as either adaptive spacing or adaptive survey paths. Then making sure that it runs well, is solid and robust. Then re-introduce and implement the cut topics in 5.5.

Even though execution of the implementation is lacking. The approach and the idea of the algorithm works, which is proven in both this thesis, and from the work it is inspired from.

6.2 Contribution

By taking inspiration from Manda et al., and implementing the algorithm as I interpret it. An algorithm for offline planning a complete survey path inside an obstacle free survey boundary is proposed. This algorithm considers the depth of the area to reduce overlap and shorten down the complete path while keeping the amount of coverage high. The algorithm follows simple and straightforward ideas, and is built mainly using Shapely.

6.3 Future work

6.3.1 Algorithm

- Rework and refactor the algorithm to increase robustness. Implement solutions to deal with the issues covered in 5.5.5.
- Create a solution that in general better analyzes the environment and determines direction and initial survey line.
- Set up a solution for selecting and optimizing parameters - for both data set and algorithm.
- Introduce optimization during planning by e.g. adjusting overlap factor on a per lap basis (adaptive straight survey lines), or on a per vertex basis (adaptive survey path).
- Determine stats such as time complexity, optimality and completeness for the algorithm.
- Look into doing circular motion instead of back-and-forth motion.
- Create a solution for an online version of the algorithm.

6.3.2 Environment

- Run algorithm on environments with more variety in depth and seabed topology.
- Create solution for dealing with environments where information/data beforehand is lacking. Look into doing assumptions, interpolation or prediction from few data points to generate the gaps.
- Introduce more complexity into the environment by including obstacles, and then proceed to solve it with cellular environment decomposition mentioned in 5.5.2.

6.3.3 Autonomous Surface Vehicle

- Include dynamics of the ASV into the algorithm during planning by e.g. adjusting angles (adaptive survey path).
- Include dynamics of ASV in post-processing by e.g. path smoothing.

- Include dynamics of the ASV to determine transitions between survey lines/paths.
- Optimise in terms of energy/fuel usage, get estimation from ASV model from complete path.

6.3.4 Visualisation

- Create a 2D density map to better visualize overlap, as explained in 5.5.3.
- Create 2D simulation and visualisation in Unity3D, as explained in 5.5.4.

Bibliography

Acar, U. E., Choset, H., Rizzi, A. A., 2002. Morse decompositions for coverage tasks. In: The International Journal of Robotics Research 21.

Bernhard, A., 2018. The quest to map the mysteries of the ocean floor. Last accessed 1 December 2018.
URL http://www.bbc.com/future/story/20180404-the-quest-to-map-the-mysteries-of-the-ocean-floor

Blyth-Skyrme, V., 2011. Current application and future needs of the noaa pacific islands fisheries science center's coral reef ecosystem division seabed mapping program.

Boyd, S. K., Coggan, R. A., Birchenough, S. N. R., Limpenny, D. S., Eastwood, P. E., Foster-Smith, R. L., Philpott, S., Meadows, W. J., James, J. W. C., Vanstaen, K., Soussi, S., Rogers, S., 2006. The role of seabed mapping techniques in environmental monitoring and management. In: Science Series Technical Report no. 127.

Cao, L. Z., Huang, Y., Hall, L. E., 1998. Region filling operations with random obstacle avoidance for mobile robots. In: Journal of Robotics Systems.

Choset, H., 2001. Coverage for robotics - a survey of recent results. In: Annals of Mathematis and Artificial Intelligence. Netherland.

Choset, H., Pignon, P., 1997. Coverage path planning: The boustrophedon decomposition. In: International Conference on Field and Service Robotics.

Correll, N., 2016. Introduction to Autonomous Robots. Magellan Scientific.

Englot, B., Hover, S. H., 2012. Sampling-based coverage path planning for inspection of complex structures. In: Proceedings of the Twenty-Second International Conference in Automated Planning and Scheduling.

Englot, B., Hover, S. H., 2016. The application of optimal search to marine mapping. In: OCEANS 2016 MTS/IEEE Monterey. Monterey, CA, USA.

Fang, C., Anstee, S., 2010. Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle. In: OCEANS'10 IEEE SYDNEY. Sydnet, NSW, Australia.

Frischkorn, K., 2017. Why the first complete map of the ocean floor is stirring controversial waters. Last accessed 1 December 2018.
URL `https://www.smithsonianmag.com/science-nature/first-complete-map-ocean-floor-stirring-controversial-waters-18096399`

Galceran, E., Carreras, M., 2012a. Coverage path planning for marine habitat mapping. In: 2012 Oceans. Hampton Roads, VA, USA.

Galceran, E., Carreras, M., 2012b. Efficient seabed coverage path planning for asvs and auvs. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura, Portugal.

Galceran, E., Carreras, M., 2013a. Planning coverage paths on bathymetric maps for in-detail inspection of ocean floor. In: 2013 IEEE International Conference on Robotixs and Automation (ICRA). Karlsruhe, Germany.

Galceran, E., Carreras, M., 2013b. A survey on coverage path planning for robotics. In: Robotics and Autonomous Systems 61.

Galceran, E., Nagappa, S., Carreras, M., Ridao, P., Palomer, A., 2013. Uncertainty-driven survey path planning for bathymetric mapping. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. Tokyo, Japan.

Jung, Y.-S., Lee, K.-W., Choi, H. M., Lee, H.-B., 2009. An efficient underwater coverage method for multi-auv with sea current disturbances. In: Regular Papers Robotics and Automation.

Kapctanovic, N., Miskovic, N., Tahirobic, A., 2018. Information gain-guided online coverage path planning for side-scan sonar survey missions. In: 2018 26th Mediterranean Conference on Control and Automation. Zadar, Croatia.

Khan, A., Noreen, I., Habib, Z., 2017. On complete coverage path planning algorithms for non-holonomic mobile robots: Survey and challenges. In: Journal of Information Science and Engineering.

Kim, H. D., Hoang, G., Bae, M. J., J, K. W., Yoon, M. S., Yeo, K. T., Sup, H., Kim, B. S., 2014. Path tracking control coverage of a mining robot based on exhaustive path planning with exact cell decomposition. In: 2014 14th International Conference on Control, Automation and Systems.

Lehtla, T., 2008. Introduction to Robotics. Dept. of Electricial Drives and Power Electronics.

Liu, Z., Zhang, Y., Yu, X., Yuan, C., 2016. Unmanned surface vehicles: An overview of developments and challenges. In: Annual Reviews in Control. 41.

Manda, D., May-Win, T., Armstrong, A., 2016. Depth adaptive hydrographic survey behaviour for autonomous surface vessels. In: OCEANS 2015 MTS/IEEE Washington.

MaritimeRobotics, 2018. The portable usv system. Last accessed 3 December 2018.
URL https://maritimerobotics.com/mariner-usv/otter/

Matsuura, T., Kimura, T., 2017. Covering salesman problem with nodes and segments. In: American Journal of Operations Research, 2017.

Moravec, P. H., Elfes, A., 1985. High resolution maps from wide angle sonar. In: Conference: Robotics and Automation. Proceedings. 1985 IEEE International Conference.

Morin, M., Abi-Zeid, I., Y, P. R., Quimper, C. G., 2013. A hybrid algorithm for coverage path planning with imperfect sensors. In: IEEE/RSJ International Conference on Intelligent Robots and Systems.

NOAA, 2018a. National ocean service (nos) office of coast survey u.s bathymetric and fishing maps. Last accessed 3 December 2018.
URL https://www.ngdc.noaa.gov/mgg/bathymetry/maps/nos_intro.html

NOAA, 2018b. What does an oceanographer do? Last accessed 3 December 2018.
URL https://oceanservice.noaa.gov/facts/oceanographer.html

NOAA, 2018c. What is hydrography? Last accessed 3 December 2018.
URL https://oceanservice.noaa.gov/facts/hydrography.html

NOC, 2018. Autonomous surface vehicles. Last accessed 3 December 2018.
URL https://www.noc.ac.uk/facilities/marine-autonomous-robotic-systems/asv

OER, 2006. Learning Ocean Science through Ocean Exploration. Explore.

Pandian, P. K., Ruscoe, J. P., Shields, M., Side, J. C., Harris, R. E., Kerr, S. A., Bullen, C. R., 2009. Seabed habitat mapping techniques: An overview of the performance of various systems. In: Mediterranean Marine Science.

Paull, L., Saeedi, S., Seto, M., Li, H., 2013. Sensor- driven online coverage path planning for autonomous underwater vehicles. In: IEEE/ASME TRANSACTIONS ON MECHATRONICS.

Pratama, S. P., J, K. W., Kim, K. H., Yoon, S. M., Yeu, K. T., Yon, S., 2015. Path planning algorithm to minimize an overlapped path and turning number for an underwater mining robot. In: 2015 15th International Conference on Control, Automation and Systems.

Tommy, J, L., 2017. Autonomous robot coverage paths.

Vasconcelos, C. J., 2015. Design of autonomous surface vessels.

Waanders, M., 2011. Covering path planning for mobile cleaning robots. In: 15th twente student conference on IT.

Williams, B. S., Wilson, T., 2017. Adaptive path planning for depth-constrained bathymetric mapping with an autonomous surface vessel. In: Journal of Field Robotics.

Wong, C. S., MacDonald, A. B., 2003. A topological coverage algorithm for mobile robots. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453). Las Vegas, NV, USA.

Zhu, D., Tian, C., Sun, B., Luo, C., 2018. Complete coverage path planning of autonomous underwater vehicle based on gbnn algorithm. In: Journal of Intelligent and Robotic Systems.

Appendix

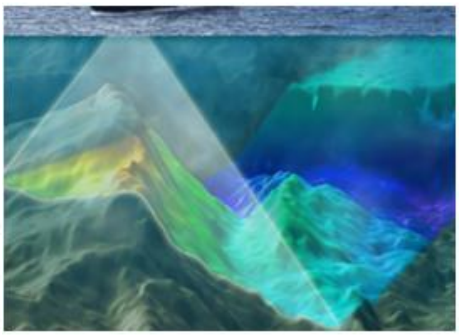
Appendix A: Project proposal

Date: 04.12.18
Department: Department of ICT and Engineering

Coverage Path Planning for Seabed Mapping using Autonomous Surface Vehicles

Introduction

This project is a collaboration with Maritime Robotics in Trondheim. The topic is about using an Autonomous Surface Vehicle (ASV) for seabed mapping using sonar. The ASV must follow a path that is considered safe and efficient and avoid obstacles. For it to be efficient, the vehicle must follow a path that maps the seabed without overlap. This means the properties of the sonar and the dynamics of the ASV must be taken into consideration. Coverage Path Planning (CPP) will be utilized. CPP is a variant of Path Planning where the goal is to cover a surface instead of simply finding a path through it. CPP is fundamental in applications such as: cleaning, agriculture, lawn moving, demining and more.


Left: USV Otter (maritimerobotics.com) Right: Visualization of seabed mapping with sonar (dragonfishoffshore.com)

Motivation

Maritime Robotics is a company that believes it is important to follow the technological development and solving problems like these will contribute to better autonomous products. The company also has something called the Vehicle Control Station (VCS), which is a software that allows planning and monitoring of missions and data acquisition. Implementing automatic path planning and better path planning algorithms will increase the value of this product by decreasing the amount of human labor and make it easier to plan missions. Seabed mapping is also a repetitive task that is ideal for automation.
More generally speaking, mapping the seabed is important for many reasons. It is important for safe sea traffic, increasing safety in expeditions, geologists for undercovering minerals and resources, archeologists for finding sunken treasures or old civilizations, and more. It can even help with revealing important information about the climate and determine weather patterns and ocean currents. By being able to do these mapping operations better and less expensive benefits many areas and sciences.

Scope

The scope of this thesis will be path planning, autonomous surface vehicles and oceanography which seabed mapping falls within. Looking at different CPP algorithms for seabed mapping and review different aspects to see if one or more fits my requirements. Defining the level of abstraction and assumptions. Starting with simple models, scenarios and cases and incrementally add complexity and details. Use cases and compare my solution versus Maritime Robotics solution in terms of efficiency and cost.

Objectives
Create a solution for automatic path planning for efficient autonomous seabed mapping.

- Research Question 1: Can I use Coverage Path Planning to solve this problem?
- Research Question 2: How can I use Coverage Path Planning to solve it?
- Research Question 3: How can I compare my proposed solution to current methods used by Maritime Robotics?
- Research Question 4: What are the benefits of using my proposed solution? Advantages / disadvantages?

Milestones:
Tasks:

1. Reviewing state-of-the-art CPP algorithms/methods for seabed mapping.
2. Modelling: ASV, sonar, environment, determining levels of abstraction and simplifications.
3. Running simulations: testing and evaluation of algorithms / models.
4. Test solution in different scenarios (complex environment (surface / seabed), obstacles, etc.).
5. Testing solution versus cases from Maritime Robotics and compare.
6. Testing solution in the real-world.
7. Writing

The work scope may prove to be different than initially anticipated. Subject to approval from the supervisor, topics may be added or deleted from the list above or reduce in extent.

The thesis shall be written as a research report, following the template given in Inspera. During preparation of the text, the candidate should make efforts to create a well-arranged and well-written report. To ease the evaluation of the thesis, it is important to cross-reference text, tables and figures. For evaluation of the work a thorough discussion of results is needed. Discussion of research method, validation and generalization of results is also appreciated.

The thesis shall be submitted in electronic version according to standard procedures (.PDF or .ZIP files). Instructions are found on the NTNU website (Inspera) and on Blackboard. In addition to the specified tasks, an A3 poster should be prepared and delivered together with this proposal, and a conference paper will be handled at the end of the research.

After finalizing and delivering the thesis, it must be sent a copy to the supervisor(s).

Karl Eirik Aasen
MSc. Simulation and Visualization
karleaa@stud.ntnu.no

Robin T. Bye
Associate Professor
Supervisor

Ottar L. Osen
Associate Professor
Head of study program automation
CO-Supervisor