Autonomous navigation for a depth measuring vehicle in an unknown naval environment

Lubomir Kotzev
lubomir@kth.se

Niklas Rolleberg
nrol@kth.se

May 30, 2015

Supervisor: Petter Ögren
Abstract

The accuracy of depth measurements in shallow parts of the Stockholm archipelago nautical charts is relatively low at present. In order to improve it, an autonomous depth measuring craft equipped with three downward facing sonar sensors is being developed at KTH.

In this report, a navigation algorithm that does not require any prior knowledge of the target environment is proposed to be implemented in the craft. It consists of a predefined sweeping pattern, a contour following that only relies on sensor data, and a way of identifying and scanning missed partitions. Performance and the impact of sensor inaccuracies in addition to environmental factors, such as wind and waves, are investigated through simulations.

Results from simulations show that the proposed navigation algorithm is able to complete scans of complex environments. However, the uncertainty in sensor readings is proven to have a large influence on performance.

Finally, further improvements to the algorithm are proposed and the realism of the simulation is discussed.
Sammanfattning

Noggrannheten av djupmätningar i grunda delar av Stockholms skärgårds sjökort är för tillfället bristfällig. För att förbättra dessa utvecklar KTH en autonom djupmätningfarkost, utrustad med tre nedåtriktade sonarer.

I denna rapport föreslås en navigeringsalgoritm för denna farkost. Algoritmen behöver inte någon tidigare kännedom om området och består av ett fördefinierat svepande sökmönster, en metod som endast använder sonardata för att följa land och ett sätt att identifiera samt täcka ej besökta områden. Prestanda och påverkan av störningar i form av sensorosäkerheter och miljöfaktorer, som vind och vågor, undersöks genom simuleringar.

Resultat från simuleringarna visar att den föreslagna algoritmen klarar av att täcka komplexa områden. Däremot har störningar i sensordata stor påverkan på prestandan.

Till slut föreslås förbättringar och simuleringens realism diskuteras.
# Contents

1 Introduction 3  
   1.1 Motivation 3  
   1.2 Problem statement 3  
   1.3 Related work 4  
   1.4 Background 5  
      1.4.1 Theory 5  
      1.4.2 Coverage approaches 6  
         Predetermined search patterns 6  
         Real time approach 7  

2 Proposed approach 8  
   2.1 Test environment 8  
      2.1.1 Map generation 8  
      2.1.2 The boat 9  
      2.1.3 The controller 9  
   2.2 Navigation 9  
      2.2.1 Decomposition of the search area 10  
      2.2.2 Detecting land 10  
      2.2.3 Hybrid coverage approach 11  
      2.2.4 Identifying uncovered regions 12  
      2.2.5 Travelling to a new region 12  
      2.2.6 Compensating for wind and currents 13  
      2.2.7 Reducing noise from sensors and waves 14  
   2.3 Implementation 15  
      2.3.1 Overview 15  
      2.3.2 Scanning triangles 15  
      2.3.3 Identifying new regions 17  

3 Results 18  
   3.1 Coverage approach comparison 18  
      3.1.1 Choice of coverage approach 20  
   3.2 Wind correction 21  
   3.3 Impact of waves and measurement noise 21  
   3.4 Complete solution results 22  
      3.4.1 Step by step 22  
      3.4.2 Robustness and consisency 24
4 Discussion and Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Simulation realism</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>The choice of coverage approach</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>Evaluation of the complete solution</td>
<td>28</td>
</tr>
<tr>
<td>4.4</td>
<td>On the ethics of automation</td>
<td>28</td>
</tr>
<tr>
<td>4.5</td>
<td>Further work</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

The depth in large parts of the Stockholm archipelago is currently known with relatively low accuracy. In order to address this issue, vehicle engineering scientists at KTH have been given the task to build an unmanned depth-measuring craft. It will be equipped with three downward facing sonar sensors: one facing straight down and two placed at the front, aimed slightly forward and to the sides. Using these sensors the boat has to be able to perform a depth measuring survey of an unknown environment. The craft will have no prior knowledge about the environment, except for a polygon containing the area to be surveyed. This means that the craft has to travel around inside that polygon and take depth measurements until a given accuracy is achieved, much like an automated vacuum cleaner has to do when cleaning a room.

1.2 Problem statement

A crucial difference between the problem at hand and a regular automated vacuum cleaner application is the fact that a robotic vacuum cleaner usually detects obstacles by bumping into them, while the autonomous craft will have to detect obstacles in time, reduce its velocity and avoid collisions completely. This will have to be achieved by using input from the three sonars which can measure the distance to the ocean floor. One of these sonars will be aimed straight down while the other two, positioned at the front of the boat, will be aimed slightly forward. By reading data from the sonars, the navigation algorithm has to make its own assumptions about the ocean floor and take evasive action by following shore lines or navigating to previously covered areas.

The problem is further complicated by the complex and irregular shapes of islands, bays and coastlines, contrasting the simple shapes of a room and furniture. The craft has to be able to operate in all kinds of naval environments so a standard solution to the vacuum cleaner problem might not be sufficient. Environmental factors such as wind and waves need to be taken into account and their impact on the performance of the navigation algorithm has to be evaluated.

Since there is no prior knowledge about the environment, the boat also needs a way to keep track of, and navigate to uncovered subregions. To this end, a shortest path graph algorithm combined with a way to systematically split the search area into smaller subregions is required.
Finally, we summarize the problem statement as follows:

*Design a robust navigation algorithm that is suitable for depth surveys in unknown naval environments and evaluate its performance through simulations.*

### 1.3 Related work

The problem of path planning for coverage of a given area has been extensively studied and has multiple applications, such as automated vacuum cleaners and lawn mowers, seabed scanning for underwater vehicles, agricultural machines and minesweepers. For all these applications, there is a clear need to split the initial search area into a manageable amount of smaller regions or cells. This so called *decomposition* is another well studied field and can be performed in numerous ways, which can be roughly divided into two subcategories: offline and online algorithms. In an offline solution, complete prior knowledge of the environment is presumed and no adaptation is being performed during execution. For the purposes of the problem discussed in this paper, an online (also known as "sensor driven") approach needs to be implemented.

In their very popular work from 2002 [1], Acar & Choset demonstrate a way to incrementally construct a graph containing search cells, while simultaneously covering the area. The algorithm uses an omnidirectional array of sensors to detect connectivity changes in the free space in order to locate *Morse function critical points* [2], used as cell boundaries. A substantial upside to this approach is that it can be proven to encounter all critical points, and is therefore guaranteed to cover the entire area. A similar approach is utilized by Dugarjav et al. [3], using lasers to scan the immediate environment, which is assumed to be rectilinear. The scans are then matched with previous and reference scans to obtain an image of the environment. A sweeping search pattern is then combined with an online decomposition algorithm to cover the entire area.

For the purpose of mine countermeasures with an unmanned underwater vehicle, a slightly different approach to cell decomposition is implemented by L. Paul et al [4]. In this study, the authors propose an online approach using a hexagonal cell decomposition, where each hexagonal cell is marked with the distance to the currently occupied cell and also maintains a list of its neighbours. Each neighbour of the current cell is considered a branch with an entropy value, denoting how much uncovered area there is down that branch. Using this entropy function alongside expected information gain evaluations, the navigation algorithm makes decisions on whether to change its heading.

Simpler approaches include conventional coverage algorithms, such as the "Plane Sweep Algorithm" [5], used for automated vacuum cleaners. This type of algorithm is, as previously discussed, better suited to an environment containing simpler shapes, and their shortcomings are readily demonstrated by Acar & Choset [1]. Given an unknown and rather complex environment, naive approaches such as a standard sweeping pattern are at risk of missing entire areas situated behind narrow straits or otherwise unlikely to come across.

While the papers mentioned in this section do solve problems very similar to ours, none of them have the same limited sensor capabilities as the craft being described in our report. Combined with the need to adapt to the measured depth instead of just covering the area, this leads us to the conclusion that a new approach needs to be developed, based on the ones described above.
1.4 Background

In this section, we provide a brief overview of algorithms and approaches used in this study, followed by a short description of coverage approaches.

1.4.1 Theory

A* search algorithm  First described in 1968, A* is a shortest path algorithm [6]. It will find the shortest path from one node in a graph to another. In contrast to other shortest path algorithms like BFS-search and Dijkstra, A* uses a heuristic to guide the search towards the target. This allows the algorithm to look at fewer nodes and therefore be faster. A* works with both weighted and unweighted graphs. A weighted graph could for example be the road network with traversal times as weights. In order to find the fastest way from one place to another in that network the speed limits must be taken into account.

PID-controller  The purpose of a PID controller is to minimize the error between a reference signal and a measured value. PID stands for Proportional Integral Derivative. The PID regulator is an extension of the P regulator, which is a basic regulator with an output signal that is proportional to the error $\epsilon$ with a factor $K_p$ [7]. The simple P regulator has a few drawbacks, it is not able to remove the error completely (steady state error) and it has a tendency to make the system oscillate if $K_p$ is large. In order to address the problem with the steady state error, an integral term can be introduced, creating a PI regulator. The integral part will allow the regulator to reduce the steady state error to 0, but further increases the risk of oscillations. To deal with the oscillations, a derivative part can be added. This part will slow the system down when it changes quickly, which reduces oscillations. The final form of the PID controller is:

$$output = K_P \epsilon + K_I \int_0^t \epsilon \, d\tau + K_D \frac{d}{dt} (\epsilon)$$  (1.1)

Ear clipping algorithm  The ear-clipping algorithm splits a polygon into a set of triangles. The algorithm is called ear clipping because it creates the set of triangles by removing ears of the polygon. An ear of a polygon is a triangle formed by three consecutive vertices for which no other vertices of the polygon are inside the triangle [8]. An advantage of splitting a polygon into triangles is their guaranteed convexity, which simplifies calculations when trying to determine if a point is inside a triangle.

Gift wrapping algorithm  Also known as the Jarvis march after R. A. Jarvis, who published it in 1973, the gift wrapping algorithm is used to obtain the convex hull of a given set of points [9]. In order to initialize the algorithm, a point that is guaranteed to be on the convex hull is needed. Picking an extreme point, i.e. one that has the lowest x- and y- coordinates satisfies this condition. The algorithm then “wraps” the convex hull around the set of points by comparing angles and distances from the current hull point to all other points in the set, and choosing the appropriate one to add to the hull. It terminates when it is back at the initial point. In two dimensions, the process can be likened to wrapping a string around a set of pegs on a board.
1.4.2 Coverage approaches

The most efficient way, in terms of time and distance travelled, of covering an entire area is often by following a predetermined search pattern. The pattern can be optimized for the boats’ characteristics, e.g. if the boat has difficulties turning and keeping the speed up at the same time, a search pattern with as few turns as possible is preferred. The problem with using a predetermined search pattern in an unknown environment is that the boat will be unable to avoid obstacles while still staying true to the pattern.

Another way of covering the area is a real time approach: by analysing data from the sensors the boat autonomously decides on its actions. A simple example of this is choosing a heading and going in that direction until an obstacle is encountered. When an obstacle is detected, a new direction is chosen, and then the process is repeated. This approach is used in many vacuum cleaners.

We will consider different types of predetermined patterns and real time approaches.

Predetermined search patterns

Spiral pattern  The spiral pattern makes the boat run in a circle with an increasing radius [10], see figure 1.1. This method has the potential to cover a large area in a short time if there are no obstacles in its way. This approach is well suited for a boat that is able to keep a high speed while turning slowly.

![Figure 1.1: Spiral pattern](image)

Figure 1.1: Spiral pattern
Sweeping pattern  The boat will start at the top of the given region and travel from one side to another [10], see figure 1.2. When the boat has traversed the region it will turn and make another pass slightly lower than the previous one. This approach suites a boat with a small turning radius.

![Figure 1.2: Sweeping pattern](image)

Real time approach

Contour following  The contour following algorithm is our idea of following shorelines. Instead of keeping the boat at a certain distance from the shore, it keeps the boat on a contour line defined by a reference depth. To achieve this, the boat has to be aware of what way the seafloor tilts, in order to know which way to turn when it gets into deeper or shallower waters. In our case, this can be done by comparing data from the two front facing sonars. The magnitude of the turning angle, $\theta$, can then be calculated by using the difference between the measured and target depths. A way to achieve this is to use a PID controller (see 1.4.1). This concept is shown in figure 1.3

![Figure 1.3: PID controller concept for calculating $\theta$](image)
Chapter 2

Proposed approach

In this chapter, we describe our approach to the problem and the simulation environment built in order to test it.

2.1 Test environment

In order to simulate different algorithms and verify results, a test environment is constructed, using Java. The aim is to be able to write the code in a way that is as close to the real implementation as possible. It consists of four major parts: the generation of a random map, the craft itself, a real-time animation of the boat and environment, and a controller package, containing coverage and navigation algorithms. All of these parts are running in separate threads, to make the simulation architecture resemble the real application as closely as possible. In this section, a brief overview of the program structure is provided.

2.1.1 Map generation

Since the proposed solution has to be able to navigate and scan through all kinds of unknown naval environments, a random map generator is created. The maps are saved as files so that they can be reused to compare different approaches in the same environment. All generated maps are unique, which allows for testing the approaches in a wide variety of environments. Examples of such maps are shown in figure 2.1.

Figure 2.1: Random generated maps
2.1.2 The boat

The boat model is controlled by the input of a waypoint and a desired speed. It simulates the sensors, stores the data for the algorithm to collect and performs calculation of new sonar data at a predetermined frequency in order to mimic the update frequency of a real sonar.

The boat has three types of sensors to access: a sonar pointed straight down, another two pointed slightly forward and to the sides for navigation, and a GPS. The data for the sonars is obtained by checking the depth of the generated map at the current position. The GPS coordinates are just the boat's current coordinates in a 2D simulated coordinate system. In figure 2.2, the sonar configuration is shown. Each sonar gives the distance to the closest object within its field of view.

![Figure 2.2: The boat and its sonars. The sonar pointing straight down is used to measure the distance to the seabed and its values are stored for later analysis. The pair of sonars pointing forwards and to the sides are only used for navigation.](image)

The real life sensors will never be able to achieve the consistent and exact measurements that the simulated virtual sensors do. In order to mimic sensor data inaccuracies, a uniformly distributed random sensor noise factor $\phi$ is introduced. Thus, the final value from the sonar $S_{out}$, is:

$$S_{out} = \phi S_{actual}, \quad 0.9 \leq \phi \leq 1.1$$

(2.1)

2.1.3 The controller

This part of the test environment acts as the "brain" of the boat and collects sensor data from the boat object. It contains instructions on how to decompose the area into cells, avoid collisions with land, perform coverage and navigate the free space. It also ensures that the entire given area is surveyed before stopping. By utilizing this set of capabilities, it outputs a waypoint for the boat object to travel to.

The long-term goal of this project, as described in section 1.1, is to provide a good base for a controller which can be implemented in the real boat architecture.

2.2 Navigation

In this section, the different components of the navigation and coverage approaches are explained in further detail.
2.2.1 Decomposition of the search area

In order to keep track of the status of the current area being scanned and avoid exhausitively re-scanning a subregion, a cellular decomposition is introduced. Initially, the entire search area is discretized into a matrix containing a number of elements based on a desired resolution as shown in figure 2.3. Each of these elements contains information about its coordinates, a status (covered, uncovered, out of bounds, tentative land or unreachable), number of times visited, a list of its neighbouring elements, and a depth value. When visited more than once, a mean value for the depth is assigned to the element. At the starting point of the algorithm, the initial area to be surveyed is split into triangles, using the ear clipping method described in section 1.4.1. A scan of each of these triangles is then attempted, further explained in section 2.3. When this process is completed, a new set of triangles is generated by identifying all continuous subregions that are still unvisited, splitting them into triangles and starting surveying again. If there are no such subregions left, the scan is considered complete.

![Image](image.png)

Figure 2.3: Discretizing the area into elements. The parameter $\delta$ is calculated from the desired resolution.

2.2.2 Detecting land

The controller behaviour is solely based on data from the sonars, meaning that exact detection of the coastline is impossible. A simple way to estimate the occurrence of land and avoid it is to set a minimum depth limit for the boat. Every time the boat passes this limit, it has to either reverse or change heading. Another, more advanced way is to use depth data from several consecutive measurements and calculate the change in depth per length unit and use this value to approximate the distance to land. If the boat gets too close to land it has to take action to avoid it. Due to its simplicity, the method of setting a minimum depth and considering everything shallower than that depth "land" is selected and implemented.

The controller adapts the speed to the water depth. If the boat is in deep water the risk
of encountering land is marginal and the boat can travel at top speed, but at shallower depths the risk is much greater and the boat has to reduce its speed in order to reduce the distance between measurements to gather more data about the seafloor. This adaptation also gives the boat more time to react if it gets into too shallow water, and reduces potential damage if the boat were to hit an undetected obstacle.

2.2.3 Hybrid coverage approach

When using solely a predetermined search pattern, the algorithm has no way of knowing what to do upon encountering an obstacle. A pure sensor driven algorithm, however, lacks purpose and needs a randomly generated waypoint in order to progress.

Therefore, we propose a hybrid approach between a predetermined pattern and a real time approach. It uses a predetermined search pattern as a base, and when an obstacle is detected, the contour following algorithm (see section 1.4.2) will take over control. While following land, unscanned elements adjacent to the boat are marked as not accessible. When the obstacle is avoided, the boat will continue following its predetermined pattern.

Below, the two combinations that we will investigate are described.

Spiral pattern with contour following. This hybrid approach is based on the spiral pattern described in section 1.4.2. The boat travels in a spiral until it detects land, and then uses the contour following algorithm (section 1.4.2), until the boat reaches the desired distance from the spiral’s center, where the spiral pattern is resumed, as illustrated in figure 2.4. The scanning of a region is considered done when a circle defined by the radius of the spiral completely encloses the region.

Figure 2.4: Spiral hybrid pattern

Sweep pattern with contour following. In this approach, the boat follows the sweep pattern described in section 1.4.2 until it encounters land, upon which it switches to contour following (section 1.4.2) and follows the shoreline until it either clears land or
reaches a position where it can continue to travel back by following the next virtual line as illustrated in figure 2.5. The scanning of a region is considered done when the boat has reached the highest or lowest point (depending on upward or downward oriented pattern) of the region or when the boat is stuck (checked by measuring the time spent in a particular location).

Figure 2.5: Sweeping hybrid pattern

2.2.4 Identifying uncovered regions

Even with a hybrid search pattern approach, there will be areas left uncovered, as illustrated in figure 2.6. The set of uncovered elements in each such area serve as a basis for a new polygon. The boundaries of each set are first extended to include their previously scanned neighbours, and then the entire set is sent to the gift wrapping algorithm (see section 1.4.1) in order to obtain the convex hull of the new polygon. As previously explained in section 2.2.1, the new polygon is then split into triangles, which in turn are added to a set of areas to be covered.

Figure 2.6: Areas left uncovered

2.2.5 Travelling to a new region

In order to travel to a point on the map, a graph representation containing nodes and edges is created, see figure 2.7.
When the current region being scanned is completed the boat has to find its way to a new region. This is achieved by calculating the cost to all available elements on region boundaries using the A* search algorithm (section 1.4.1) and picking the lowest one. The total travel cost to a destination is determined by the sum of all node transitions along the way, where each such transition cost is determined by the euclidean distance between the two, multiplied by a factor which depends on the depth. This is done in order to be able to keep the boat at deep water where it can maintain a higher speed, and to minimize the risk of running aground.

These capabilities can also be used to return to a given point after completing the entire survey.

2.2.6 Compensating for wind and currents

In order to be able to mimic a real naval environment, the occurrence of wind and currents is simulated by altering the craft’s position with respect to these forces. Due to this disturbance, the craft is not able to travel along an ideal straight line, as shown in figure 2.8.

To keep the boat close to the ideal path, the target has to be adjusted and changed over time. There are different techniques of solving this problem. One way is to set a temporary target on the ideal line for the boat to steer towards at a given percentage of the way left to travel. Another way is to set the temporary target at a specified distance ahead of the boat on the ideal line, shown in figure 2.9.

Since the technique of putting the temporary target at a specified distance ahead on the
line allows for better control of how far off the ideal path the boat will drift, it is chosen as the preferred approach to this problem and implemented.

By introducing $\vec{n}$ as the normalized vector from the start point to the target, $\vec{v}$ as the vector from start to the boat’s current position, and $\sigma$ as the distance denoting how far ahead the temporary target should be on the ideal path, the new temporary target $T_{temp}$ is calculated as follows:

$$T_{temp} = start + \text{proj}(\vec{n}) \vec{v} + \sigma \ast \vec{n}$$  

(2.2)

Figure 2.9: This simple technique acts as a control system for counteracting wind and waves.

2.2.7 Reducing noise from sensors and waves

The navigation algorithm is dependant on accurate measurements in order to be able to perform contour following. Noise in readings from the downwards facing sonar, due to waves and hardware inaccuracies, will make it harder to stay on the reference depth. Noise in the forward facing sonars can make it hard to determine which way the ocean floor tilts.

In order to reduce noise, a mean value of the last 10 readings is used, instead of just the latest one. By reducing the speed at which the boat travels, more data is collected and this method becomes more effective. The PID controller is tuned to further improve the behaviour of the contour following in conditions where there is a lot of noise. This is done by reducing the proportional constant, thus making the controller react more slowly to changes.
2.3  Implementation

In this section, an overview of the entire navigation and coverage algorithm is provided, followed by a more detailed explanation of certain key features.

2.3.1  Overview

A simplified description of the entire process is shown in figure 2.10. Initially, input data is provided. It consists of a polygon expressed as a set of 2D points and a resolution parameter representing the desired distance between measurement points. The resolution parameter is used to discretize the area into a grid, as explained in 2.2.1. In the next step, the polygon is split into triangles using the ear clipping algorithm (see section 1.4.1). The resulting triangles are saved in a list. The algorithm then attempts to scan each of these (explained in more detail in section 2.3.2). The status of all elements is then checked, in order to determine if the survey is complete. If there are uncovered regions left, these will be converted to new polygons (section 2.3.3), triangulated and added to the list.

![Figure 2.10: Basic structure of the algorithm](image)

2.3.2  Scanning triangles

When new regions have been identified and split into triangles, the algorithm has to decide which triangle to scan. This process is shown in figure 2.11.
Algorithm 1 Find closest triangle and point on boundary

1: for all Triangles do
2: \hspace{1em} for all Element on triangle boundary do
3: \hspace{2em} Use A* to calculate the cost if element can be reached
4: \hspace{1em} end for
5: end for
6: if path found then
7: \hspace{1em} Find element and triangle with lowest cost
8: \hspace{1em} return triangle and path
9: else
10: \hspace{1em} return NULL
11: end if

The cost of a path is defined as the sum of the cost of all transitions, which is turn is calculated with the A* heuristic function. The weighted heuristic makes it more costly to travel in shallow waters.

Algorithm 2 A* heuristic

1: calculate euclidean distance between nodes
2: if node depth ≤ 1 then
3: \hspace{1em} return distance * 10
4: end if
5: if node depth ≤ 3 then
6: \hspace{1em} return distance * 2
7: end if
8: return distance

The boat will then travel to the triangle returned by algorithm 1 by following the coordinates in the path previously calculated with A*, and then scan it according to algorithm 3. If no path was found, all triangles are removed from the list and the algorithm proceeds with identifying new regions.
Algorithm 3 Scan triangle

1: Start Coverage approach
2: while Coverage approach is running do
3: store depth data in local map
4: if Coverage approach malfunctions or is stuck then
5: abort coverage approach
6: end if
7: end while
8: remove scanned triangle from list

2.3.3 Identifying new regions

The stages of the process of identifying new regions is shown in the list below.

1. Find continuous uncovered regions by checking status of the elements
2. Add already scanned neighbours to the boundary of each new region
3. Use the gift wrapping algorithm to find the convex hull of the new region.
4. Triangulate the convex hull
5. Loop through all triangles, remove those who are already scanned
Chapter 3

Results

In this chapter, results from simulations are shown. Initially, only the basic functionality of the spiral and sweeping patterns is tested. The best one is chosen and more results are presented, showing the performance and robustness of the complete solution.

3.1 Coverage approach comparison

In this section, two environments are used as a base of evaluation for the two different coverage approaches, the hybrid spiral and hybrid sweeping patterns. Only basic functionality is tested, meaning that no triangulation or identifying of uncovered regions is done. This is done in order to estimate the compatibility of the patterns with the rest of the final algorithm.

The coverage approaches are evaluated with respect to three key aspects: a coverage percentage, distance travelled and time consumption. The shape of uncovered regions is also taken into account.

In figure 3.1, the sweeping pattern is applied to a map containing mostly open water and a few small islands. Unmarked elements are coloured black, red represents unreachable, and the shades of blue represent different depths. As expected, there are some unmarked regions, whose shapes are fairly regular. This is a desired outcome for the convex hull and triangulation algorithms.

![Figure 3.1: Coverage by the sweeping pattern](image-url)
Figure 3.2 shows the same environment, but with the spiral pattern applied to it. Here, it is apparent that a lot of redundant scanning around the islands is being performed. Due to an exceeded time limit, a large part of the region is left unmarked. The shape of the region left unmarked would make the triangulation and convex hull algorithms less effective, because a substantial part of the elements in the resulting polygons would already be scanned.

In figure 3.3, a performance comparison is shown. It is clear that the sweeping pattern achieves a higher and more consistent coverage percentage. The reason for the apparent lower distance covered by the spiral pattern is the low speed maintained on shallow waters.

Figure 3.2: Coverage by the spiral pattern

Figure 3.3: Performance comparison
In the previous test, the coverage algorithms have been given ideal starting points: at the upper edge for the sweeping pattern and in the middle for the spiral one. That will not be the case in the final solution, since the boat will travel to a new area and begin the scanning from a point on the boundary of the area. Such a case is shown in figure 3.4.

![Figure 3.4: Comparison between spiral (left) and sweeping (right) patterns with a start point near the boundary. It is clear that the sweeping pattern behaves much better than the spiral pattern is this comparison. Not only does it achieve more consistent coverage along the sides of the polygon, but it also maintains its shape.](image)

3.1.1 Choice of coverage approach

With the results from the previous tests it is quite clear that the sweeping pattern is superior for this application. It provides good coverage even in areas with obstacles and the uncovered areas left behind have a suitable shape for the rest of the algorithm. It also proves to be better suited for starting a search on the boundary, which is what most commonly happens with all features implemented. Therefore the sweeping pattern is chosen as the coverage approach for the complete solution and we conclude that the spiral pattern is unsuitable for this application.
3.2 Wind correction

In this section the performance of the wind and current compensation algorithm proposed in section 2.2.6 is presented.

Figure 3.5: No compensation used in the image to the left, and wind compensation turned on in the image to the right. The direction of the wind is facing down in the images.

It is clear that the wind compensation does make the boat travel in a straight line as intended.

3.3 Impact of waves and measurement noise

In this section results showing the impact of waves and noise are presented. These disturbances mostly affect the algorithm when it is following land, because that is when accurate data is needed. In figure 3.6, two runs are shown: the one to the left has no noise countermeasures active, while the one to the right has the countermeasures described in section 2.2.7 activated.

Figure 3.6: Comparison between noise countermeasures disabled and enabled

As seen in the figure it is possible to make the boat follow land as intended even with a lot of noise. But it is not without a significant drawback: due to a slower controller, the risk of hitting an obstacle increases significantly because the boat might not have time to avoid it.
3.4 Complete solution results

The results shown in this section are obtained by using all features described in Chapter 2.

3.4.1 Step by step

In order to illustrate the complete process of a depth survey, as described in section 2.3.1, a series of figures (3.7 to 3.12) is shown in this section. The subregion currently being scanned has green borders, while the rest have red borders. Only a small amount of noise is introduced.

Figure 3.7: At the beginning of the algorithm, the initial polygon has been split into triangles and scanning is about to start

Figure 3.8: Scanning of the first triangle is completed and scanning of the next one is in progress
Figure 3.9: Scanning in another polygon, a few of the islands have been detected.

Figure 3.10: About half the area has been surveyed. Due to the triangulation, the boat is able to make its way to the other side of the group of islands.

Figure 3.11: Most of the area is covered, several continuous regions are still unknown, but identified.
3.4.2 Robustness and consistency

Here we present results on the noise impact on performance.

Figure 3.12: Scanning completed. There are no more continuous regions to identify.

Figure 3.13: Baseline test without noise.
Figure 3.14: Test with wind, waves and sensor inaccuracies. Despite the rough conditions, the scan is completed.

Figure 3.15: Noise impact on performance. It is clear that the simulated noise has a large influence on both distance travelled and time elapsed.

The consistency of the algorithm is shown by attempting to scan the same area 20 times. The map of the area is shown in figure 3.16. In figure 3.17, the coverage data is shown.
Figure 3.16: Map of the area being tested

Figure 3.17: Collection of coverage data
Chapter 4

Discussion and Conclusions

In this section, we discuss the results and put them in a larger context. We suggest further work and discuss ethical aspects of automation.

4.1 Simulation realism

The goal of this project is to provide a good basis for a navigation algorithm that is applicable in a real application. In order to do so, we aimed to create a realistic test environment. However, within the scope and time limitations of this project, certain simplifications had to be made. A major such simplification is the absence of physics: for instance, collisions are not being simulated. If the boat were to hit land in the simulation, no action is taken. Further, the effect of waves is simply being modelled as deviations in sonar data. In a real application, the waves will cause the boat to pitch and roll, making the sensors point at changing directions. Thus, a real navigation algorithm will need to compensate for these phenomena. In contrast, a lot of effort has been put into making the map data resemble a real seafloor. In order to make sure that data can be obtained at every coordinate, an interpolation between data points has been performed.

4.2 The choice of coverage approach

Both the sweeping and spiral patterns have their advantages and disadvantages. They are fully capable to complete a survey of fairly simple environments, but the shortcomings of the spiral pattern become apparent as the complexity increases. The spiral pattern proved to have numerous disadvantages that resulted in it being unsuitable for the task of complete surveys of unknown environments. Compared to the sweeping pattern, it follows land for long periods of time. This part of the navigation algorithm is most susceptible to noise, as shown in figure 3.6. Noise in the real world application can potentially be much worse than in the simulation, making it virtually impossible to run the contour following for an extended period of time. Combined with the fact that the same coastline is often revisited numerous times and the shape of the resulting unscanned regions, we conclude that it is unfit for this application. However, it should not be dismissed completely, since search missions could benefit from the basic shape of a spiral. Examples of such missions are search and rescue or looking for a shipwreck on the seafloor.
4.3 Evaluation of the complete solution

The task at hand, as stated in Chapter 1, was "Design a robust navigation algorithm that is suitable for depth surveys in unknown naval environments and evaluate its performance through simulations." From the results in Chapter 3, we can conclude that the complete solution, consisting of decomposition, coverage and navigation algorithms, is capable of achieving consistent and reliable coverage of unknown and complex environments.

As discussed throughout the report, noise has a major impact on performance. The results shown in section 3.2 suggest that compensating for wind might not be as difficult as one might think. Our rather simple solution proved to be very effective at keeping the boat on the intended course, allowing better coverage. Sensor inaccuracies, on the other hand, turned out to be a much bigger problem. Not only do they corrupt measurement data, but they also make it harder for the navigation to perform its task. Our countermeasures proved to have some benefit, as shown in section 3.3, but not without a costly drawback. In order to make the countermeasures work effectively, a lot of fine tuning consisting of a slower controller is required. This makes the boat slower to react, which in turn makes the boat less susceptible to noise, but it also makes it less capable to avoid obstacles when following land.

4.4 On the ethics of automation

Since the industrial revolution, an increasing number of processes have been automated. This has led to an endless debate as to how machines affect society. The automation has made a lot of professions obsolete and continues to do so. In the modern digital world, this phenomenon has taken yet another form. As technology advances, more and more control is turned over to automation. Algorithms control an increasing number of functions, ranging from stock trading to automated news generation. This makes society increasingly dependent on automation and this leads us to the question of how far we can take this and what happens if everything should go awry.

While automating a depth survey might not have any profound socioeconomical effects, it is still a small part in this development. The question of public safety needs to be raised, as the proposed craft has no way of identifying another boat. Another aspect is the decreased demand for unskilled jobs and increased demand for highly skilled jobs.

4.5 Further work

Even though the proposed approach solves the problem at hand, there is still a lot that can be done to improve it. Despite the triangulation and remaining area identification, some redundant coverage is still being performed. This poses a problem for crafts with low operating range. Another contributing factor to inefficient, and therefore expensive, fuel use is the way a new polygon is chosen for scanning. In the current solution, the closest one is chosen, regardless of how many others exist or where they are located. A "Travelling Salesman Problem" approach to this decision making could be implemented in order to minimize fuel costs and time consumption. A collaboration between several boats working on the same area could be another way of achieving this.
References


