NTNU

Norwegian University of
Science and Technology

# Development of an Obstacle Detection and Avoidance System for ROV

Ørjan Grefstad

Marine Technology
Submission date: June 2018
Supervisor: Ingrid Schjølberg, IMT
Co-supervisor: Ole Alexander Eidsvik, IMT
Vegard Wie Henriksen, IKM Technology

Norwegian University of Science and Technology
Department of Marine Technology

# Project Description

Subsea inspection, maintanance and repair (IMR) missions require remotely operated vehicles (ROVs). These tasks are traditionally performed manually, but is well suited for automation. The vehicles considered in this project are therefore fully actuated ROVs and/or AUVs. Survey AUVs with torpedo shape and no sway actuation are thus not considered.

This project is part of addressing the main challenge of increasing the level of autonomy and robustness for automatic mapping, monitoring and intervention, high-level planning/re-planning and reconfiguration of single and multiple vehicles subject to the particular mission, environmental condition, available energy, communication constraints, and any failure conditions.

This project will consider a Merlin class ROVs, which is a Class 3 work class-ROV with fully electric propulsion. The vehicle is equipped with a front-facing, single beam sonar that should be used to implement an obstacle detection and avoidance system. The developed system should be able to give commands to the vehicles autopilot system and provide a level of safety during automatic operations. IKM Technology facilitates a high-fidelity simulator to aid the development and verification of the proposed motion planning algorithms prior to full-scale tests.

# Scope

The following subtasks should be addressed:

1. Develop an algorithm for obstacle detection using measurements from a sonar and available maps. The sonar is single beam, front facing with 180 degrees viewing angle. Secondary input is two-dimensional map data.

2. Develop a motion-planning algorithm for obstacle avoidance using the developed obstacle detection algorithm.

3. Use the algorithms and design a guidance system that can give commands to an autopilot system. Assume the autopilot is capable of dynamic positioning and path following of curved paths by providing it waypoints.

4. Implement and verify the guidance system in Matlab/Python.

5. Verify the developed system using the high-fidelity Merlin simulator at Bryne.

6. Verify the developed system in full-scale experiments by implementing the solution into the existing guidance and control system.

7. Conclude your results.

Supervisor: Professor Ingrid Schjølberg
Co-supervisor: Vegard Wie Henriksen, IKM Technology AS

# Abstract

Unmanned underwater vehicles, such as remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) are commonly used for inspection, maintenance and repair (IMR) missions in the oil and gas industry. This is a cost-driven industry and advances in autonomy is a key factor to reduce the mission expenses.

Collision avoidance is one of the main challenges in the field of AUVs. In this thesis a system for detecting obstacles and planning a new path around the obstacles is proposed. The system is divided into three modules: an object detection module, a collision avoidance module and a guidance module.

The object detection module uses a single-beam mechanically scanning sonar to populate a probabilistic occupancy grid. The sonar data is related to the probability of occupancy through a dynamic inverse-sonar model. The occupancy grid is vehicle-fixed and thus position or velocity data is needed for translating and turning the grid. The translation and rotation is archived using an image processing technique known as affine transformation.

A global occupancy grid has to account for the growth of the positional uncertainty, and thus the accuracy of the map will fall over time. With a local map the obstacles will be correctly located with respect to the vehicle, which is sufficient for the purpose of collision avoidance. The obstacles are detected through a contour detection algorithm. The detected contours are then expanded to make a safety margin around the obstacles.

The collision avoidance module compares the current path with the detected obstacles and initiates a path recalculation if they coincide. The path recalculation is done with a combination of Voronoi diagrams and a modified version of Dijkstra's shortest path algorithm. Once a new path is calculated, it is smoothed using Fermat's spiral and sent to the guidance system.

The guidance system is a classic line of sight guidance scheme, with a velocity dependent lookahead distance. The velocity guidance uses the path curvature as an input parameter. This enables the guidance system to reduce the commanded velocity when sharp turns are detected.

Several simulations were performed and the complete system was tested on a ROV stationed on the Snorre B oil field. During the field experiments, it was confirmed that a mechanically scanning single-beam sonar is sufficient as the only sensor for detecting and avoiding obstacles in an underwater environment. The technology is also applicable to AUVs as the calculated paths have a continuous curvature.

Regarding further developments of this system, it is suggested to look into other guidance schemes, such as trajectory tracking. The object detection system would benefit from introducing additional sensors, such as cameras for detection of close obstacles.

# Sammendrag

Ubemannede undervannsfarkoster, som fjernstyrte kjøretøyer (ROVer) og autonome undervannsfarkoster (AUV), brukes ofte til inspeksjon, vedlikehold og reparasjon (IMR) operasjoner i oljesektoren. Dette er en kostnadsdrevet industri, og fremskritt i autonomi er en sentral faktor for å redusere kostander forbundet med IMR operasjoner.

Kollisjonsunngåelse er en av hovedutfordringene innen AUV-feltet. I denne oppgaven er det foreslått et system for å oppdage hindringer og planlegge en ny bane rundt hindringene. Systemet er delt inn i tre moduler: en modul for å oppdage objekter, en for å oppdage kollisjoner og planlegge en ny bane og en navigasjonsmodul.

Modulen for objektoppdagelse bruker en sonar med en mekanisk roterende stråle. Disse dataene blir så brukt for å beregene sannsynligheten for at celler i et rutenett har en hindring i seg. Sonardataene er relatert til sannsynligheten i rutenettet gjennom en dynamisk omvendt sonarmodell. Rutenettet er festet til kjøretøyet, og dermed er det behov for posisjon- eller hastighetsdata for å flytte og rotere rutenettet. Dette gjøres ved hjelp av en bildebehandlingsteknikk kalt affin transformasjon.

Et globalt rutenett er nødt til å ta hensyn til posisjonsusikkerheten, og dermed vil nøyaktigheten falle over tid. Med et lokalt rutenett er hindringene være riktig plassert i forhold til kjøretøyet, noe som er tilstrekkelig for å en unngå en kollisjon. Objektene oppdages gjennom en konturdeteksjonsalgoritme. Konturene blir deretter ekspandert for å lage en sikkerhetsmargin rundt dem.

Modulen for kollisjonsunngåelse sammenligner den nåværende banen med de kjente hindringene og setter i gang beregningen av en ny bane, hvis det er fare for kollision. Denne beregningen er gjennomført med en kombinasjon av Voronoi-diagrammer og en modifisert versjon av Dijkstras algoritme for å finne korteste rute. Når en ny bane er beregnet, blir den glattet med Fermats spiral og sendt til navigasjonssystemet.

Navigasjonssystemet er et klassisk siktelinjesystem (LOS) med en hastighetsavhengig synsavstand. Hastighetsveiledningen bruker banekurvaturen som en inngangsparameter. Dette gjør det mulig for Navigasjonssystemet å redusere hastigheten når skarpe svinger oppdages.

Flere simuleringer ble kjørt og hele systemet ble testet på en ROV stasjonert på Snorre B platformen. Ved hjelp av disse testene ble det bekreftet at en sonar med en mekanisk roterende stråle er tilstrekkelig som den eneste sensoren for å oppdage og unngå hindringer i et undervannsområde. Teknologien virker også for AUVer fordi de beregnede banene har en kontinuerlig krumning.

For videre forskning og utvikling av dette systemet anbefales det å undersøke andre navigasjonssystemer, slik som banesporing. Det vil være hensiktsmessig å inkludere flere sensorer i modulen for objektdeteksjon, for eksempel et kamera slik at det blir lettere å oppdage hindringer på nært hold.

# Preface

This thesis is the product of my work during the spring semester of 2018 at the Department of Marine Technology, Norwegian University of Science and Technology (NTNU). The work is based on the results of the project thesis from the fall semester of 2017. The work has been carried out under the supervision and guidance of Professor Ingrid Schjølberg and in cooperation with IKM Technology.

I would like to thank Professor Ingrid Schjølberg for interesting discussions and valuable help on the thesis. I would also like to express gratitude to Vegard Wie Henriksen, Øyvind Løberg Aakre and Vidar Eriksen at IKM Technology for input and help during the field experiment and implementation of the system. In addition, I would like to thank IKM Subsea for the ability to perform field experiments and their ROV pilots for their help in flying the ROV and interpreting sonar images.

Ørjan Grefstad
Trondheim, June 25, 2018

# Table of Contents

# List of Tables

# List of Figures

# Abbreviations

| | | |
|---|---|---|
| UCV | = | Ultra Compact Vehicle |
| ROV | = | Remotely Operated Vehicle |
| AUV | = | Autonomous Underwater Vehicle |
| UID | = | Underwater Intervention Drone |
| DOF | = | Degree of Freedom |
| IMU | = | Inertial Measurement Unit |
| DST | = | Digital Sonar Technology |
| UDP | = | User Datagram Protocol |
| IP | = | Internet Protocol |
| RGB | = | Red-Green-Blue |
| MORSE | = | Modular Open Robots Simulation Engine |
| MOOS | = | Mission Oriented Operating Suite |
| GUI | = | Graphical User Interface |
| GPU | = | Graphical Processing Unit |
| IMR | = | Inspection, Maintenance and Repair |
| SNR | = | Signal to Noise Ratio |
| CHIRP | = | Compressed High Intensity Radar Pulse |
| EM | = | Electromagnetic |
| LBL | = | Long Base Line |
| SBL | = | Short Base Line |
| USBL | = | Ultra Short Base Line |
| LOS | = | Line-of-sight |
| INS | = | Inertial Navigation System |
| WGS84 | = | World Geodetic System - 1984 |
| WP | = | Waypoint |
| VD | = | Voronoi Diagram |
| ROA | = | Region of Acceptance |

Chapter 1

# Introduction

## 1.1 Background and Motivation

Unmanned underwater vehicles, such as remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) are commonly used for inspection, maintenance and repair (IMR) missions in the oil and gas industry. This is a cost-driven industry and advances in automation is a key factor to reduce the mission expenses.

ROVs are underwater robots connected to the surface or an underwater hub through an umbilical transferring power and information. Due to the supply of power from the surface, ROVs can normally carry large payloads and tools, such as robotic arms, multiple cameras and powerful lights. ROVs are normally fully actuated, meaning that it can be controlled in all degrees of freedom (DOFs).

AUVs are not directly connected to the surface, and thus the power supply is limited. Communication with the surface is possible through acoustic transmissions, but the data rate is limited. In contrast to ROVs, AUVs are normally underactuated, which limits the hover capabilities.

The industry-driven innovation is currently moving towards resident underwater intervention drones (UIDs) [16], where the goal is to have vehicles with the intervention potential of an ROV with the freedom of an UAV. This is a way of greatly reducing costs, as the need for costly surface intervention vessels is removed. A resident and tetherless UID can also service several subsea production areas as it can move freely between docking stations. The need for such innovation is expressed in Equinor's (former Statoil) development schedule for the next 4-7 year period [16], where they plan to install a pilot for autonomous UIDs that can fly between docking stations. This goal introduces the need for innovation in several areas such as power, communication and autonomy systems.

The National Research Council [6] defines an autonomous vehicle as an *"unmanned vehicle with some level of autonomy built in"*, which includes vehicles such as UAVs and UIDs, but also less autonomous vehicles such as ROVs. The National Research Council [6] also defines four levels of autonomy, where the highest level of autonomy describes a system where all mission-related functions are executed automatically. Several other

descriptions of an autonomous system exist in the literature, but the main point is that the system should be able to sense, plan and act on its own. The ability to sense, plan and act introduces the possibility of performing complex tasks without human intervention.

Collision avoidance is one of the main challenges in the field of autonomous underwater operations. This challenge is often solved using multibeam sonars, in a simultaneous localization and mapping (SLAM) approach as done by Palomer et al. [30] or with image recognition based techniques such as the method proposed by Braginsky and Guterman [2]. To accommodate these methods a multibeam sonar is needed. Multibeam sonars have superior performance for obstacle detection as they scan a large sector in one scan. The superior performance does however come at a price, the cost is significantly higher, the physical size is larger and the power consumption is higher. These factors is a concern when UIDs are shifted towards battery power, which requires smaller vehicles to reduce power usage. However; a single beam sonar does not scan the entire sector in front of the vehicle at once, which complicates the object detection process. This has been done with occupancy grids by authors such as Ganesan et al. [13] and with a potential field method by authors such as Solari et al. [32].

## 1.2 Scope

The objective of this thesis can be formulated on the basis of the challenges presented in the previous section. In order to perform safe autonomous ROV operations, the following tasks should be performed.

1. Perform a literature study on underwater object detection, sonar theory and collision avoidance.

2. Propose and develop a set of algorithms to perform collision detection and avoidance.

3. Verify the developed system in simulations using the MORSE simulator.

4. Implement and verify a guidance system, using the high-fidelity Merlin simulator at IKM's headquarters at Bryne.

5. Perform preliminary field tests of the collision avoidance system using the resident Merlin UCV at the Snorre B oil field.

## 1.3 Contributions and Thesis Outline

The contribution of the thesis is a collision avoidance system combining the theory of occupancy grids for single-beam sonar object detection and Voronoi diagrams for the planning of a new path. The work in this thesis resulted in a contribution to *2018 IEEE OES Autonomous Underwater Vehicle Symposium*, and the preliminary abstract is presented in the appendix. The outline of the thesis is as follows:

In **Chapter 2** the background theory necessary for the developed algorithms are presented together with a short literature study regarding occupancy grids, Voronoi Diagrams and Fermat's spiral is presented. The chapter also includes a presentation of the software frameworks used in the thesis.

In **Chapter 3** a study of underwater acoustics, with a special focus on relevant theory for single-beam sonars are presented.

In **Chapter 4** the developed system is presented. The chapter describes the object detection module, the collision avoidance module and the guidance module. The main contributions of the thesis are presented in this chapter.

In **Chapter 5** the experimental setup for the MORSE simulations, the Merlin simulator simulations and the field trials are presented.

In **Chapter 6** the results from the simulations are presented.

In **Chapter 7** the results from the field trials at the Snorre B oil field are presented.

In **Chapter 8** the results from Ch. 7 and Ch. 8 are discussed.

In **Chapter 9** a conclusion is formulated and suggestions for further work in the subject are presented.

In **Appendix A** the preliminary abstract of a contribution to the *2018 IEEE OES Autonomous Underwater Vehicle Symposium* is presented.

In **Appendix B** additional results from the field trials are presented.

**Appendix C** lists the electronic attachments delivered in DAIM.

Chapter 2

# Background

In this chapter previous work in the field of underwater collision avoidance will be presented. Then the necessary background theory for the developed collision avoidance system will be presented. The last section is a short introduction of the software used in the thesis. In this chapter previous work in the field of underwater collision avoidance and the necessary background theory for the developed collision avoidance system will be presented. Finally, section 2.9 presents a short introduction of the software used in the thesis.

## 2.1 Spatial Mapping Using Occupancy grids

An occupancy grid is a powerful tool for a robot moving in an unknown environment. It can be used as a basis for obstacle detection, collision avoidance and simultaneous localization and mapping(SLAM). The first occupancy grids were developed in the early 90's by Elfes [8]. Since then it has become the dominant solution for environment modeling for mobile robots. An occupancy grid is a spatial representation of the robot's surroundings. It is most common to divide the grid into a set of equally spaced quadratic cells, but it is also possible to use a polar occupancy grid. The occupancy grid map can be given either as a set of probabilities of occupancy or as a binary set of occupied or empty cells.

### 2.1.1 Occupancy Grids

In this section, the mathematics behind the occupancy grid is described. Most of the information in this section is based on Elfes [8] and Thrun [37].

An occupancy grid is a mapping of the environment into grid cells. Each grid cell contains an estimate of the probability of occupancy. In addition, each grid cell can also contain a binary occupied or not occupied value. To make the estimation problem less complex the high-dimensional problem is decomposed into a set of binary problems, one for each cell.

The state of one cell with index $(x, y)$ can either be occupied, $m_{x,y}$ or not occupied, $\widehat{m_{x,y}}$. The occupancy grid mapping problem can then be formulated as the probability of each cell being occupied given the measurements, as stated in Eq. (2.1) [37].

$$p(m|Z_t) = p(m|\{z_t, z_{t-1}, \ldots, z_0\}) \tag{2.1}$$

where $z_t$ is the range measurement at time instant $t$, and $Z_t$ is all the range measurements up to time $t$. These measurements also might include the pose of the vehicle as well as the sonar heading, beam-width and other relevant parameters. These measurements have information about all the grid cells overlapping the sonar beam up to the measurement distance. The formulation in Eq. (2.1) is hard to compute, due to the interdependence of the cells and can be simplified by making two basic assumptions [37]:

- *Static world assumption*: The past sensor readings are conditionally independent, given knowledge of the map.
- *Cell independence assumption*: Different grid cells are conditionally independent, given knowledge of the individual cell.

Using these assumptions together with Bayes' theorem the recursive equation for updating grid cells can be written on a log-odds form [37]

$$l_{x,y}^t = \log \frac{p(m_{x,y}|z_t)}{1 - p(m_{x,y}|z_t)} + \log \frac{p(m_{x,y})}{1 - p(m_{x,y})} + l_{x,y}^{t-1} \tag{2.2}$$

where $l_{x,y}^t$ is the log-odds of a grid cell at time $t$. This equation consists of two parts, the prior $p(m_{x,y})$ which depends on the expected obstacle density in the environment and $p(m_{x,y}|z_t)$ which is the probability that there is an obstacle given the current sensor measurement. This probability is called the inverse sensor model.

### 2.1.2 Inverse sensor model

The probability $p(m_{x,y}|z_t)$ is the probability of the grid cell being occupied given the sensor measurement $z_t$. In other words, this is a mapping from the measurement to the cause. Elfes [8] used a probability density function depending on the range and angle, where both parameters were modeled as Gaussian uncertainties. This model was later extended by Konolige [21] to include multiple targets and specular reflections.

The inverse sensor model was extended by Zhou et al. [44], which uses a dynamic inverse-sensor model based on the sonar equation where the probabilities are adjusted for the incident angle. The incident angle is calculated by using linear regression on all cells above a certain threshold to find the face of the obstacle, and subsequently computing the angle between the sonar beam and the face of the obstacle.

### 2.1.3 Forward sensor model

The mapping from measurement to cause in the inverse sensor model is the opposite of how the measurement process is done. Thrun [37] introduced a method for Occupancy grid

mapping with a forward sensor model. The forward sensor model computes the probability of getting the measurement given the obstacle, $p(Z_t|m)$.

This method creates more accurate maps, but due to computationally expensive optimization algorithms, the method is not capable of running in real time. The greatest difference between the forward and inverse sensor model is that the forward model does not have to assume independence of cells.

Shvets et al. [31] extended the method by using a gradient descent optimization method, and claim that it is capable of running in real time, but no such experiments have yet been carried out.

## 2.2 Image Processing

Several different image processing techniques were used throughout the thesis. This section describes these techniques and how a digital image is represented.

### 2.2.1 Image representation

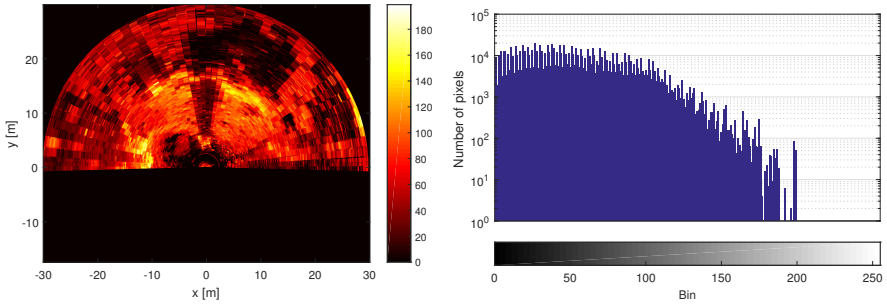
In a computer, an image is represented as a series of bytes. Some common image representations are:

- Grey-scale image where each pixel is represented by an unsigned byte, with an integer value between 0 and 255.
- Red-green-blue (RGB) image where each pixel are represented by 3 bytes, one for each color. This is the most common for color-images.
- Blue-green-red (BGR) image. This uses the same structure as RGB images, but with the colors switched. This is the format used by OpenCV.

With a higher level interface, such as in Python, the images are represented as a matrix $\boldsymbol{M}_{h\times w\times c}$, where $h$ is the height of the image, $w$ is the width of the image and $c$ is the number of colors. With this matrix structure, the coordinate system of an image has the y-axis flipped as opposed to the normal Cartesian coordinate system.

### 2.2.2 Histogram

Histograms are used for images as a way for quantifying the distribution of pixel intensities. An example of a gray-scale histogram for a plot of raw sonar data can be observed in Fig. 2.1b. Histograms can be used for applications such as identifying the correct threshold value. In this thesis, histograms were used to identify a good threshold value for the sonar signals.




**(a)** Plot of raw sonar data.

**(b)** Histogram of the image in Fig. 2.1a.

**Figure 2.1:** Example of the histogram from raw data plot.

### 2.2.3 Thresholding

The most simple form of thresholding is to modify the pixels as described in Eq. (2.3).

$$I_T(i,j) = \begin{cases} k, & \text{for } I(i,j) \geq T \\ 0, & \text{for } I(i,j) < T \end{cases} \tag{2.3}$$

where $I_T$ is the intensity values of the thresholded image, $I$ is the original intensity values, $T$ is the threshold value and $k$ is a constant, often 1, 255 or *True*. More sophisticated thresholding methods include histogram-shape methods, where the shape of the histogram is used to find a thresholding value and clustering-based methods. A common example of clustering based methods are Otsu's method, where the optimum threshold is selected such that the variance within the two groups is minimal [29].

### 2.2.4 Dilation

Dilation is a mathematical morphology operation defined in Eq. (2.4) [14, p. 19]. The dilation operation can be used for inflation of obstacles to create a safety margin or other purposes where inflation of an image feature is necessary. The dilation operation can be explained as looking at each pixel $a_{i,j} \in \boldsymbol{A} \neq 0$ and superimposing a structuring element $S$, as a binary OR-operation, centered at $(i,j)$ onto the final image.

$$\boldsymbol{A} \oplus \boldsymbol{S} \equiv \bigcup_{s \in \boldsymbol{S}} \boldsymbol{A}_s \tag{2.4}$$

where $\oplus$ is the dilation operator, $\boldsymbol{A}$ is the image, $\boldsymbol{S}$ is the structuring element and $\boldsymbol{A}_s$ is the translation of $\boldsymbol{A}$ by $s$. An example of dilation of a binary image can be observed in Fig 2.2, where the original image is to the left and the right image is dilated with a structuring element $\boldsymbol{S} = \mathbf{1}_{20\times 20}$.

**(a)** Original image. $200 \times 200$ pixels

**(b)** Dilated image. $\boldsymbol{S} = \mathbf{1}_{20\times 20}$

**Figure 2.2:** Example of image dilation.

### 2.2.5 Contour detection

A contour is a continuous curve around the boundary of an object, where the object is normally defined by the image intensity. A contour detection algorithm can efficiently transform the pixel information to geometrical shapes, which are useful for a collision avoidance algorithm. Computer vision tools, such as OpenCV usually use a border following-algorithm developed by Suzuki and Be [34] to detect the contours. This algorithm works best on binary images, and thus thresholding has to be performed first. The result of running the algorithm on an image will be the set of every border point of each contour. This can consume a lot of memory and thus it is common to approximate the contours with fewer points. A simple way to do this is to remove all points on a straight line, except the endpoints. A more accurate representation can be archived by using Teh and Chin [35]'s algorithm for detecting dominant points on a closed curve. To further reduce the number of points the contours can be approximated as different shapes, such as rectangles, ellipses or a convex shape. In Fig. 2.3 an example of the different contour approximation methods can be observed, and in Tab. 2.1 an overview of the number of points is shown.

| Approximation method | Number of points |
|---|---|
| No approximation | 1203 |
| Simple approximation | 156 |
| Teh-Chin approximation | 82 |
| Convex approximation | 14 |

**Table 2.1:** Number of approximation points for different contour approximations for the shape shown in Fig. 2.3.

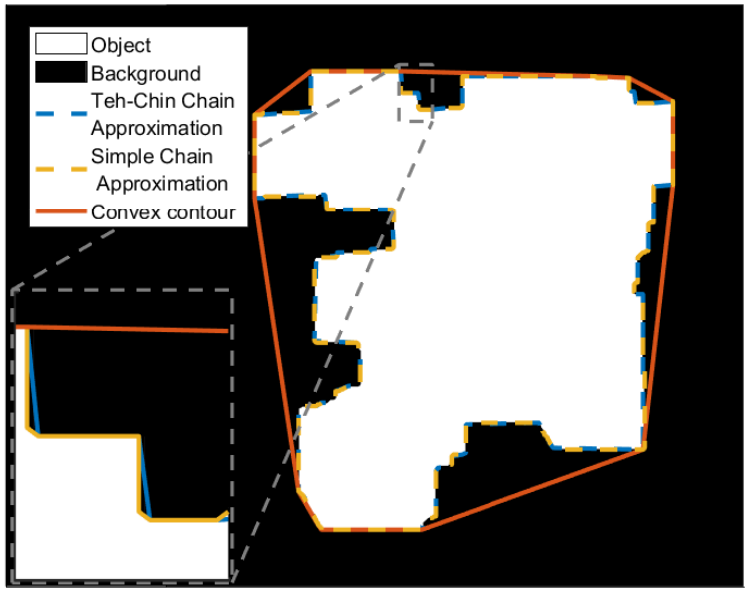



**Figure 2.3:** Example of different contour approximations.

### 2.2.6 Affine Transformation

An affine transformation is a map between two affine spaces, where collinearities are preserved [40]. Affine transformations are commonly used for many different purposes in image manipulation. Some examples are scaling, reflection, rotation and translation of images. The transformation is done by multiplying the transformation matrix, $M$ with the position of each pixel in the image, to get a new position as shown in Eq. (2.5) [40]. When every pixel has a new position, an interpolation algorithm has to be applied as the new positions are not necessarily in a position that fit the integer requirements of an image. A common interpolation method for images is a bicubic interpolation, but other methods, such as a nearest-neighbour interpolation or a linear interpolation can also be used.

$$\boldsymbol{T} = \boldsymbol{MP} = \boldsymbol{M}\left[x, y, 1\right]^T \tag{2.5}$$

where $\boldsymbol{T} = \left[x', y'\right]^T$ is the vector of transformed xy-coordinates. Some common transformation matrices are shown in Eq. (2.6).

$$\boldsymbol{M}_{\text{rotation}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \end{bmatrix}, \qquad \boldsymbol{P} = \left[x - x_0, y - y_0, 1\right] \tag{2.6}$$

$$\boldsymbol{M}_{\text{translation}} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{M}_{\text{translation}} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \end{bmatrix} \tag{2.8}$$

where $(x_0, y_0)$ is the point to rotate around, $(\Delta x, \Delta y)$ is the translation and $s$ is a scale factor.

## 2.3 Voronoi Diagrams

A Voronoi diagram is a method of dividing a plane, $\boldsymbol{X}$ into regions based on the distance to a set of finite points. These points are called generator points, $\boldsymbol{P} = \{p_1, ..., p_2\}$. A metric function $d(x, p_i)$ associates each point $x \in \boldsymbol{X}$ into a region, $\boldsymbol{R}_i$ such that the metric function for all points in the region is less than or equal to the metric function for all other regions. This can be expressed mathematically as in Eq. (2.9) [23].

$$\boldsymbol{R}_i = \{x \in \boldsymbol{X} | d(x, p_i) \leq d(x, p_j) \qquad \forall i \neq j\} \tag{2.9}$$

A common candidate for the metric function is the Euclidean distance, as expressed in Eq. (2.10).

$$d(x, p_i) = |x - p_i| = \sqrt{\left(x_x - p_{i_x}\right)^2 + \left(x_y - p_{i_y}\right)^2} \tag{2.10}$$

From this, a set of lines, $v_i \in \boldsymbol{V}$ can be chosen as the the borders of the regions $\boldsymbol{R}_i$. Using Eq. (2.10) as the metric function will result in the set $\boldsymbol{V}$ being connected by a set $\boldsymbol{E}$ of straight edges.

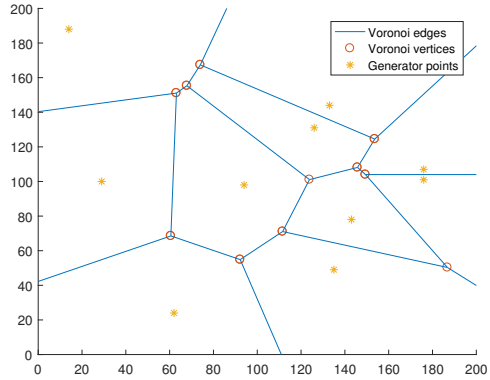



**Figure 2.4:** Example of Voronoi diagram, showing the generator points, the vertices and the edges.

## 2.4 Dijkstra's Algorithm

Dijkstra's algorithm is an algorithm used for finding the shortest path between nodes in a graph. It was first developed by Dijkstra [7], but further developments made by Fredman and Tarjan [12] drastically reduced the runtime of the algorithm from $\boldsymbol{O}(|\boldsymbol{V}|^2)$ to $\boldsymbol{O}(|\boldsymbol{E}| + |\boldsymbol{V}| \log |\boldsymbol{V}|)$, where $\boldsymbol{V}$ is the number of nodes and $\boldsymbol{E}$ is the number of edges. The algorithm visits all nodes in the graph uniformly and finds the shortest paths between them. A more common variation of the algorithm finds the shortest path between a source and sink node.

Yen [43] made a variation of the algorithm that can be used to find the k-shortest path, meaning the second shortest path, the third shortest path and the k-shortest path. This algorithm is the basis for a modification made by Lekkas [23]. This algorithm uses the optimal path from the previous run, but once a node is invalid due to clearance constraints all connections to this node are removed. A new path is then constructed from the previous node to the goal node. The final path will then be constructed of the initial path up to, but not containing the invalid node and the new path to the goal node.

## 2.5 Fermat's Spiral

A path consisting of straight line segments is possible to follow with a fully actuated ROV, but due to the curvature discontinuous nature of such a curve, a full stop would be required at each waypoint. A better solution would be to modify the path such that the curvature is continuous and as small as possible. A solution to this is presented by Lekkas [23], where Fermat's spiral is used to make a continuous curvature path. Fermat's spiral is a geometrical curve which can be described in Cartesian coordinates as done by Lekkas [23]:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_0 + a\sqrt{\theta}\cos(\rho\theta + \chi_0) \\ y_0 + a\sqrt{\theta}\sin(\rho\theta + \chi_0) \end{bmatrix} \tag{2.11}$$

where $a$ is a scaling factor, $\rho = 1, -1$ is the turning direction and $\chi_0$ is the initial tangent angle. The formulation of Fermat's spiral in Eq. (2.11) can then be used to join two straight line path segments in a continuous curvature path consisting of two mirrored curves [23]. A continuous path is not suitable for a waypoint based guidance system, and thus a discretization should be performed.

## 2.6 Coordinate Systems

In this thesis, several different reference frames and coordinate systems are in use. A description of these reference frames is presented in this section.

### 2.6.1 World Geodetic System - 1984

The World Geodetic System (WGS84) is a geodetic reference system centered at the Earth's center of mass [27]. This is the reference system used by Global Positioning

System (GPS), and the position is provided as a latitude and longitude.

### 2.6.2 Universal Transverse Mercator

The Universal Transverse Mercator (UTM) projection is a 2-dimensional coordinate system. The projection is divided into 60 different zones [20]. A position consists of northing, easting and a grid zone. It is also possible to specify a vertical position. In this thesis all coordinates given in UTM-coordinates are in zone 32 North, which covers most of the southern part of Norway, and the offshore-areas. The UTM system can be used as a North-East-Down (NED) coordinate system, denoted $n$.

### 2.6.3 Body-fixed reference frame

A body-fixed reference frame is a moving coordinate frame that is fixed to the body. In this thesis two different body-fixed reference frames are used. The vehicle reference frame is denoted with $veh$ subscript and has units meters. The axis are defined, by Fossen [9] as:

- $x_{veh}$ - Longitudinal axis, from aft to fore
- $y_{veh}$ - Transverse axis from port to starboard
- $z_{veh}$ - Normal axis from top to bottom

The grid reference frame is a two-dimensional body-fixed reference frame used for the occupancy grid and is denoted with a $g$ subscript. This reference frame has no units. The origin of $g$ is located at the upper left corner of the grid, which has a position $(x_{veh}, y_{veh}) = (r_{scale}, -r_{scale})$ with $r_{scale}$ being the sonar range. The horizontal x-axis is from left to right and the vertical y-axis is from top to bottom. An example of this coordinate system can be seen in Fig. 4.6.

## 2.7 Translation between reference frames

The transformation of velocities between the vehicle-fixed reference frame and the NED-frame is done by using the rotation matrix $\boldsymbol{R}^n_{veh}(\boldsymbol{\Theta_{nb}})$, as described by Fossen [9, p. 22]. Assuming stability and small angles in pitch and roll, the rotation matrix can be reduced to $\boldsymbol{R}^n_{veh}(\psi)$, which is shown in Eq. (2.12).

$$\boldsymbol{R}^n_{veh}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

The transformations between $veh$ and $n$, and back are then

$$\boldsymbol{v}_{veh} = \boldsymbol{R}^n_{veh}(\psi)\boldsymbol{v}^n \tag{2.13}$$

$$\boldsymbol{v}_n = \left(\boldsymbol{R}^n_{veh}(\psi)\right)^T \boldsymbol{v}^{veh} \tag{2.14}$$

The conversion between the latitude and longitude in the WGS84 reference frame to NED in UTM coordinates can be done by tools such as the Helmert 7 point transformation, described in [27]. The transformation uses a set of 7 parameters to do the transformation.

## 2.8 Guidance and Control

The guidance system of a vehicle is a system that translates high-level objectives to set-points for the control system. A typical guidance system will decide the heading needed to get the vehicle to converge on a path, or keep the vehicle in a stable position.

### 2.8.1 Line-Of-Sight Guidance

Line-of-Sight (LOS) guidance is a 3-point guidance scheme, which is commonly used for path following of straight-line paths. The idea behind this guidance scheme is to force the vehicle to track the path. This is done by constructing a vector from the vehicle, to either the next waypoint on the path or to a point on the line between two waypoints. The desired heading angle can then be calculated from the LOS-vector. The following definitions are based on Fossen [9, Ch. 10.3].

The first step is to calculate the cross-track error, $e(t)$ and the along-track distance, $s(t)$ as shown in Eq. (2.15).

$$\boldsymbol{\varepsilon}(t) = \big[s(t), e(t)\big]^T = \boldsymbol{R}_p(\alpha_k)^T(\boldsymbol{p}^n(t) - \boldsymbol{p}_k^n) \tag{2.15}$$

where $\boldsymbol{R}_p$ is the rotation matrix, defined in Eq. (2.16) from the path-frame to the NED-frame of reference, $\alpha_k$ is the angle between the line segment and the north-axis, defined in Eq. (2.17), $\boldsymbol{p}^n(t)$ is the vehicle position and $\boldsymbol{p}_k^n$ is the LOS-point.

$$\boldsymbol{R}_p(\alpha_k) \equiv \begin{bmatrix} \cos(\alpha_k) & -\sin(\alpha_k) \\ \sin(\alpha_k) & \cos(\alpha_k) \end{bmatrix} \tag{2.16}$$

$$\alpha_k = \arctan\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) \tag{2.17}$$

where the waypoints are defined as $\text{WP}_k = (N_k, E_k) = (x_k, y_k)$. The desired course-angle can then be calculated as

$$\chi_d(e) = \chi_p + \chi_r(e) = \alpha_k + \arctan\left(\frac{-e(t)}{\Delta(t)}\right) \tag{2.18}$$

where $\Delta(t)$ is a look-ahead-distance which could be fixed or time-varying. A switching mechanism is needed, such that a new waypoint can be selected once the vehicle is close enough. This is normally done by the use of a region-of-acceptance (ROA). The ROA can be understood as a circle enclosing the waypoint, and the switch to the next waypoint will happen when the vehicle enters this circle. The ROA is defined in Eq. (2.19).

$$|\boldsymbol{p}^n(t) - \boldsymbol{p}_k^n| = |\boldsymbol{\varepsilon}(t)| \leq \text{ROA}_k \tag{2.19}$$

To achieve the motion control objective a velocity-guidance law is needed as well. This law can be fixed velocity or it can be a time-varying velocity depending on for example the curvature.

### 2.8.2 PID-controller

A PID-controller is a commonly used control-loop consisting of the different terms: proportional (P), integral (I) and derivative (D). These three terms is efficient and easy way of controlling a dynamic system. The P-term handles the error, the I-term takes care of steady-state offset and the D-term prevents rapid change. Mathematically the PID-controller can be formulated as in Eq. (2.20). The error is defined as $\tilde{x}(t) = x(t) - x_d(t)$, where $x$ is the state to be controlled and $x_d(t)$ is the desired state. The control objective is then to force $\tilde{x}(t)$ to zero.

$$\tau = K_p\tilde{x} + K_d\frac{d\tilde{x}}{dt} + \int_0^t \tilde{x}dt \tag{2.20}$$

where $\tau$ is the control-action and $K_p, K_d, K_i$ are tunable gains. All the terms are not necessary for every application. For a heading controller, it is normally sufficient with a PD-controller and the surge-velocity can normally be handled with PI-controller. A controller does not normally handle big steps in the desired state well, and thus a smooth reference signal is often needed.

### 2.8.3 Reference Models

A reference model is a method for creating a smooth trajectory for the controller. This is helpful for avoiding steps, which in the worst case can make a controller unstable. A common way of designing a reference model is a low-pass filter. Another solution is to use a mass-damper-spring system, which is convenient for marine vehicles, according to Fossen [9]. The transfer-function for such a system is given as

$$h(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.21}$$

where $\omega_n$ is the natural frequency and $\zeta$ is the relative damping ratio. $\omega_n$ and $\zeta$ are tunable variables, which should be as close as possible to the real signal.

## 2.9 Software frameworks

In this section, a brief overview of the different software frameworks used in this project is presented. All of the software mentioned in this section is open-source if not stated otherwise.

### 2.9.1 Python

Python is a high-level programming language, well suited for rapid prototyping. It has an extensive standard library, which makes code development effective.

The NumPy library is an open-source library for mathematics. It has good support for matrices and vectors. One of the main advantages of NumPy is that the core functions of the library are written as compiled C/C++ code, which makes it computationally effective.

The SciPy library is an extensive library for mathematics and scientific programming. It has support for making Voronoi diagrams, using Dijkstra's algorithm and a variety of interpolation algorithms.

The pyProj wrapper for PROJ.4 is a library for performing cartographic transformations and geodetic computations [41]. This is a useful library for transforming between latitude and longitude in the WGS84 system to UTM-coordinates.

PyQt is a Python interface to the Qt framework for Graphical User Interface(GUI) programming. This is a useful and powerful tool to easily make a cross-platform GUI for Python code.

PyQtGraph is an extension to PyQt which makes plotting and displaying/images easier and faster.

### 2.9.2 OpenCV

The OpenCv library is a software library for computer vision and machine learning. The library is written in C++, but has interfaces for multiple languages across several platforms [28]. In this thesis the main use is for it's contour and drawing features

### 2.9.3 Software for field test

#### IKM Autopilot Server

The IKM Autopilot Server(APS) software is a proprietary guidance and control software for IKM's ROVs. The software runs on the onshore control rack, and communicates with the ROV over a network connection. The APS can operate in five different modes, which are listed below[1]:

- Dynamic Positioning
- Semi Automatic Mode
- Circular Inspection Mode
- Path following of predefined paths
- Cruise mode for external path following algorithms

In this thesis *Dynamic Positioning* and *Cruise Mode*(CM) are the only modes necessary. The CM accepts desired trajectories in surge, sway, heave and yaw, which is then filtered through a reference model. The APS reads position data from an INS onboard the ROV, and uses this data to calculate the control action. The communication interface to the APS is a binary data protocol, which uses UDP-messages as means of transportation. A screen-shot of the APS is shown in Fig. 2.5

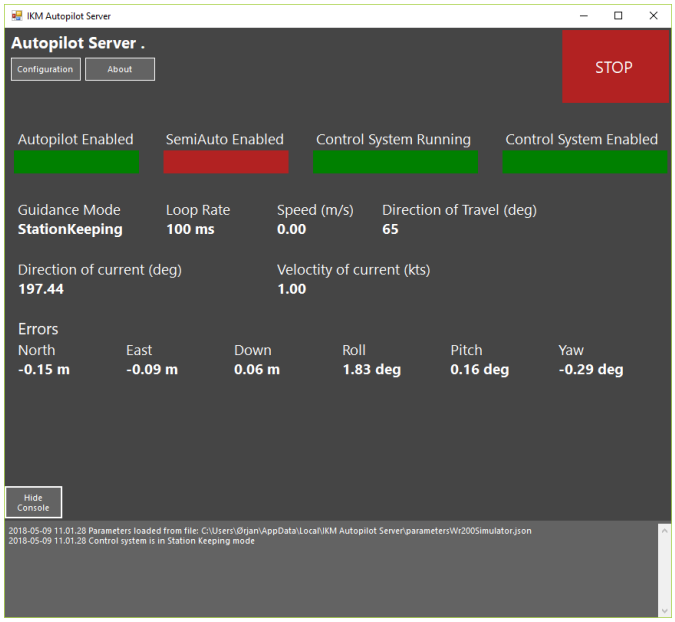



**Figure 2.5:** A screen-shot of the IKM Autopilot Server.

### 2.9.4 Software for Simulation

#### Modular Open Robots Simulation Engine

MORSE (Modular Open Robots Simulation Engine) is an open source python based robot simulator [22]. The simulator integrates *Blender game engine* for 3D rendering and editing and the *Bullet Physics Library* for rigid-body physics simulations. MORSE is written in python which makes it easy to modify and extend the simulator. MORSE also integrates several middle-wares, such as *MOOS* and *ROS*. Through these middle-wares, MORSE can both send and receive messages in an easy manner.

#### Blender

Blender is an open source software suite for 3D modelling and rendering. In MORSE, the Blender game engine is used to render the environment and robots, as well as providing input to several sensors, such as the sonar.

#### Bullet Physics Library

Bullet Physics Library is physics engine used by blender for simulation rigid-body dynamics and collisions.

### Mission Oriented Operating Suite

MOOS (Mission Oriented Operating Suite) is an open source software package, whose main goal is to provide infrastructure for autonomous marine vehicles. In this section MOOSDB and pyMOOS will be the main focus, as these are the only parts of the MOOS package used in this thesis. The MOOS communication network's core is the MOOSDB application [26]. This is a server application that handles all the MOOS messages in the network. In this communications network, all applications can send and receive messages to and from the server, but not to each other. The messages have to follow a strict format, which is handled by the python interface to MOOS, pyMOOS. pyMOOS is capable of sending and receiving messages as well as handling message queues and executing call-back functions when a message is received.

Chapter 3

# Underwater Acoustics

This section presents a description of sonar theory. A short section is also dedicated to other uses of underwater acoustics. Most of this information is based on Hodges [17].

Sonars can be divided into two main groups, namely active and passive sonars. In this section, only active sonars will be discussed, as passive sonars do not have much of an application for collision avoidance. In an active sonar system sound waves are emitted from a transmitter/projector. These sound waves propagate through the medium and are reflected by a target. The reflected sound waves will then propagate back to a hydrophone, which is the receiver. A common name to use for transmitters and hydrophones is a transducer, and in most systems, they are the same unit. In the following text, transducer will be used when they are the same unit, and hydrophone/transmitter will be used when they are separate.

## 3.1 Acoustics

Acoustic waves can move through a medium as longitudinal and transverse waves. Due to the lack of shear strength in water, only transverse waves are possible. The relationship between the velocity and frequency is given in Eq. (3.1).

$$c = \lambda f \tag{3.1}$$

where $\lambda$ is the wavelength, $f$ is the frequency and $c$ is the speed of sound through water. The speed of sound through water depends on several different parameters, where the most important are temperature, salinity and depth.

During the signal's travel from the transducer to the target, and back it is distorted and weakened by different factors, such as sound attenuation, ambient noise, reverberation, which will be discussed below.

### 3.1.1 Sound attenuation

The term sound attenuation includes losses caused by surfaces absorbing sound and transforming it into heat energy, as well as well as losses caused by the interaction with the surface and seafloor. The sound that is absorbed by the ocean itself is mainly caused by two factors: the excitation of ions from magnesium sulfate, boric acid and carbonate and the viscosity of the water [17, p. 92-92]. The most common equation for calculating the sound attenuation is given by Thorp [36] as

$$\alpha = 1.09 \left( 0.1 \frac{f^2}{1 + f^2} + 40 \frac{f^2}{4100 + f^2} \right) \tag{3.2}$$

where $\alpha$ is the intensity absorption coefficient given in dB/km and the frequency is given in kHz. A more thorough calculation, that also includes temperature, salinity, viscosity, depth, pH, and pressure is given by Francois and Garrison [10, 11].

### 3.1.2 Sea surface and seafloor interaction losses

The interaction with the sea surface causes losses that are heavily dependent on the sea state and are deemed outside the scope of the thesis as ROVs on IMR missions mostly operate well below the wave interaction depth.

The sound also interacts with the seafloor, which is typically not a solid object, and thus absorbs much energy. The energy absorption is heavily dependent on the type of the sea floor.

### 3.1.3 Ambient Noise

The dominant source of ambient noise is from thrusters and engines [4, p. 380-381], both from close and far-away surface vessel. According to Christ and Wernli Sr. [4] the ambient noise from surface vessels is approximately 40 dB stronger than other sources. Some other sources are listed below.

- Seismic background noise
- Turbulent pressure fluctuations
- Surface waves
- Second order pressure effects
- Marine life

### 3.1.4 Reverberation

The transmitted sound can return from several different sources than the target. These returning sound waves are called reverberation and can originate from inhomogeneities in the water, air bubbles, marine life, seafloor and sea surface. Reverberation is often the

dominant source of noise for high-frequency sonars [17, p. 143]. The level of reverberation, $Rl$ can according to Hodges [17, p. 153-154] be calculated as

$$Rl = 10 \log \left( \frac{I_0}{r^4} S_s \int Bt\left(\theta, \phi\right) Br\left(\theta, \phi\right) dA \right) \tag{3.3}$$

Where $Bt$ and $Br$ are the beam patterns of the transmitter and receiver, depending on the horizontal angle, $\theta$ and the vertical angle $\phi$. The range of the sonar is $r$, and the transmission power is $I_0$. $S_s$ is the backscattering strength, which is determined by the surface type and incident angle. This strength can vary from $-45dB$ for soft ground to $-25dB$ for solid ground. The strength will also increase with the frequency for smooth surfaces.

### 3.1.5 Target strengths

The target strength refers to the targets ability to return an echo. According to [17, p. 167] the target strength is defined as

$$\text{Nts} = 10 \log \left( \frac{I_r}{I_i} \right) \tag{3.4}$$

where $I_r$ is the reflected sound intensity at a 1 meter distance and $I_i$ is the incident acoustic intensity. The target strength is heavily dependent on the geometry and reflective properties of the target.

### 3.1.6 Doppler Effect

The Doppler effect is the change in frequency when there is a relative velocity between the observer and the source[17, p. 9-10]. If the source and observer are moving away from each other the frequency will be reduced, and if they are moving towards each other the frequency will increase. This relationship is given by Hodges [17, p. 9] as

$$f_r = f_s \frac{c - v_r}{c - v_s} \tag{3.5}$$

where $f_r$ is the observed frequency, $f_s$ is the source frequency, $c$ is the sound velocity and $v_r$ and $v_s$ is the velocity of the observer and source. In the ocean, this can be approximated by

$$\Delta f \approx 3.5 * 10^{-4} f_s \Delta v \tag{3.6}$$

where $\Delta f$ is the change in frequency and $\Delta v$ is the relative velocity in knots.

For a ROV with a typical operational velocity of 1.5 knots and a sonar frequency of 625 kHz the resulting frequency shift is 0.35 kHz, which is negligible. For an AUV with an operational velocity of 5 knots and a sonar frequency of 625 kHz the resulting frequency shift will be over 1 kHz, which should be considered by a digital sonar system.

## 3.2 Sonar equation

The signal to noise ratio(SNR) is a method of estimating the quality of the signal from the transmitter, to the target and back again. The SNR of an active sonar is defined by Horton [18] as

$$\text{SNR} = \text{Lp} - 2\text{Nw} - \text{Nts} + \text{Nag} \tag{3.7}$$

where all units are in *dB*. The variables are defined as

| | | |
|---|---|---|
| *Lp* | = | Strength of source signal |
| *Nw* | = | Losses due to transmission |
| *Nts* | = | Strength of target |
| *Ln* | = | Total noise level |
| *Nag* | = | Gain from antenna array |

The SNR is an important parameter for estimating the performance of a sonar. Using Eq. (3.7) the following equation can be obtained [18].

$$\text{Xs} = \text{Lp} - 2\text{Nw} - \text{Ln} + \text{Nag} - \text{Nrd} \tag{3.8}$$

where *Xs* is the excess signal to noise ratio and *Nrd* is the detection threshold. The excess signal to noise ratio is the amount of SNR above the threshold value. This means that the *Xs* is the signal strength that will be measured by the receiver.

## 3.3 Measurement and discretization

Using the time difference between the transmitted and received signal, the distance, $d$ to a measurement of *Xs* can be calculated as

$$d = \frac{t_1 - t_0}{2} * c \tag{3.9}$$

where $c$ is the sound velocity, $t_0$ is the time of transmittance and $t_1$ is the time of reception.

The sonar pulse is discretized in the receiver. The receiver measures sound intensity, and this level is sampled with a sampling time, $t$ according to

$$\delta t = \frac{2 * R}{V_{sound} * n_{bins}}$$

where $R$ is the scan range, and $n_{bins}$ is the number of sampling bins. Each of these bins contains the sampled sound intensity at range $r_i$, given by

$$r_i = \frac{bin_i}{n_{bins}} R$$

The measurements are then returned as a set of polar bins, where each bin corresponds to a polar grid cell at a position $(r, \theta)$. The length of a grid cell is given as $l_{cell} = R/n_{bins}$ and the width of the cell is equal to the beam width.

### 3.3.1 Statistical Detection Theory

In this section, the classic approach to statistical detection theory will be presented. The received signals can be divided into two categories[17, Ch. 12]:

- $\mathcal{H}_0$ : Null hypothesis, no target present
- $\mathcal{H}_1$ : Alternative hypothesis, target present

These two hypotheses can either be true or false, which makes two different types of errors possible, namely *false alarms* when the null hypothesis is true, but is falsely rejected and *false rests* when the alternative hypothesis is true, but the null hypothesis is selected. The probability of $\mathcal{H}_0$ and $\mathcal{H}_1$ can according to [17, p. 202] be calculated by Baye's theorem as

$$P\left(\mathcal{H}_0|y_t)\right) = \frac{p_0\left(y_t\right)P\left(\mathcal{H}_0\right)}{p(y_t)} \tag{3.10}$$

$$P\left(\mathcal{H}_1|y_t)\right) = \frac{p_1\left(1-P\left(\mathcal{H}_0\right)\right)}{p(y_t)} \tag{3.11}$$

where $P\left(\mathcal{H}_i|y_t\right)$ is the probability of $\mathcal{H}_i$ being true, and $p_i(y)$ is the probability density function for the measurement. This can also be rewritten as the likelihood ratio

$$\lambda = \frac{p_1(y_t)}{p_0(y_t)} \tag{3.12}$$

$p_1$ can be modelled as a white noise signal with a Gaussian probability distribution. The probability of detection versus the probability of false alarm can be modelled as a ROC(Receiver operating characteristics) curve where the probabilities are parameterized by a detection threshold[17, p. 205]. This curve will depend on the SNR and other environmental characteristics and is a useful tool for choosing the correct parameters in a model.

## 3.4 Types of Sonar

This section presents a brief overview of different sonar configurations. The two main types of sonars are passive and active. Passive sonars only listen for sound waves and active sonars transmit the sound waves as well. Most sonars use directional transducers, and they can be further divided by the use of a single transducer(single beam) or an array of transducers(multibeam) [5, p. 401-416]. Single and multibeam sonars are further described below.

Another categorization is imaging and profiling sonars. Profiling sonars only return the strongest echoes, which corresponds to the cross-section of the scanned object, while imaging sonars return all the echoes along the sonar beam. Profiling sonars are mostly used for depth profiles and bottom characterization, while imaging sonars are very useful for object detection and collision avoidance.

### 3.4.1 Single Beam Sonars

Single beam sonars have three possible configurations: Fixed, mechanically rotating and mechanically translating. Fixed sonars are not useful for object detection.

Mechanically rotating sonars use a stepper motor to rotate the sonar head a small step between each subsequent scan. At each step, the transducer sends out a highly directional pulse of acoustic energy. The transducer listens for a predefined time interval corresponding to the time the signal uses to travel to the set range and back again. When the listening period is done the stepper motor rotates one step and repeats the process. The frame rate is limited by the travel time for the sound, as well as the time it takes to rotate the transducer. For a 50 m range and a $180^\circ$ field of view, with steps of $3^\circ$, assuming a sound velocity of 1500 m/s the complete field of view is updated in approximately 4 seconds.

A side-scan sonar uses the same technology, but the difference is in the locomotion of the sonar head. The side-scan sonar is linearly translated, either by the movement of an AUV, a tow fish, etc. or by a motor.

### 3.4.2 Multibeam Sonars

A multibeam sonar system transmits one wide pulse of acoustic energy and the backscatter is received by an array of highly directional receivers. The time delay between the different receivers enable the sonar to distinguish the direction of the signal. Multibeam sonars requires much computing power and more complex electronics than single beam sonars, and this drives the price up. Due to the complex calculations, the frame rate in commercially available multibeam sonars is limited to around 10 frames per second at a short range [5, p. 403]. To allow the distinction of different echoes a technique called CHIRP is used.

### 3.4.3 Compressed High-Intensity Radar Pulse

Compressed High-Intensity Radar Pulse (CHIRP) is a great tool to increase the resolution from both single- and multibeam sonars. CHIRP sends out a signal consisting of a wide range of frequencies[5, p. 410-413]. This signal burst functions as a 'signature' and using signal processing techniques more of the noise can be filtered out and targets closer together can be distinguished. With a bandwidth of 100 kHz the resolution can be improved by a factor of 5. Another advantage over single or dual frequency sonar is the reduced power consumption from the high-speed digital circuitry.

## 3.5 Other Applications for Underwater Acoustics

In addition to sonars, underwater acoustics is also useful for communication and localization. These applications are briefly discussed below.

### 3.5.1 Communication

Sound waves transmitted from a hydrophone can be used to send information over greater distances than the electromagnetic(EM)-waves commonly used above the sea surface[45,

p. 3-4]. The main advantage over the EM-spectrum is small power loss as the waves propagate through water. The main disadvantage is the low propagation speed and bandwidth. The information is transmitted through phase- and frequency modulation.

### 3.5.2 Localization

Acoustic positioning is based on triangulation between transducers [25] and can be used with transducers mounted either on the seafloor, or on a support vessel. The time difference between the departure and arrival of the signals is used to calculate the position relative to each other. The positioning systems can be divided into three different classes, dependent on the distance between the transducers(baseline).

#### Ultra Short Base Line

Ultra-Short-Base-Line(USBL)/Super-Short-Base-Line(SSBL) systems normally mount one single unit, containing an array of three or four transducers, on a surface vessel [24, p. 29]. The baseline is normally less than 10 cm. The underwater vehicle has a transponder that sends out an acoustic pulse. The position is calculated by the time of flight and the bearing is calculated by using the phase difference between the transponders. The global position can then be calculated by using the navigation system of the surface vessel. USBL systems loose accuracy with depth, as the angular distance between the transducers, become smaller.

#### Short Base Line

Short-Base-Line(SBL) has a baseline from 5m to 20 m. Normally three to four hydrophones are placed on the surface vessel's hull[25]. The bearing and position is calculated in the same manner as for USBL. SBL systems are more expensive to install and requires more calibration than USBL systems, but they are more accurate.

#### Long Base Line

In Long-Base-Line(LBL) positioning the pulse transmitted from the vehicle transducer is measured by hydrophones fixed to the sea floor. The position and orientation are then calculated by triangulation. LBL systems are more accurate than SBL and USBL systems, but the disadvantage is that the hydrophones must be mounted on the sea floor and then calibrated.

Chapter 4

# Collision Avoidance

The task of avoiding collisions can be divided into three different parts: *object detection*, *path planning* and *path execution*. This chapter describes this process in depth, while presenting the developed algorithms. An overview of the different modules composing the complete system can be seen in the flowchart in Fig. 4.1.

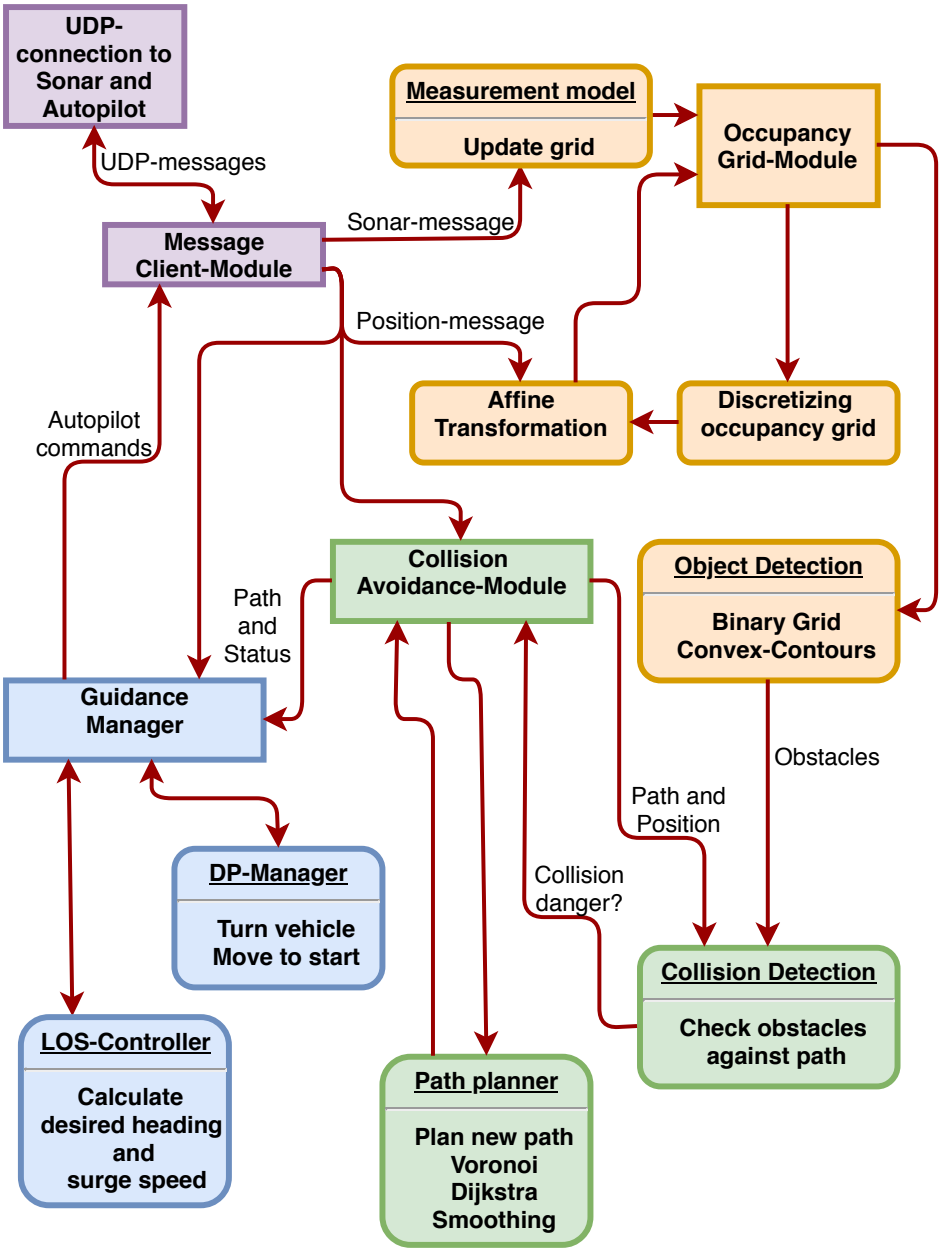



**Figure 4.1:** Flowchart of the object detection and collision avoidance process. The sharp-cornered boxes indicates a module, while the ones with rounded corners represent functions/methods for the corresponding module.

## 4.1 Object detection

The purpose of the object detection module is to use the available sensor data to generate a object map, that is as accurate as possible.

### 4.1.1 Occupancy Grid

A local occupancy grid is constructed as a an $m \times m$ matrix of real numbers. The position of the sonar is placed at $(c, c)$ with $c = \frac{m-1}{2}$ and the size of each cell is given as $l \times l$, where $l = \frac{r_{scale}}{c}$. The cell size is dependent on the current range of the sonar. Each cell contains the log-odds representation of the probability of occupation $P_O(i, j)$ of that cell.

### 4.1.2 Measurement model

When a new measurement is obtained, the occupancy grid has to be updated. The first step here is to determine if any of the return echoes could belong to an obstacle. Three different methods was used. The first method use a threshold value which is a function of the operators sonar settings. This method has the benefit of being easy to tune using the IKM sonar software to adjust the settings for base and span. The threshold value is transformed according to Eq. (4.1).

$$T_f = \frac{T \times \text{span}}{255} + \text{base} \tag{4.1}$$

where $T_f$ is the final threshold and $T$ is the selected threshold value. The data above $T_f$ is then selected as hits. This method can be observed in Fig. 4.2a.

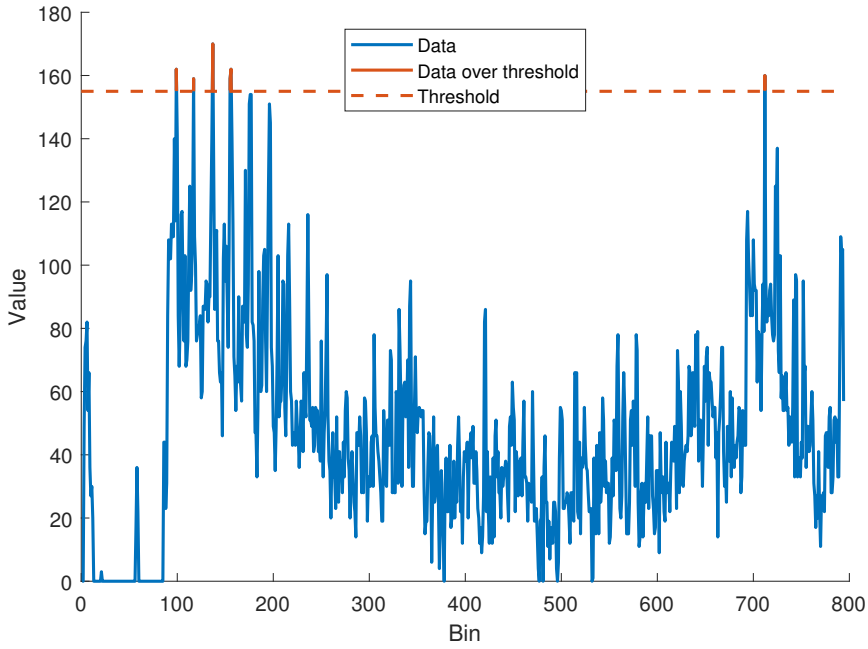
The second method analyses the shape of the signal. The method is described in Alg. 1, and an example can be observed in Fig. 4.2b.

**Algorithm 1:** Pseudo-code for threshold method 2.

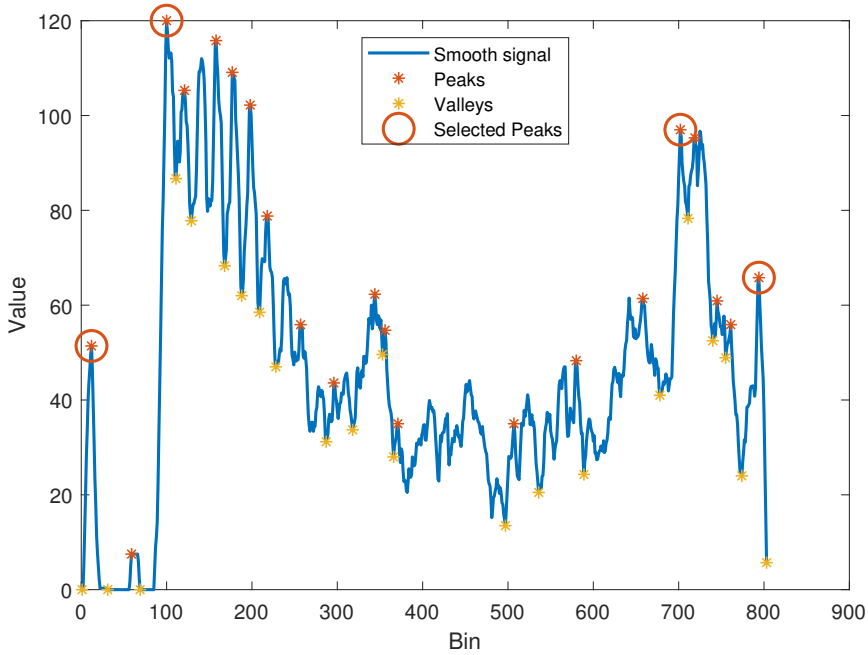
**Data:** $signal$, threshold $T$
**Output:** $p_{ret}$

$s$ = Convolution of $signal$ and $\boldsymbol{I}_{n\times 1}$
$p$ = All peaks in $s$, where a peak is a point with lower values on both sides
$v$ = All valleys in $s$, where a valley is a point with higher values on both sides
Remove consecutive peaks in $p$, such that only the tallest remains
Remove consecutive valleys in $v$, such that only the lowest remains
Remove insignificant peaks in $p$
Remove insignificant valleys in $v$
$p_{ret}$ = peaks in $p$ where $p - v > T$

The last method selects a threshold based on the value in the sample after the sample having the greatest gradient. The gradient has to be above a threshold $T_g$ to be considered a hit, otherwise the threshold will be selected as 255. An example of this method is shown in Fig. 4.2c.

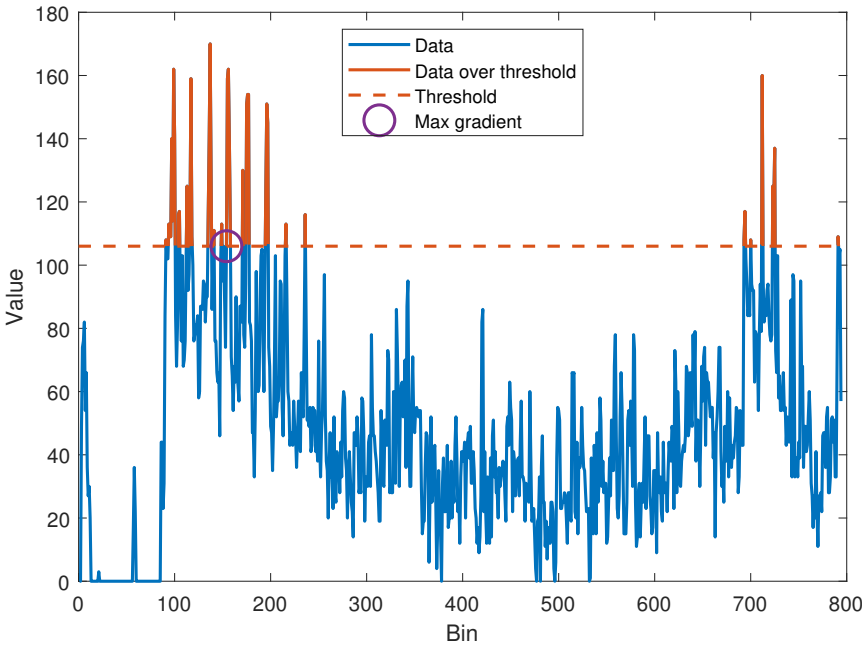



**(a)** Threshold Method 1.




**(b)** Threshold Method 2.

**Figure 4.2:** Three different methods for finding the return echoes over a certain threshold.




**(c)** Threshold Method 3.

**Figure 4.2:** Three different methods for finding the return echoes over a certain threshold (cont.).

The grid can now be updated in two different ways depending on the already detected obstacles. If the scan-line intersects an obstacle the Zhou-method[44] is used, otherwise the Thrun-method[37] is used. Both methods are described in Ch. 2.1.2. The intersection check is done by a binary-AND-check, where the matrix has the convex-obstacle-contours and the second matrix contains a straight line from the sonar position to the end of the grid, in the same direction as the scan-line. If the scan-line and a contour intersects, the angle between them is calculated as the angle between the scan-line and the first intersecting line-segment of the contour.

Both the Thrun-method and the Zhou-method requires the cells which overlaps the sonar-cone. An example of the overlap between the sonar-cone and the grid cells is shown in Fig. 4.3. This is done by a range-scale independent lookup table. The table is structured as two arrays of lists, one for each beam-width, where each row in the array corresponds to a sonar-cone angle discretized in steps of $\frac{1}{16}$gradian. This results in 6400 rows. Each row contains a sorted list of the indices of the cells in the corresponding sonar-cone. A cell is overlapping the sonar-cone if at least one of the angles are between the limits of the sonar-cone. The indices are sorted by the distance from the center. The indices corresponds to a flat indexing-scheme, where each row of the matrix is placed into a single row. This form of indexing speeds up the process.

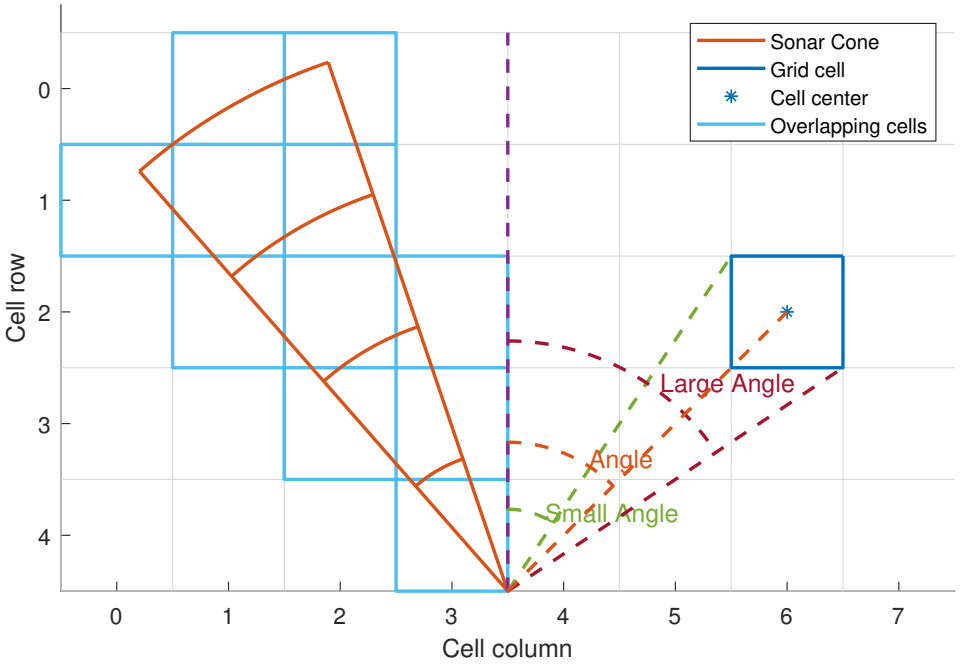



**Figure 4.3:** Overlap between grid cells and sonar-cone and angles used in the generation of lookup-tables.

### 4.1.3 Motion

The occupancy grid is vehicle fixed, and thus any motion of the vehicle needs to be taken into account. The motion can be modeled as probabilistic or deterministic, but due to the high accuracy of relative motion it is reasonable to model the motion as deterministic. Using the approach given by Ganesan et al. [13] the probabilities of occupation proved to diminish quickly, and thus a new approach is described here. Each cell in the occupancy grid, $G$ can be discretized as $n \times n$ smaller cells, such that the discretized grid, $G_d$ becomes $mn \times mn$ cells. The discretized grid is centered at $(c_d, c_d)$ with $c_d = \frac{m*n}{2}$ and the size of each cell is given as $l_c \times l_d$, where $l_d = \frac{r_{scale}}{c_d}$. The relation between the two grids are then given by Eq. (4.2), where $\otimes$ is the Kronecker product.

$$G_d = G \otimes \boldsymbol{I}_{n \times n} \tag{4.2}$$

The rotation and translation of the grid is archived through an affine-transformation, as described in Ch. 2.2.6 with a transformation-matrix, $\boldsymbol{T}$ as described in Eq. (4.3) and linear interpolation over the transformation. The border points are chosen as $p_{initial}$.

$$\boldsymbol{T} = \begin{bmatrix} \alpha & \beta & (1-\alpha)\, c_d - \beta c_d - \Delta x_g \\ -\beta & \alpha & \beta c_d + (1-\alpha)\, c_d + \Delta y_g \end{bmatrix} \tag{4.3}$$

, where the variables are defined in Eq. 4.4.

$$\begin{aligned} \alpha &= \cos \Delta\psi \\ \beta &= \sin \Delta\psi \\ \Delta x_g &= \Delta y_{veh} \frac{c_d}{r_{scale}} \\ \Delta y_g &= \Delta x_{veh} \frac{c_d}{r_{scale}} \end{aligned} \tag{4.4}$$

, where $g$ denotes the grid reference frame and $veh$ denotes the vehicle reference frame.

### 4.1.4 From occupancy grid to obstacles

Going from the probabilistic occupancy grid representation to an obstacle representation suited for the path planning algorithm involves several steps. These are listed below.

- Binary thresholding
- Contour detection
- Filtering out small contours
- Dilation to include safety margins
- Finding contours of dilated map
- Approximating contours as convex hulls

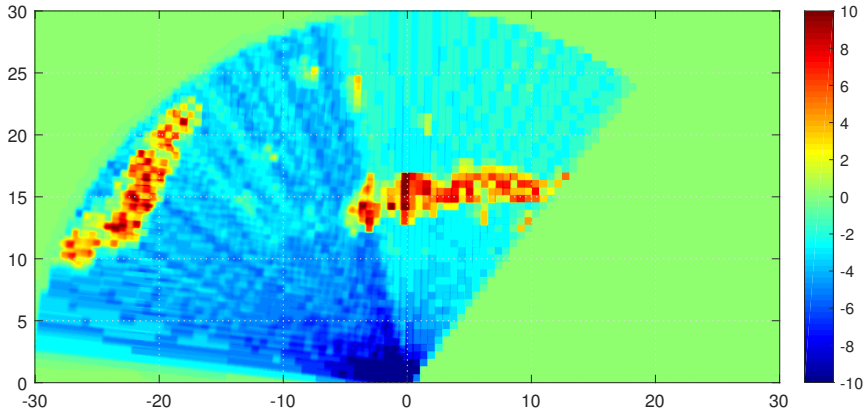
The the thresholding process is shown in Eq. (4.5).

$$\boldsymbol{B_d}(i,j) = \begin{cases} 1 & \text{if } \boldsymbol{P_d}(i,j) \geq P_{th} \\ 0 & \text{if } \boldsymbol{P_d}(i,j) < P_{th} \end{cases} \tag{4.5}$$

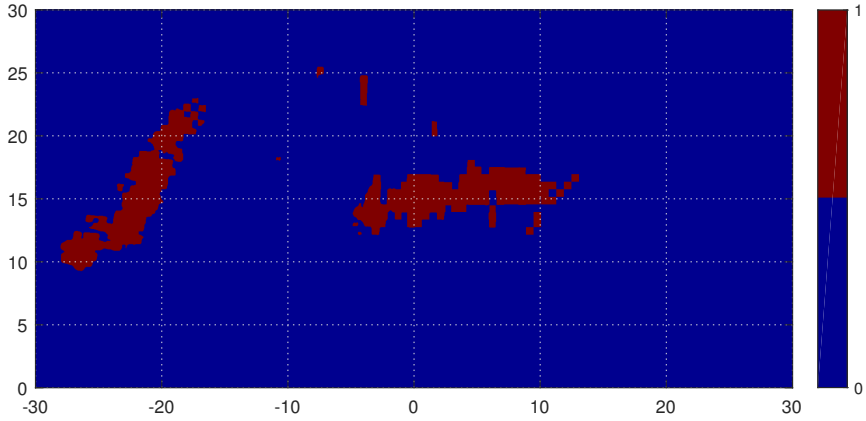
where $\boldsymbol{B_d}$ is the discretized binary grid and $P_{th}$ is the thresholding value. To save computation time, $P_{th}$ can be log-odds representation of the probability. The contours are then extracted as described in Ch. 2.2.5 and the areas of each contour is calculated. All contours with an area smaller than a certain threshold is filtered out, as these are likely due to noise or obstacles such as fish.The grid is then dilated, as described in Ch. 2.2.4, with a structuring element, $\boldsymbol{E}$ given in Eq. (4.6)

$$\boldsymbol{E} = \mathbf{1}_{n \times n} \quad \text{, where } n = \frac{Dc_d}{r_{scale}} \tag{4.6}$$

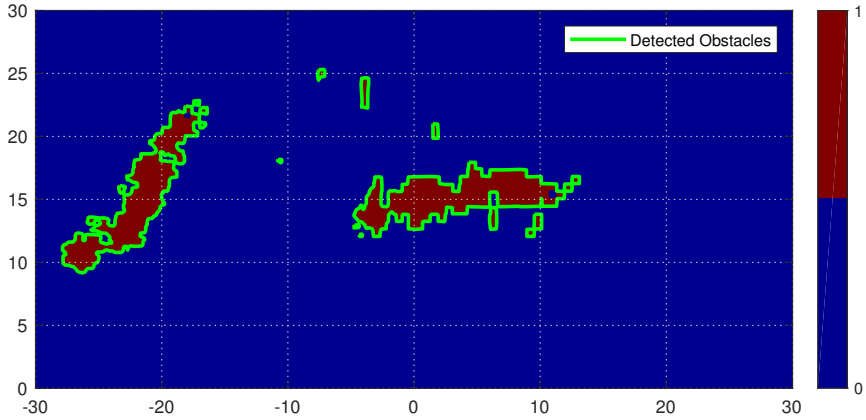
where $D$ is the safety margin. The contours of the dilated image is then extracted as a convex hull. The complete process can be observed in Fig. 4.4. After the objects are detected the next step is to check if there is a collision danger.




**(a)** Original occupancy grid. The plot is in the log-odds scale.

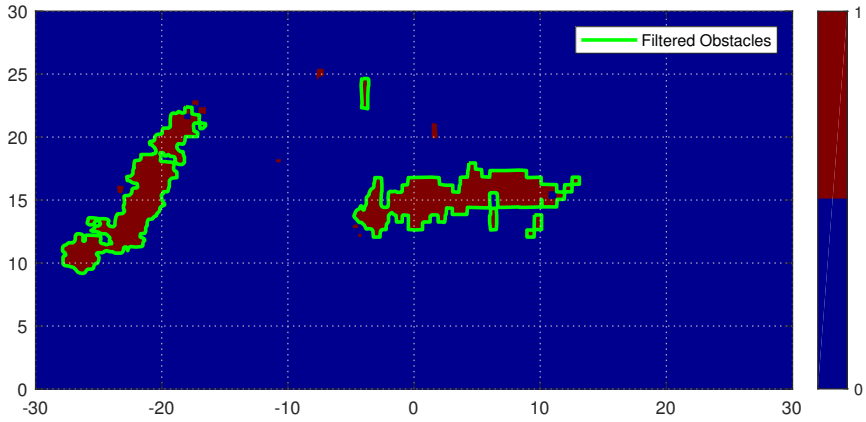



**(b)** Binary grid.

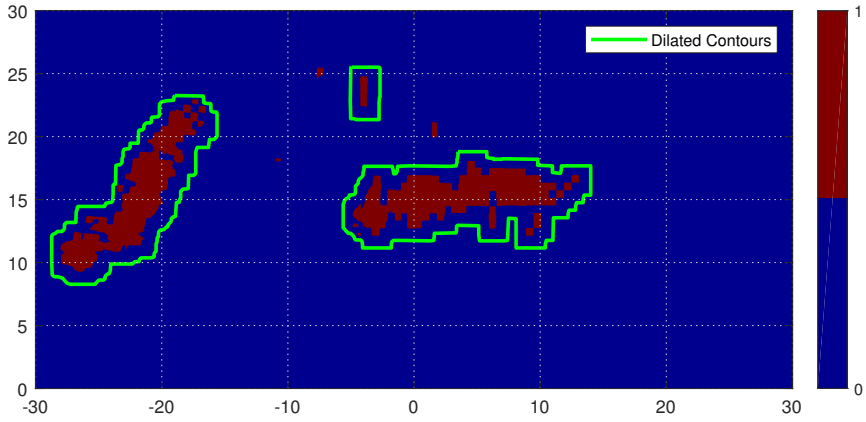



**(c)** First contour detection

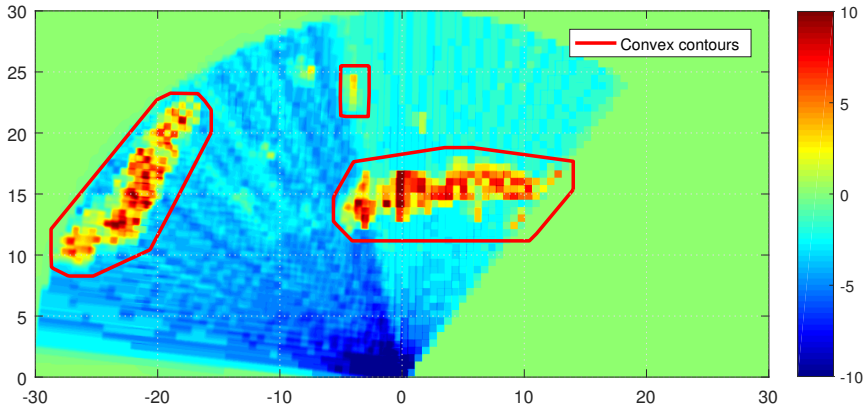
**Figure 4.4:** Object detection procedure.




**(d)** Small contours filtered away.




**(e)** Dilated contours.




**(f)** Convex contours on top of original occupancy grid. The plot is in the log-odds scale.

**Figure 4.4:** Object detection procedure (cont.).

## 4.2 Collision Detection

Collision detection is an important part of the system. A simple way of doing this is to draw the path and the obstacles on two different binary images and do a logical AND-check. This procedure is illustrated in algorithm 2. The path is drawn as a line, with a thickness corresponding to the width of the vehicle. The collision margin around the obstacles are already implemented during the the object-detection part.

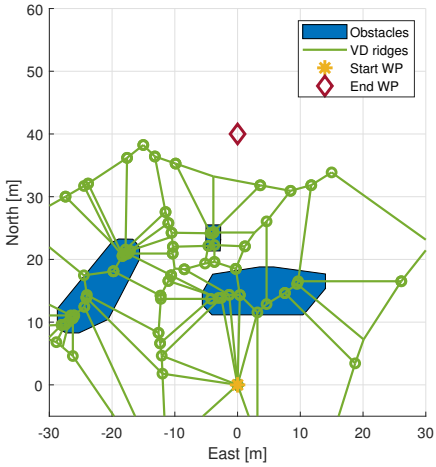
**Algorithm 2:** Pseudo-code for collision check

**Data:** $waypointList$, $obstacleContours$, $rangeScale$, $gridSize$, $vehicleMargin$
**Output:** $collisionIndex$, -1 if no collision

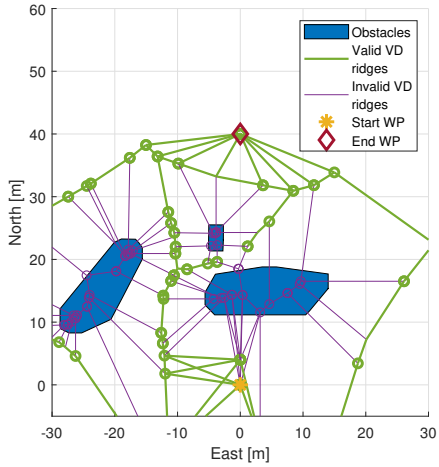
$obstacleGrid =$ binary drawing of all $obstacleContours$
$pathWidth = vehicleMargin \times gridSize/(2 \times rangeScale)$
$pathSegments =$ segments between a waypoint and the next
$collisionIndex = -1$
**for** *segment* $s$ *of* $pathSegments$ **do**
    $pathGrid =$ binary drawing of segment with $pathWidth$
    **if** *any item in* $(pathGrid \&\& obstacleGrid) = True$ **then**
        $collisionIndex = i$
        break

## 4.3 Path planning

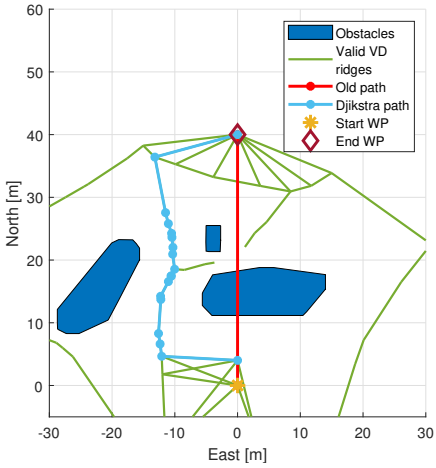
The purpose of the path planning module is to calculate a new and effective path once a collision has been detected. An example of the path planning process could be seen in Fig. 4.5. The process is described below.



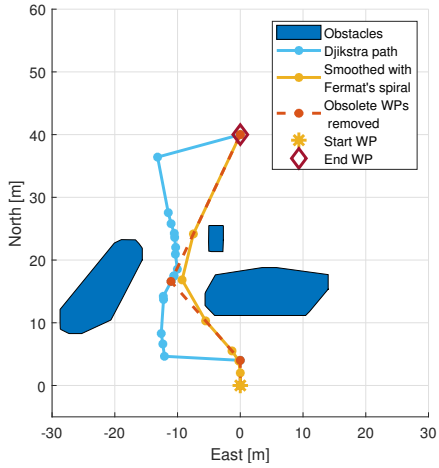

**(a)** Voronoi diagram.




**(b)** Voronoi diagram with invalid vertices and ridges removed. Connections between the first and final waypoints and the vertices of their respective region are added.




**(c)** Voronoi diagram with only valid paths. The old path is shown together with the first step of calculating the new path.




**(d)** The path from Fig. 4.5b with obsolete waypoints removed and the smooth final path.

**Figure 4.5:** Voronoi diagram and path calculation.

### 4.3.1 Voronoi Diagrams

The Voronoi diagrams (VD) described in section 2.3 can be implemented into a path planning algorithm. The generator points can be chosen as the vertices of of the contour points from the object detection algorithm, and thus the set $\boldsymbol{V}$ will consist of the points furthest away from each obstacle. This can be observed in Fig. 4.5a, where the green circles

indicate a vertex of $\boldsymbol{V}$ and the green lines is the set $\boldsymbol{V}$ itself. In addition to this four corner-points are added. The positions of these corner-points are chosen as the outermost points of the occupancy grid corners and the last waypoint included in the optimization process. It is also possible to add the center-points between the corner-points to get additional possible paths. To get a set of safe piece-wise linear paths, all paths intersecting with an obstacle has to be removed. This can be observed in Fig. 4.5b, where the invalid paths and vertices is purple.

The start and end positions also has to be included to get a complete path. These positions are included as a vertex, which are connected to all other vertices surrounding the corresponding region. In addition to these two points, another fixed point is inserted straight ahead of the vehicle to account for the run-time of the algorithm. The position of this point is a tuneable parameter, that should be tuned according to the cruising velocity of the vehicle.

The next step is to find the shortest valid path, and here Dijkstra's algorithm is a good candidate.

### 4.3.2 Dijkstra's algorithm

The first step in Dijkstra's algorithm is to populate the distance matrix $\boldsymbol{D}$ with the distances of each valid path in the Voronoi diagram. $\boldsymbol{D}(i,j) = \boldsymbol{D}(j,i)$ is the distance between node $i$ and $j$. A distance of zero indicates that there is no connection. Running Dijkstra's algorithm, as described in section 2.4 will return the shortest path between the two waypoints.

### 4.3.3 Removal of unnecessary waypoints

The path found by Dijkstra's algorithm is the optimal solution to the shortest path along the edges of the VD, but this is not the overall optimal solution. The VD generates edges that are as far away as possible from the obstacles, which results in a path that has a greater clearance-margin than necessary, as well as containing too many waypoints. An algorithm inspired by [23] is developed to remove the unnecessary waypoints. First, almost co-linear waypoints are removed based on path-segment heading threshold. The next step is to remove waypoints that has a greater clearance-margin than necessary, which is done by checking if a waypoint can be removed, without violating the clearance constraints. If a waypoint can be removed, a check is performed to see if removing the next waypoint is more beneficial. This process is outlined in Alg. 3.

**Algorithm 3:** Algorithm for removal of unnecessary nodes. The $P_{a\to b\to c}$ notation is the path segment from node $n_a$ trough node $n_b$ to node $n_c$, and $|P|$ is the length of path segment $P$. collisionDanger() is function checking whether a path-segment is violating clearance constraints.

**Data:** List of points in optimal path, $P_L$
**Output:** $P_L$ with unnecessary waypoints removed

```
for nodes n_i in P_L do
    P_{0→2} = n_i → n_{i+2}
    if collisionDanger(P_{0→2}) then
        P_{1→3} = n_{i+1} → n_{i+3}
        if collisionDanger(P_{1→3}) then
            if |P_{0→2→3}| > |P_{0→1→3}| then
                Remove n_{i+2}
        else
            Remove n_{i+1}
    else
        Remove n_{i+1}
```

The resulting path consists of linear path-segments, which gives a non-continuous curvature. Flying this path with a fully-actuated vehicle will cause the vehicle to slow down to a full stop between each path-segment, which is ineffective. The path is unfeasible for a under-actuated vehicle. To solve this problem, the path has to be smoothed to ensure a continuous curvature.

### 4.3.4 Fermat Spirals

The smoothing of the calculated path is done as described in Ch. 2.5. Once a feasible and smooth path is generated it is translated to the NED-frame as described in Ch. 4.3.5 and sent to the guidance system along with a status update from the algorithm. The possible statuses and their corresponding responses are described in Tab. 4.1. An example of the smoothed path can be seen in Fig. 4.7b.

The resulting smooth path might violate the clearance constraints. In this case, Dijkstra's algorithm has to be run again, but with a modification suggested by [23]. This modification erases the connection to the first node resulting in a constraint violation and calculates a new path from there. The algorithm can be run multiple times to generate a feasible path. If the algorithm can not find a feasible and smooth path, the guidance system receives the non-smoothed path.

### 4.3.5 Conversion between coordinate systems

Conversion between the different reference frames and coordinate systems are frequent. In addition to the transformations presented in Ch. 2.7 it is also necessary to transform

waypoints from $g$ to $veh$, which is shown in Eq. (4.7). The conversion back is presented in Eq. (4.8)

$$\begin{aligned} x_{veh} &= (c_y - y_g)\,\frac{r_{scale}}{c_y} \\ y_{veh} &= (x_g - c_x)\,\frac{r_{scale}}{c_x} \end{aligned} \tag{4.7}$$

where $c$ is half the number of cells in the associated direction.

$$\begin{aligned} x_g &= \text{round}\left(c_x\left(\frac{y_{veh}}{r_{scale}} + 1\right)\right) \\ y_g &= \text{round}\left(c_y\left(1 - \frac{x_{veh}}{r_{scale}}\right)\right) \end{aligned} \tag{4.8}$$

where round is the rounding and conversion to an integer. This is necessary as the grid reference-frame consists of a finite number of cells, rather than a position.

The transformation of a position from the vehicle frame of reference to the NED-frame is shown in Eq. (4.9) and back again in Eq. (4.10)

$$\begin{aligned} r &= \sqrt{x_{veh}^2 + y_{veh}^2} \\ \alpha &= \arctan\left(\frac{y_{veh}}{x_{veh}}\right) \\ N &= p_N + r\cos\left(\alpha + \psi\right) \\ E &= p_E + r\sin\left(\alpha + \psi\right) \end{aligned} \tag{4.9}$$

$$\begin{aligned} r &= \sqrt{(N - p_N)^2 + (E - p_E)^2} \\ \beta &= \arctan\left(\frac{E - p_E}{N - p_N}\right) - \psi \\ x_{veh} &= r\cos(\beta) \\ y_{veh} &= r\sin(\beta) \end{aligned} \tag{4.10}$$

The relations between the coordinate systems are shown in Fig. 4.6.

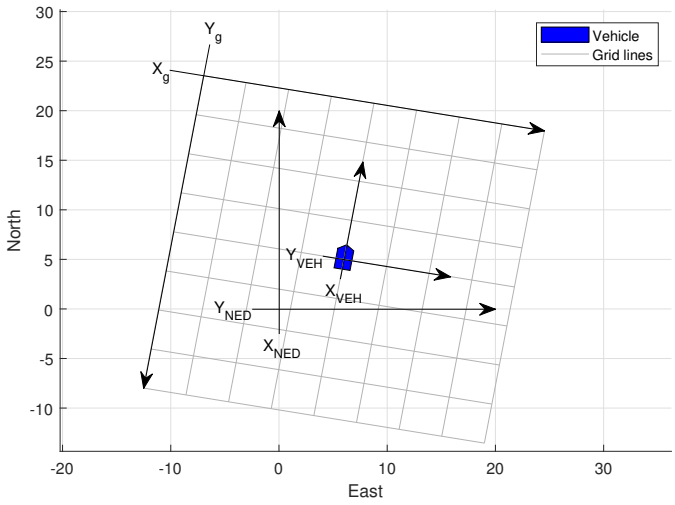



**Figure 4.6:** Relationship between the $n$-, $veh$- and $g$-reference-frames.

## 4.4 Guidance and Control

The guidance-system is responsible for providing input to the vehicles control system. A LOS-guidance system with surge-velocity control is implemented as described in Ch. 2.8, with a few modifications.

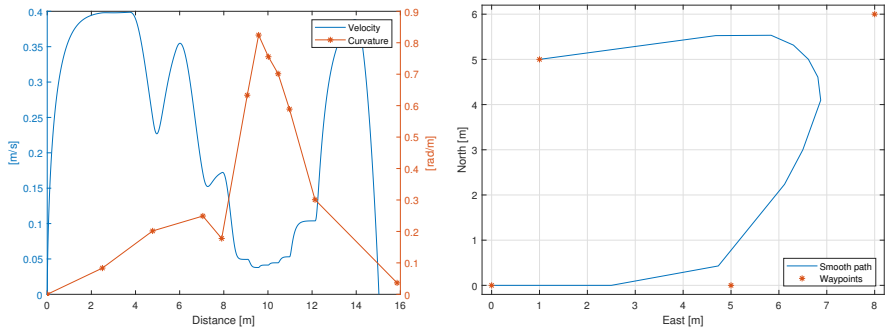
The lookahead-distance is implemented as a lookahead-time, such that lookahead-distance is a time-varying parameter dependent on the vehicle velocity. The region of acceptance is reduced if the curvature is large, resulting in close waypoints.

The setpoint for the surge-controller is also time-varying. When a new path is calculated, the guidance-system will calculate the curvature for each path-segment-corner and subsequently choose the correct velocity for the turn. The turn velocity is implemented as a constant multiplied by the curvature between the path-segments. An example of the velocity along the path in Fig. 4.7b is shown in Fig. 4.7a.

As the vehicle will need some time to reduce the velocity to the turn-velocity a break-distance is implemented in the same manner as the region of acceptance.

The guidance system tries to make the transition between paths as smooth as possible, which means that as long as the vehicle is on the new path, or sufficiently close to it, the new path is started immediately. If these conditions are not satisfied, the vehicle will stop and enter DP-mode to reach the start of the new path.

The guidance system receives status updates from the collision avoidance algorithm. The different statuses and their corresponding responses are listed in Tab. 4.1. The *No feasible path* status triggers the guidance system to a full stop. The mean values of the starboard and port sides of the occupancy grid is then calculated, such that the vehicle can start to turn towards the side with the smallest obstacle density.




**(a)** Curvature and velocity profile along the path.

**(b)** Smooth path and original waypoints.

**Figure 4.7:** A smooth path and the original waypoints are to the right. The velocity profile and curvature of the path is to the left.

| Status | Response |
|---|---|
| No danger | Continue current path |
| No feasible path | Stop and turn |
| Smooth path violates collision margin | Execute non-smooth path, with stops between WPs |
| New route OK | Execute new path |

**Table 4.1:** Possible collision avoidance statuses, and their corresponding response in the guidance system

In addition to the LOS-guidance system, a control system was also needed for the simulations. It was decided that a simple PID-controller was sufficient for controlling altitude, heading and surge-velocity. The tuning was straightforward due to the noise-free signals.

Chapter 5

# Experimental setup

In this chapter, the experimental setup during the simulations and field experiments are described. The chapter is divided into one section for each of the simulators and one section for the field experiments. The computer used for the MORSE simulations and the collision avoidance system is described in Ch. 5.4.

## 5.1 Simulation with MORSE

The initial simulations were done with the modified version of the MORSE simulator developed by Henriksen et al. [15]. The communication was done trough MOOSDB, which provided an easy communication interface. A description of the simulated environment and the ROV is provided in Ch. 6. A screenshot of the ROV can be seen in Fig. 5.1.

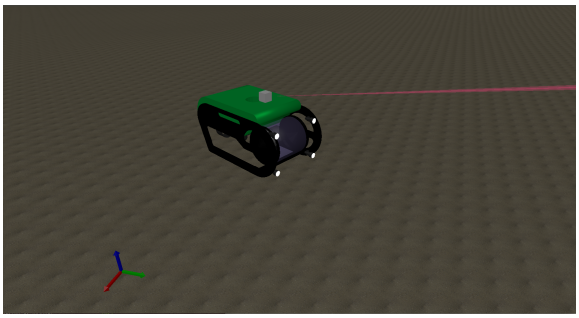


**Figure 5.1:** Close up screenshot of the ROV used in the MORSE simulations.

## 5.2 Simulation with Merlin Simulator

The Merlin Simulator is a high-fidelity ROV simulator, complete with a normal control system. This is a useful simulator for testing the communication with the IKM Autopilot server and getting some initial tuning values for the field experiments. The simulator lacks the possibility of extracting sonar data, and thus no object detection or collision avoidance tests could be carried out. An example of a simulated environment in the simulator is shown in Fig. 5.2.

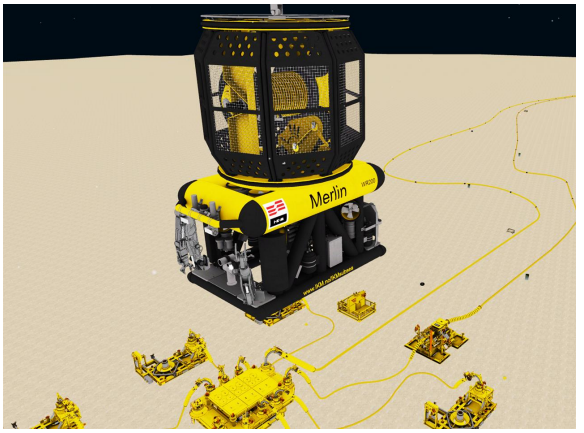



**Figure 5.2:** Screenshot from the simulated environment in IKM's simulator.

## 5.3 Field Experiments

All field experiments are performed at Equinor's Snorre B field. The field is located in the 34/4 block. The approximate position of the semi-submersible drilling rig can be observed in Fig. 5.3.

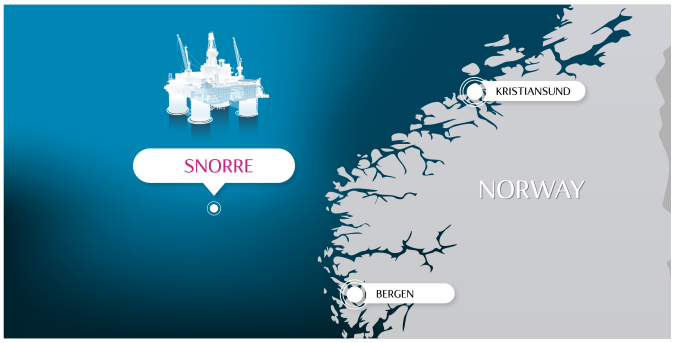



**Figure 5.3:** Location of the Snorre field[33].

### 5.3.1 Equipment

The equipment used in the field experiments include the ROV, the sonar and the positioning system. This equipment is described below.

#### Merlin UCV

The drilling rig has a resident ROV, which means that it is permanently stationed subsea. A 3D-model of the ROV is shown in Fig. 5.4. The ROV is called Merlin UCV and is a work-class ROV, with electric propulsion. The dimension of the ROV $2.5m \times 1.5m \times 1.5m$.

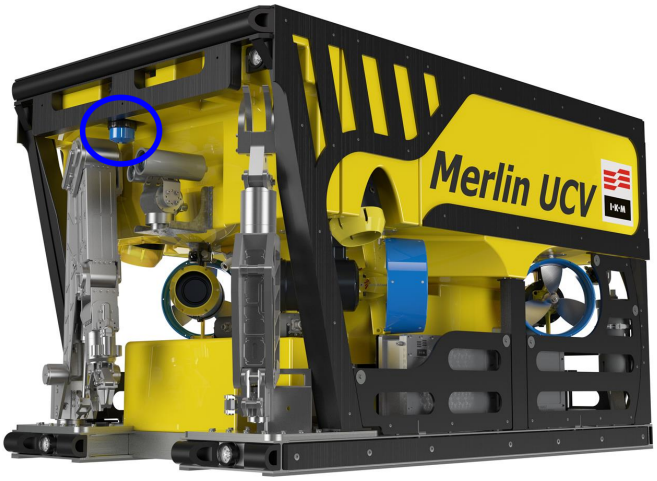



**Figure 5.4:** 3D model of Merlin UCV, with the location of the sonar highlighted. Courtesy of IKM Technology.

#### Onshore Control Room

The ROV can be controlled from an onshore control room at IKM's headquarters at Bryne. A picture from the control room can be seen in Fig. 5.5.

**Figure 5.5:** Picture from the control-room at IKM Subsea's headquarters at Bryne. Courtesy of IKM Subsea.

### Tritech SuperSeaking Mechanically Scanning Sonar

The Merlin UCV is equipped with a mechanically scanning sonar called SuperSeaking DST. This sonar is manufactured by Tritech. The SuperSeaking sonar is utilizing Digital Sonar Technology(DST) as well as Compressed High-Intensity Radar Pulse(CHIRP). The sonar has two separate transducers with frequencies centered around 650 kHz and 325 kHz to accommodate high resolutions and long ranges.

The SuperSeaking sonar has an RS-232 interface for serial communication. The serial data is then sent over Ethernet as UDP-packages. The communication protocol is defined in [39]

Some key properties of the SuperSeaking sonar are listed in tab 5.1.

| Specifications | High-Frequency | Low-Frequency |
|---|---|---|
| Frequency center | 650 kHz | 325 kHz |
| Horizontal beam width | 1.5° | 3.0° |
| Vertical beam width | 40.0° | 20.0° |
| Maximum range | 100 m | 300 m |
| Minimum range | 40 cm | |
| Maximum range resolution | 15 mm | |
| Mechanical resolution | 0.45°, 0.9°, 1.8° and 3.6° | |
| Source level | 210 dB re 1 µm | |

**Table 5.1:** Key properties of Tritech SuperSeaking DST Sonar[38]

### Inertial Navigation System

The Merlin ROVs are equipped with an INS called Sprint 500, which uses a combination of accelerometers, gyroscopes, Doppler velocity log (DVL), depth sensors and position measurements from transponders to estimate the ROV's position.

### Acoustic Positioning System

The rig at Snorre B is equipped with a super-short-base-line acoustic positioning system, see section 3.5.2, from Kongsberg called HPR-400. According to [1] the accuracy is approximately 6 meters.

### Communication

The test computer was connected to the wired network in the control room and messages were sent over UDP to the IKM Autopilot Server (APserver). The APserver communicates with the with the control room, which forwards the commands offshore. This process is outlined in Fig. 5.6.

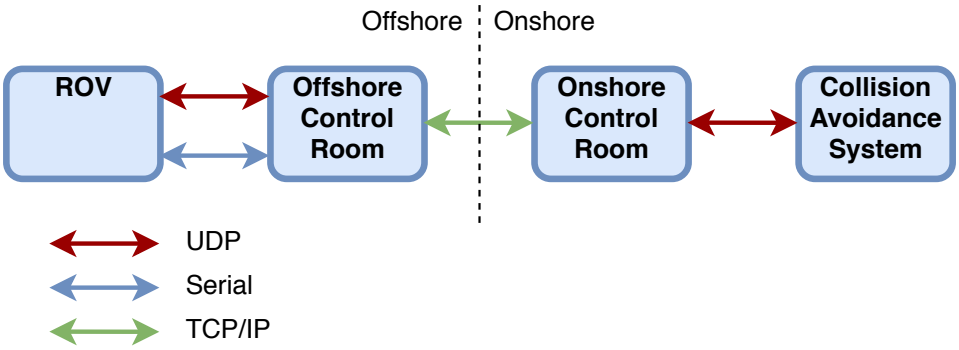



**Figure 5.6:** Communication with ROV.

## 5.4 Computer

The collision avoidance system was run on an *Acer Aspire Nitro VN7-791G* with a 4-core *Intel Core i7-4720HQ* processor with a clock frequency of 2.6GHz [19]. The laptop has 8 GB of *DDR3L SDRAM*, with a clock frequency of 1600 MHz and a *NVIDIA GeForce GTX 860M - 2 GB GDDR5 SDRAM* graphics processor. The graphics processor was not used for the collision avoidance algorithm, but is necessary for the running the simulations through MORSE/Blender.

Chapter 6

# Simulation Results

In this chapter the results from several simulations are presented. The first three sections contains different simulations, using MORSE to test the guidance system, the object detection system and the complete collision avoidance system. The last section is a test of the guidance system, and the communication with the IKM's Autopilot program, using their simulator.

The simulated environment used in the MORSE simulations contains several simple obstacles, such as ellipsoidal and rectangular pillars, as well as a more complex obstacle consisting of three small cylinders and a wall on one side. A map of this environment is shown in Fig. 6.1 and a picture of the environment is shown in Fig. 6.2. In all the experiments the sonar was running with an update frequency of 16 Hz, which is approximately equal to the update frequency of the sonar used in the field trials. The range of the sonar was 30 m and the beam width and height was equal to those of the SuperSeaKing sonar. The shorter range is due to the available ROV model being smaller than the Merlin UCV. A Gaussian distributed noise, with a mean of 10 and standard deviation of 2.9 was added to the sonar-signal. The cruising velocity was set to $0.4$m/s $\approx 0.8$knots. The ROV is quite different from the one used in the field-trial, but the result are still useful.

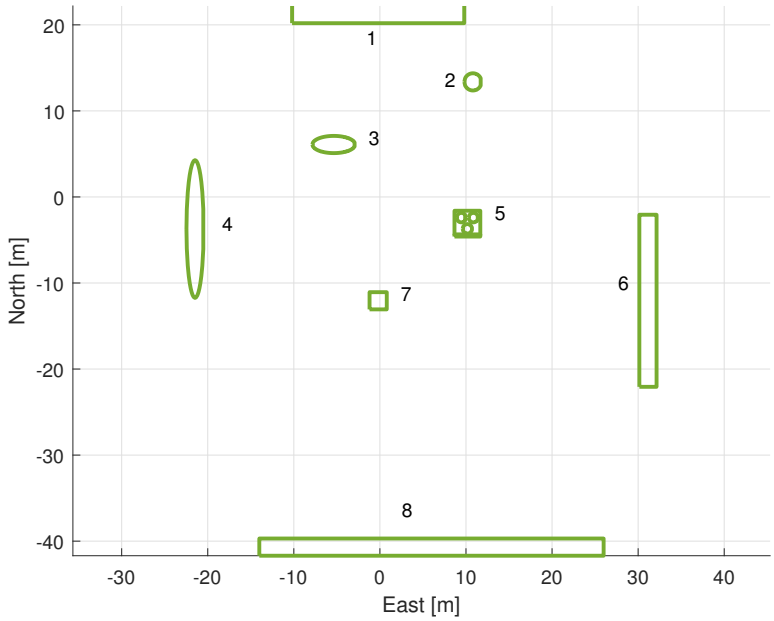



**Figure 6.1:** Map of the simulated environment used during the MORSE-simulations. The objects are numbered.

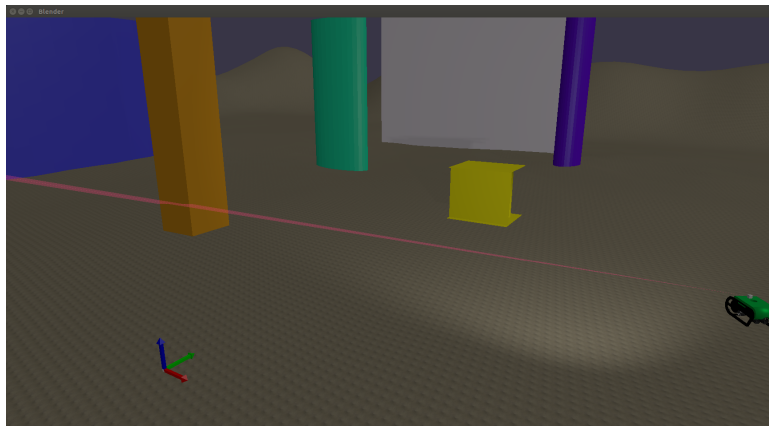


**Figure 6.2:** Screenshot from the simulated environment used during the MORSE-simulations.

The simulations carried out in IKM's simulator was for testing the communication with the autopilot server. The simulator has high fidelity dynamics, but there is no possibility of outputting sonar-data for usage in the system, and thus no test were carried out on the object detection part. The test where run without a tether, as no automatic tether system is implemented.

## 6.1 Object Detection

The object detection system was tested in several different trials on the simulated environment described above. The vehicle was flying pre-planned paths or controlled manually. The results from one of these experiments is shown in Fig. 6.3, where the vehicle is flying a pre-planned path. The detected obstacles are logged every 10 seconds, and plotted as transparent shapes, such that obstacles that are detected on multiple occasions are less transparent. It can be observed that all the objects are clearly detected. There are also objects detected at $(N, E) \approx (-5, -45), (40, -20)$ and $(10, -40)$, which are caused by steep terrain elevations at these positions, which can be observed in the background in Fig. 6.2. There is also an object at $(N, E) \approx (3, -15)$, which is due to cluttered noise. As can be observed from the opacity of the object, it was only detected for a short while, and as soon as the ROV moved, the probability was below the threshold again. It should be noted that all the objects appear larger than they are, which is due to the applied safety-margin of two meters.

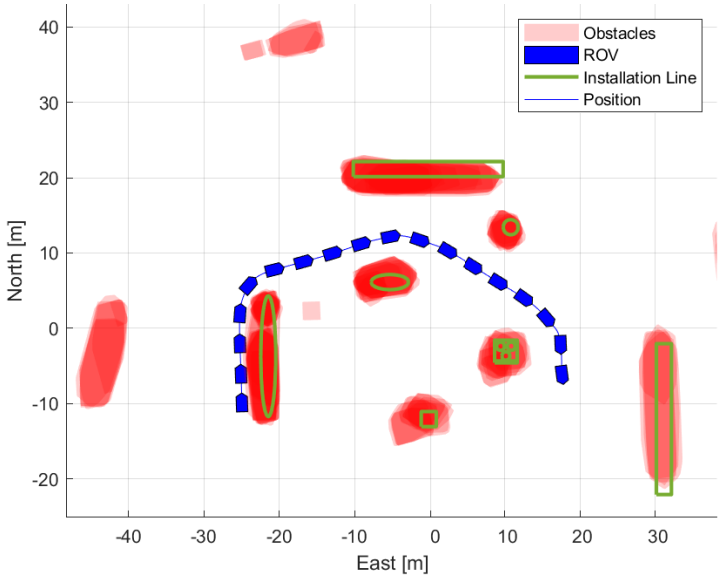



**Figure 6.3:** Results from simulated object detection trial. The detected obstacles are logged every 10 seconds and plotted as transparent polygons.

## 6.2 Guidance System

In this section a trial of the guidance system is presented. The ROV is flown trough a path in the simulated environment. A plot of the path is shown in Fig. 6.4. The figure shows both the planned path and the flown path, as well as the ROAs for each waypoint (WP). It should be noted that there is a deviation from the path after WP9, as the ROV does not manage to slow down sufficiently. This deviation can also be seen in Fig. 6.7, where the

cross-track error is shown. The reason for this is a too high velocity, which can be observed in Fig. 6.8, where an excerpt from the velocity profile is shown. The guidance system tries to reduce the velocity before WP9, but the distance available for the velocity reduction is not sufficient, and thus the velocity trough the turn is too large. There is a large deviation at WP4 as well, but this is mainly due to the large angle between the path-segments and the size of the region of acceptance (ROA). As the ROV reach the ROA of WP4, the path segment between WP4 and WP5 is selected, but the ROV is still not on the path. It should be noted the ROV reduces the cross-track error fast, as it converges towards the new path segment.

In Fig. 6.5 a plot of the heading throughout the path is displayed. It should be noted that the setpoint is too fast for the ROV, but the reference signal smooths out the steps in the setpoint. The difference between the heading angle and the reference signal can be seen in Fig. 6.6. This error is sufficiently small, except right after the change to WP4. This suggests that the acceleration of the reference signal should have been smaller.

The overall performance of the guidance system is sufficient, but the large cross-track error after WP9, might be a problem when a path is close to an obstacle.

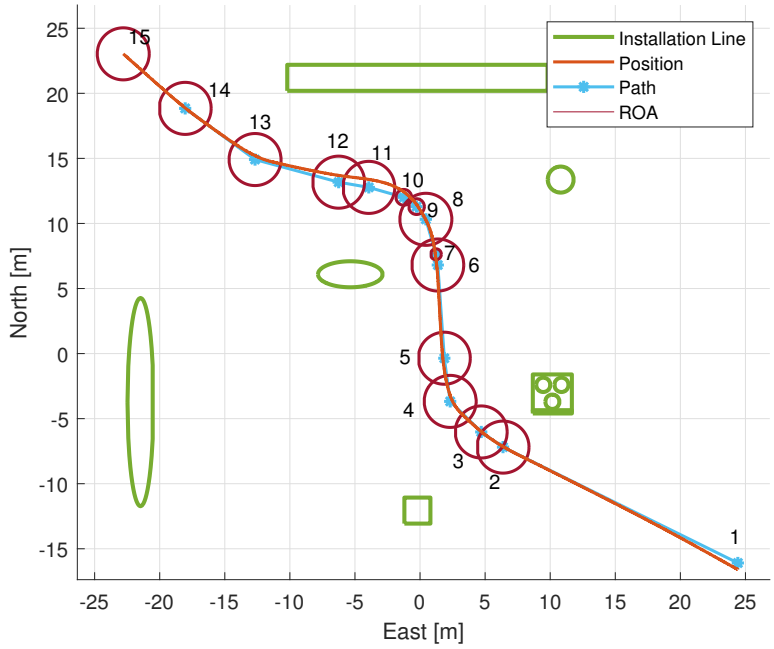



**Figure 6.4:** North-east plot of the path of the ROV during simulated guidance system trial. The ROAs are plotted at each waypoint.

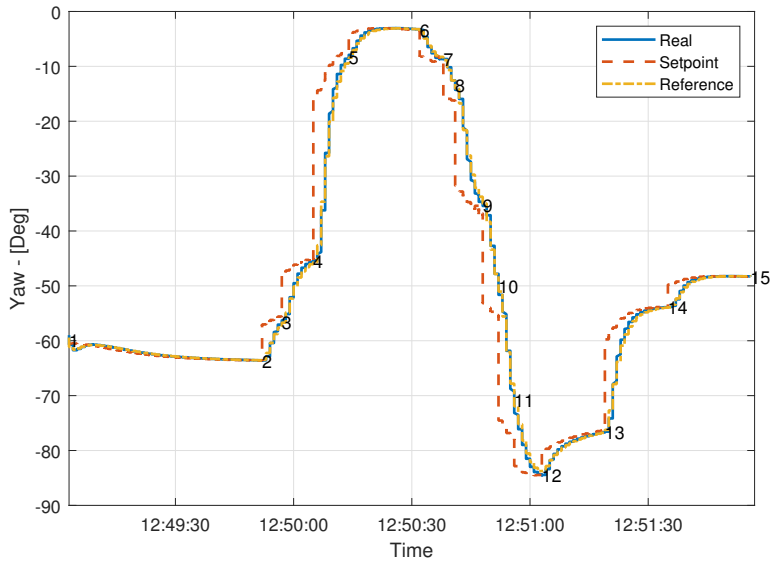



**Figure 6.5:** Plot of the heading, the reference signal and setpoint of the heading, during simulated trial of the guidance system.

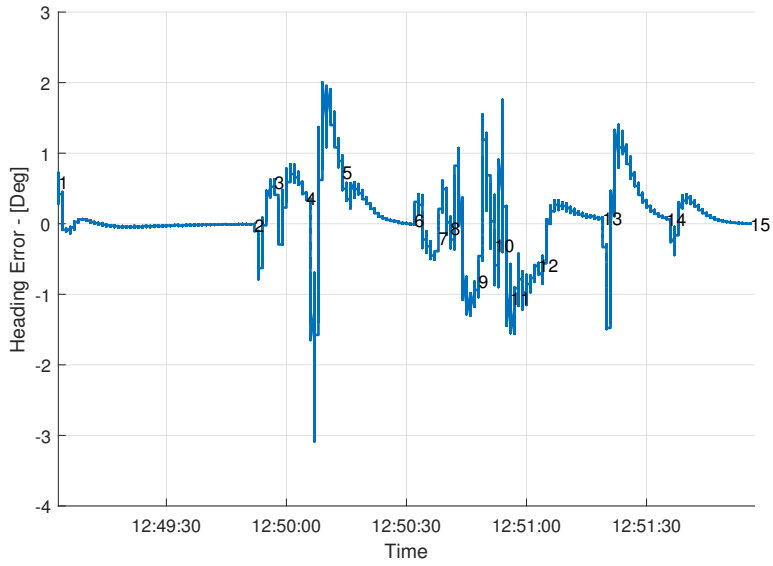



**Figure 6.6:** Plot of the difference between the heading and the reference signal of the heading, during simulated trial of the guidance system.

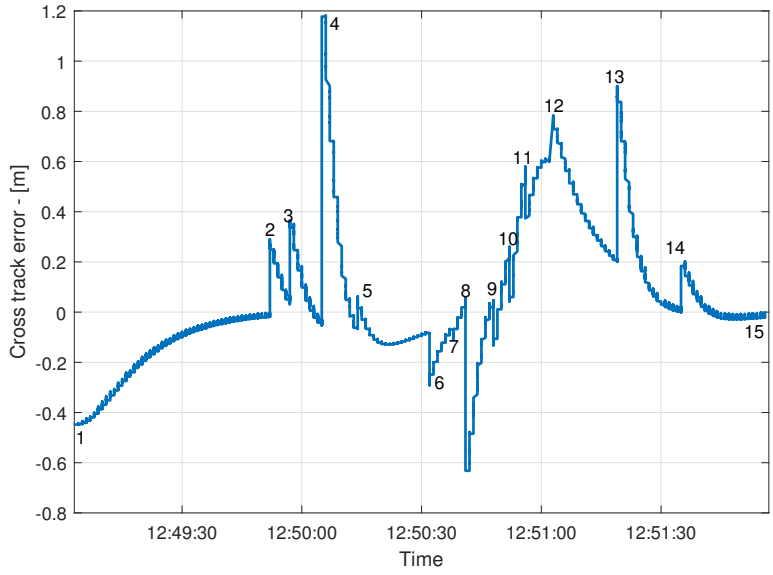



**Figure 6.7:** Plot of the cross-track error during simulated trial of the guidance system.

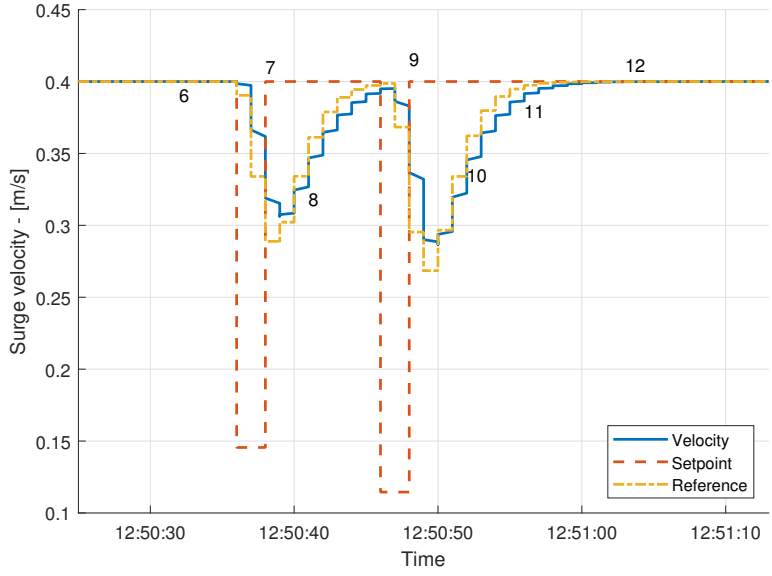



**Figure 6.8:** Plot of the surge velocity of the ROV during simulated trial of the guidance system. The plot also shows the reference signal and the setpoint.

## 6.3 Collision Avoidance

In this section a collision avoidance test is performed in the simulated environment. An excerpt from the results is provided in Fig. 6.9-6.11. In Fig. 6.9 the first re-planning of the path is presented. It should be noted that only object 7 is detected as this point, even though both object 5 and 6 is within the sonar field of view. The reasons for this is that the sonar beam started from the left, and has not yet reached these objects. The new path is planned in a wide curve around the detected obstacle, which is due to the nature of VDs. The VD finds the lines which are as far apart from the generator points as possible, and since only one obstacle is present the only present generator points are the edges of the grid and the obstacle.

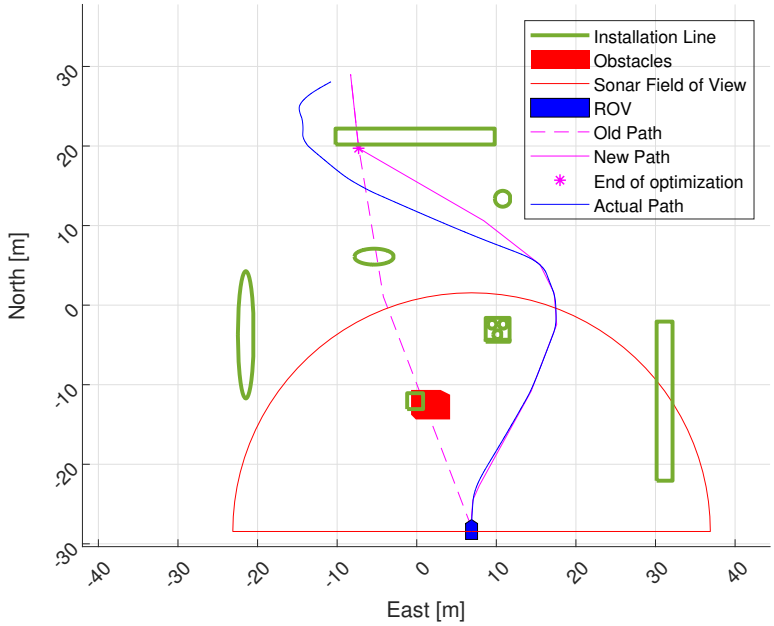



**Figure 6.9:** First path re-calculation during simulated trial of the complete collision avoidance system.

In Fig. 6.10 another path recalculation is shown. It should be noted that all the obstacles that has been inside the sonar field of view is now detected. The steep terrain at the edge of the map is also detected as an obstacle in $(N, E) \approx (5, -45)$. It should be noted that the old path is not passing trough obstacle 1, but rather the width of the vehicle, and thus the path is overlapping the detected obstacle. A new path is calculated around the obstacles.

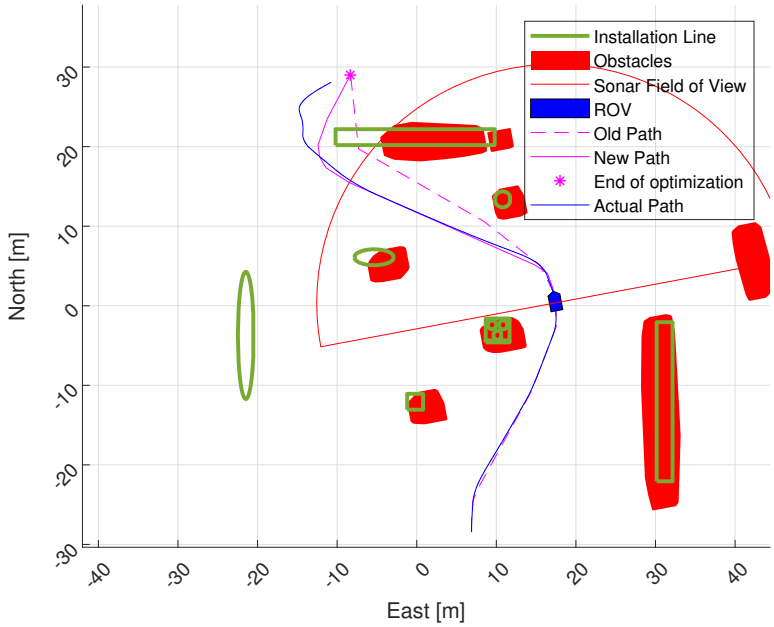



**Figure 6.10:** Second path re-calculation during simulated trial of the complete collision avoidance system.

In Fig. 6.11 the old path is to close to obstacle 1, as the detected part of the obstacle has grown when the ROV moved closer. The path is thus adjusted to get a better clearance. It should be noted that obstacle 5-7 has disappeared as they are no longer inside the occupancy grid.

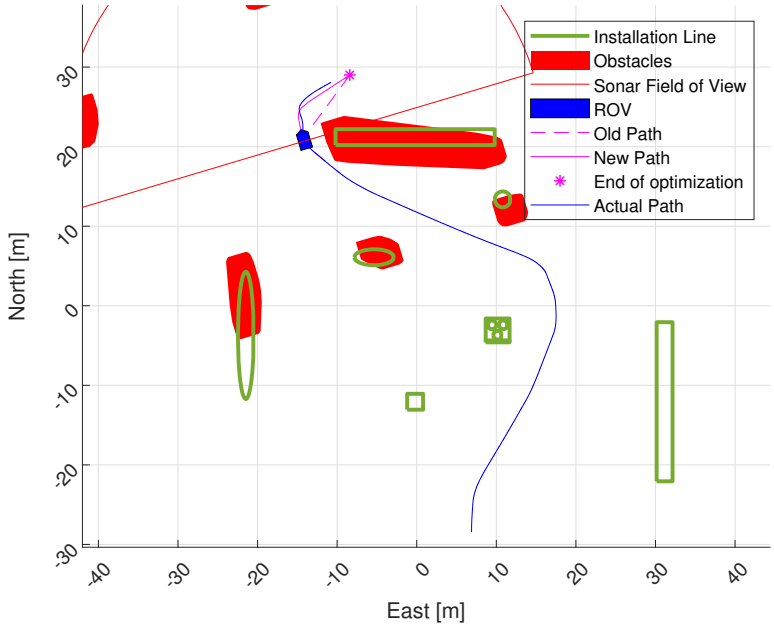



**Figure 6.11:** Last path re-calculation during simulated trial of the complete collision avoidance system.

## 6.4 Merlin Simulation

Several simulations were done in the Merlin Simulator, but no results are presented as the simulations were done to fix the communication with the IKM Autopilot Server. The ROV model in the simulator is not the same as the one used in the field experiments, but the tuning provided good initital tuning values.

Chapter 7

# Full-Scale Experiments

The complete system consisting of the object detection module, the path planning module and the guidance system was tested through real-time experiments at the Snorre B field. The raw network traffic was logged, but most of the data presented here is the result of log-files generated by the developed system. A map of the area can be seen in Fig. 7.1. Most of the experiments starts from the garage, as this is where the ROV is tethered to. The chapter is divided into three different sections, where the object detection module is tested first, then the guidance-system. The last section is a test of the complete system. Some key parameters that are equal in all of the experiments are given in Tab. 7.1.

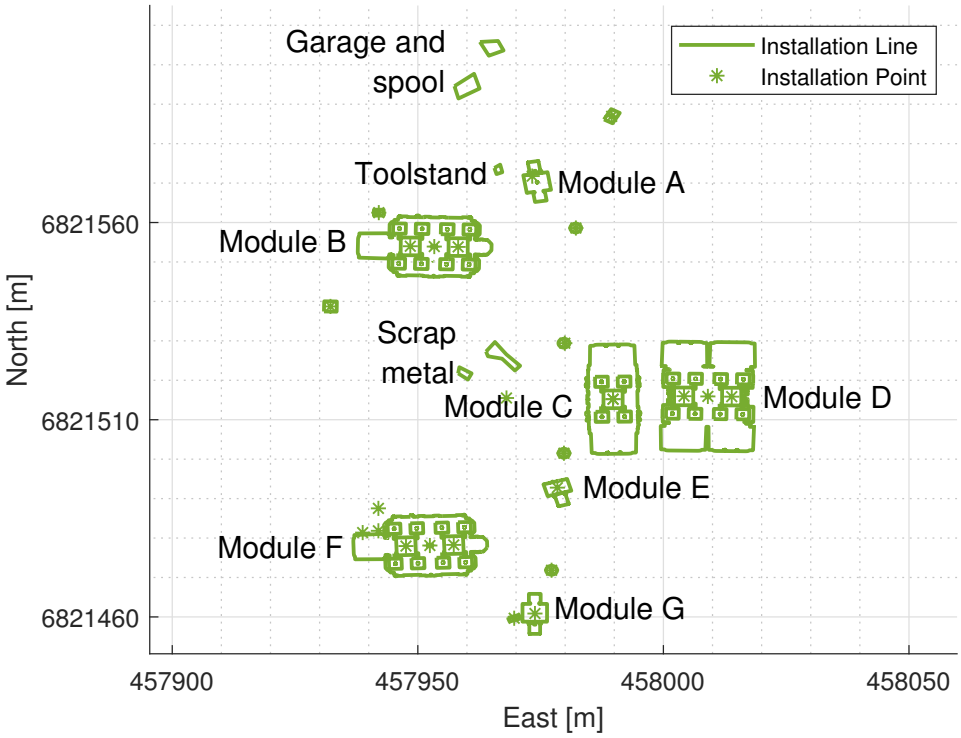



**Figure 7.1:** Map of the subsea area of the Snorre B field.

| Parameter | Value |
|---|---|
| Sonar range | 50 m |
| Sonar frequency | 650 kHz |
| Obstacle safety-margin | 2 m |
| Vehicle safety-margin | Equal to vehicle beam |
| Distance for second waypoint | 4 m |
| Co-linear threshold for removal of waypoints | 2 deg |
| Cruising velocity | 0.5m/s $\approx$ 1knot |
| LOS look-ahead time | 20 s |
| Heading update limit | 0.5 deg |
| Turn-velocity factor | $\frac{1}{32}$ |
| Initial probability | 0.5 |
| Binary probability threshold | 0.6 |
| Free probability | 0.3 |
| Occupied probability | 0.7 |
| Discrete grid size | $1601 \times 1601$ |
| Occupancy grid size | $100 \times 100$ |
| Minimum rotation of grid | 0.5 deg |
| Minimum translation of grid | 1 discrete grid cell |

**Table 7.1:** Key parameters for all field-tests.

## 7.1 Object detection

The purpose of this trial was to test the object detection capabilities of the system. The ROV was flown by a pilot from the garage, around a subsea-module and back again. The results from this flight is presented in Fig. 7.2. The detection data is logged once every 10 seconds and plotted over the map of the area. The detected obstacles are marked with red transparent, filled polygons. This results in object being detected on multiple occasions being a darker shade of red. It should be noted that the ROV sometimes appear to be flying straight through the detected obstacles. This is foremost due to the safety margin of of two meters included in the object detection procedure, as mentioned in Ch. 4.1.4, but also due to the ROV was flying above them. The ROV is flying at a mean altitude of approximately 2 meters until it reaches module C, then the mean altitude is increased to 3.5 meters for the remainder of the flight. It can be observed that all obstacles are clearly detected. It should be noted that there is some drift in the position, which can be observed by looking at the tool stand in Fig. 7.3, which has changed location between the beginning and the end. The detected obstacles appear larger than the obstacles on the map, and this is due to a safety-margin of two meters. It should also be noted that the algorithm has some problems accurately detecting the south-side of module D. This is likely due to the high altitude and proximity to the module, which causes the sonar to miss it completely.

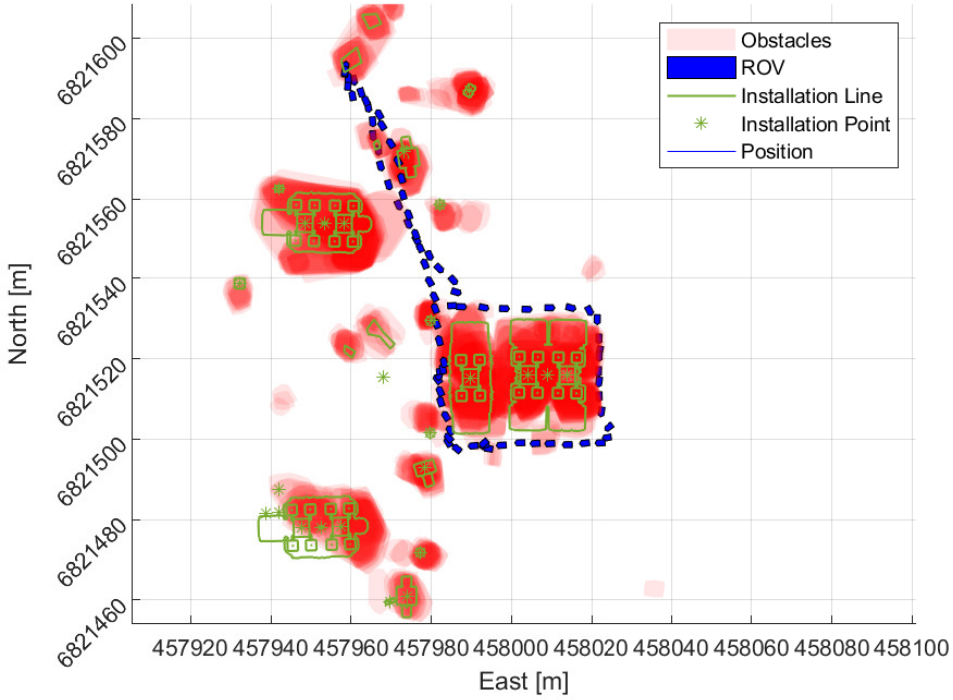



**Figure 7.2:** Detected objects and position of the ROV on a round-trip from the garage around module C and D.

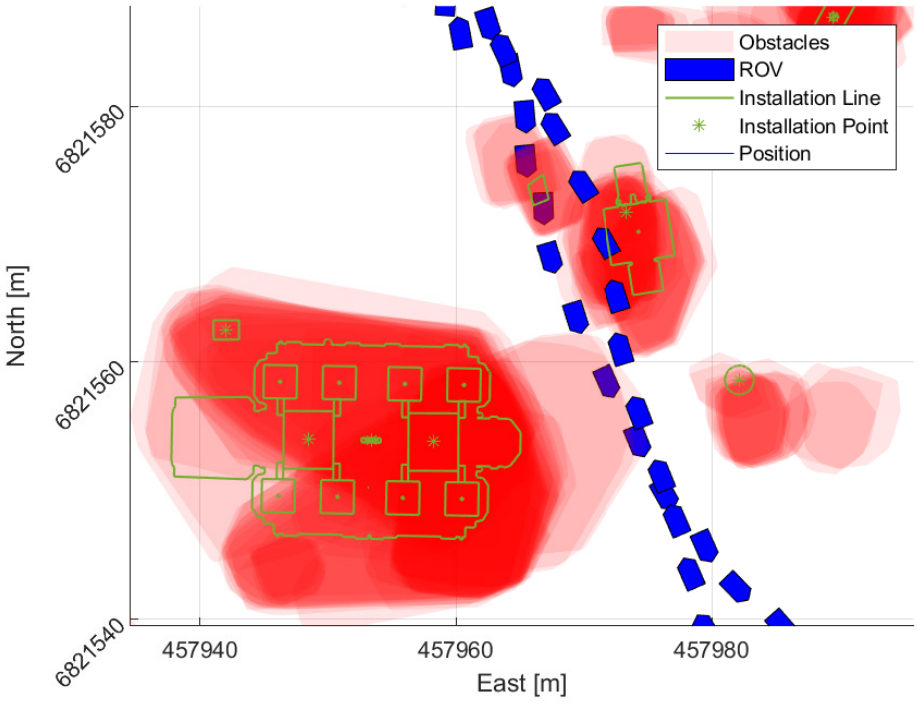



**Figure 7.3:** Closer look at the view in Fig. 7.2.

## 7.2 Guidance System

In this trial the purpose was to test the developed guidance system. The trial started between module A and B and was terminated when sufficient data was gathered, near module D. A North-east plot over the map is shown in Fig. 7.4. The result from this trial can be observed in Fig. 7.4-7.8. The numbers indicate when a new waypoint was selected. This shift happen when the ROA, indicated in orange in Fig. 7.4 is reached. In Fig. 7.5 a close-up of Fig. 7.4 can be observed. It is apparent that there are some deviations in the beginning, which can also be observed in the plot of the cross-track error in Fig. 7.8. This is due to aggressive tuning and the fact that the ROV did not start from a steady state.

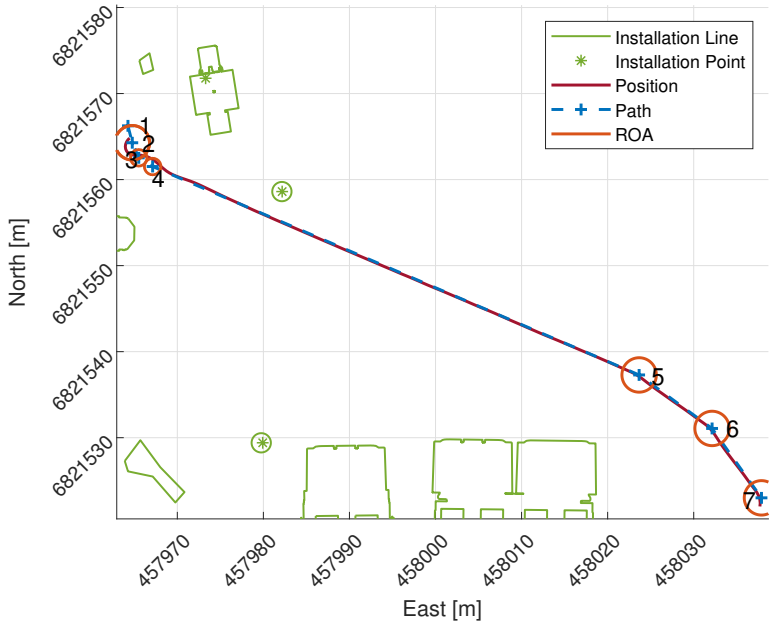



**Figure 7.4:** North-East plot of flight-path. The numbers indicate when a new waypoint is selected.

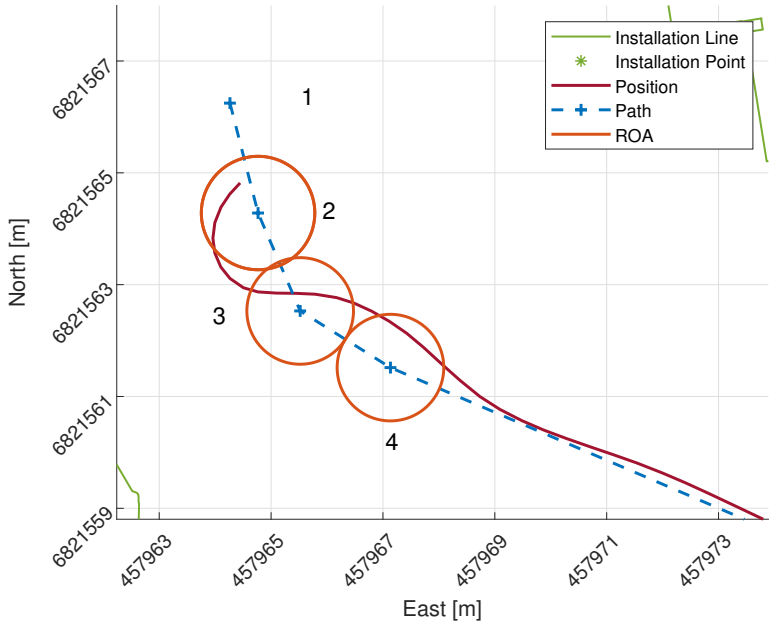



**Figure 7.5:** Close-up look at the first part of the path in Fig. 7.4.

In Fig. 7.6 a plot of the commanded heading from the LOS-controller together with the reference-signal and the estimated heading is shown. It should be noted that there is some deviations in the beginning, but as soon as the heading has had time to stabilize, the ROV is able to follow the commanded heading sufficiently good. When a new waypoint is selected there is a step in the LOS-output, but this is handled well by the reference signal.

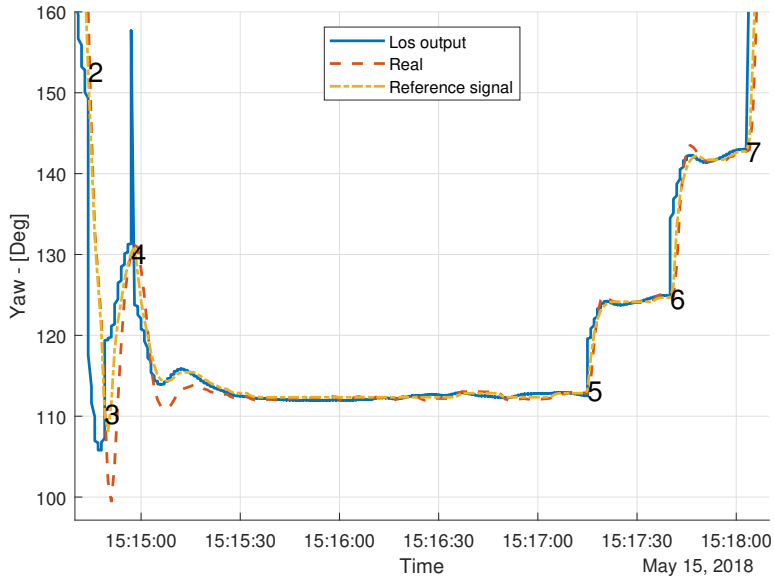



**Figure 7.6:** Plot of heading, heading reference and desired heading for the flight-path from Fig. 7.4. The numbers indicate when a new waypoint is selected.

In Fig. 7.7 the surge velocity of the ROV is shown. The last two minutes are omitted, to make the first part more visible. During the last two minutes, the velocity profile appears similar as from 15:16:30 to 15:18:00. I can be seen that between the second and third waypoint, the setpoint for the velocity is reduced to accommodate the sharper turn radius. Notice that the ROV is not able to follow the reduction in velocity, as the waypoints are to close together. After the the third waypoint, there are no more sharp turns, and thus the velocity-setpoint is adjusted back to the normal velocity again. It should be noticed that the ROV is never able to reach the commanded velocity, which is due to improper tuning of the surge-controller.

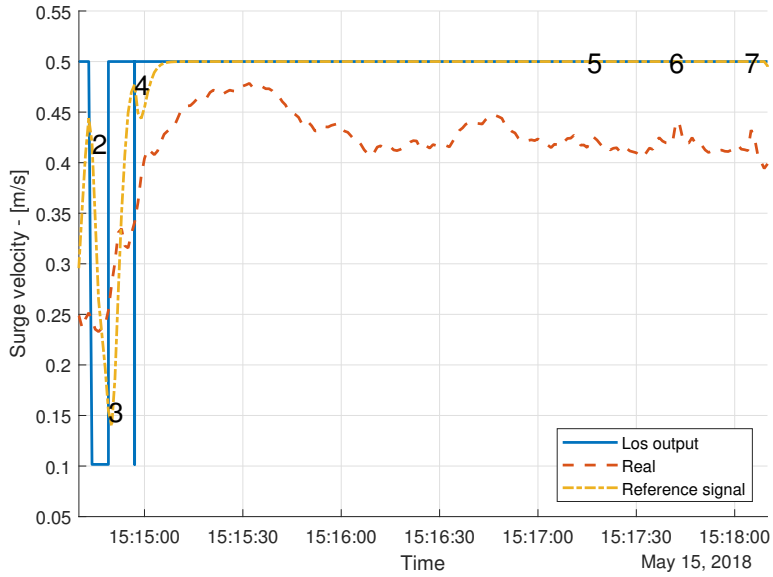



**Figure 7.7:** Plot of surge velocity, surge velocity reference and desired surge velocity for the flight-path from Fig. 7.4. The numbers indicate when a new waypoint is selected.

In Fig. 7.8 a plot of the cross-track error can be observed. The cross-track error is large in the beginning, which is due to the same reasons mentioned above. it should be noticed that for the last waypoints, there is a sudden spike in the cross-track error. This is due to $ROA = 2$m, which in causes the cross-track error to be evaluated against the next path-segment.

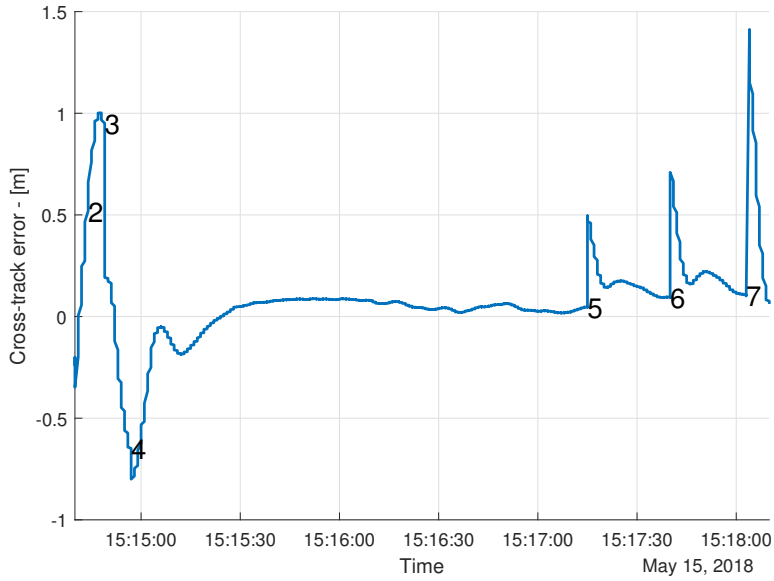



**Figure 7.8:** Plot of cross-track error for the flight-path from Fig. 7.4. The numbers indicate when a new waypoint is selected.

The results presented in this section indicate that the tuning of the controllers are not perfect, but due to the limited time available for testing the tuning was considered satisfactory.

## 7.3 Collision Avoidance

In this experiment the goal was to test the complete system. A path from the garage, going trough several obstacles towards module G was planned. As the vehicle progresses along the path, and discovers new obstacles, new paths are calculated and executed. During the experiment the obstacles and occupancy grid is logged every 10 seconds, and the detected obstacles are logged along with the path recalculation data every time a new path is calculated. An excerpt results can be observed in Fig. 7.9-7.10. All the plots of new paths in this experiment can be found in Appendix B.1 and a movie with the all the logged data can be found in Appendix B.1.

In Fig. 7.9 it can be observed that the old path is passing through one of the detected obstacles. The system then calculates a new path around the obstacle. The sharp turn between module C and D is due to an earlier recalculation of the path. It should be noted that the toolstand and module A is not detected as obstacles. Due to the altitude and close proximity several subsequent scans has showed a clear path, and thus reduced the probability below the detection threshold. These issues are further discussed in the next chapter.

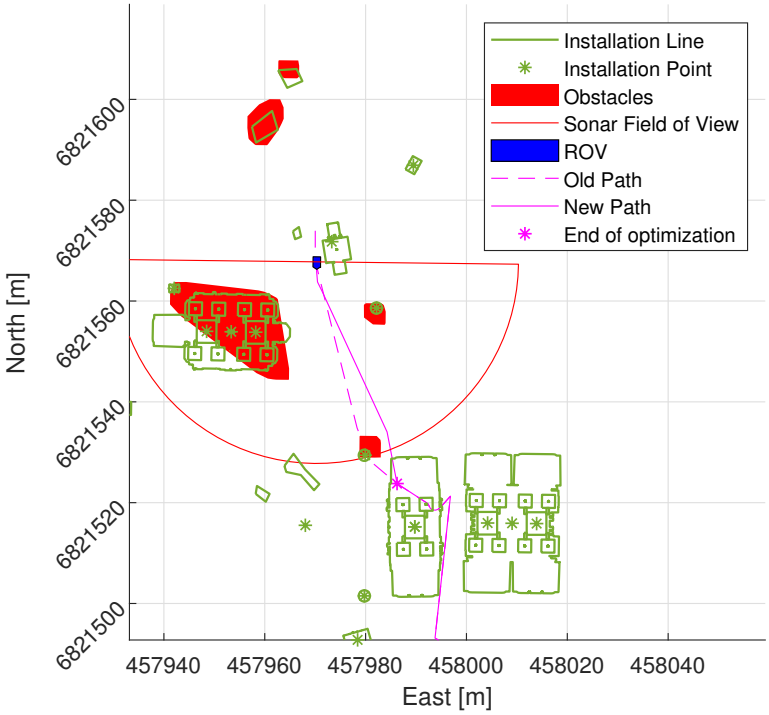



**Figure 7.9:** Excerpt from test of the collision avoidance system.

In Fig. 7.10 module C has barely come into the sonar field-of-view and the old path is once again passing straight trough the detected obstacles. The calculated path is the best possibility given the known information, but as can be seen from the map, it is passing straight through module C. The scrap-metal is not yet visible due to the current orientation of the sonar-head.

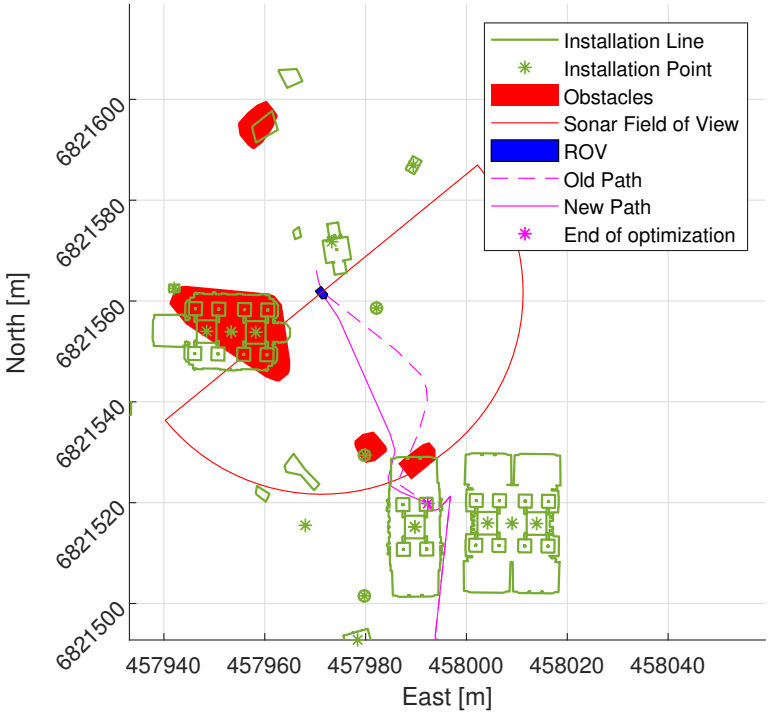



**Figure 7.10:** Excerpt from test of the collision avoidance system.

In Fig. 7.11 and 7.12 the same input data is used. Fig. 7.11 is the real-time recalculation of the path, and Fig. 7.12 is post-processing with a revised algorithm. As can be seen in Fig. 7.11 the new path takes an unnecessary long detour around module D. This is due to the nature of VD's, where the edges are as far away from each generator point as possible. In Fig. 7.12 a revised algorithm, where the midpoints between the corner points are added as generator and the revised removal of unnecessary waypoints described in Ch. 4.3.3. This algorithm has proved to calculate similar path-lengths in most cases, but significantly shorter in cases similar to this one. The new path includes a u-turn, but the path-length is shorter. Looking at the past recalculations in Appendix B.1 and the new calculations in Appendix B.2 with the revised method, it can be observed that the path on the west-side of module C would have been selected earlier, and thus removed the need for a u-turn.

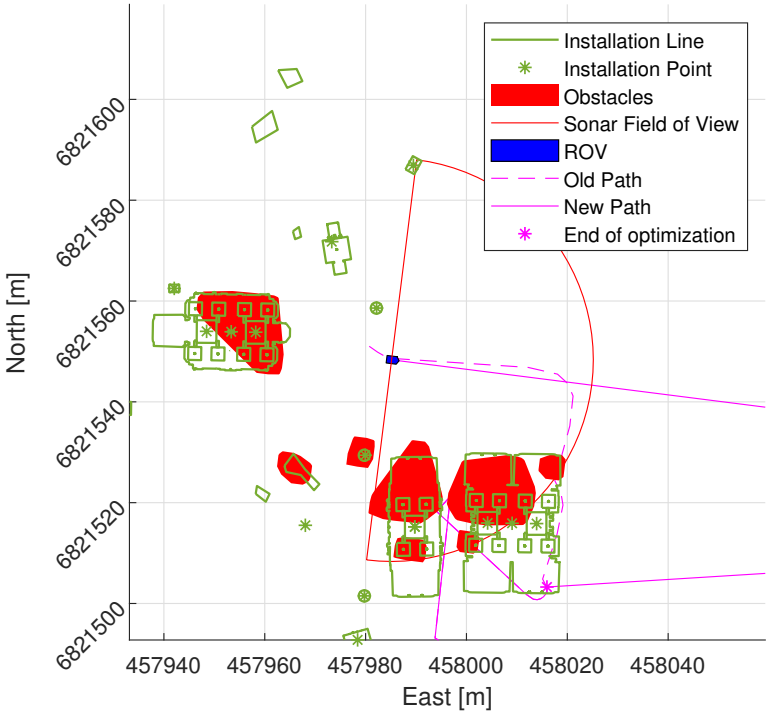



**Figure 7.11:** Excerpt from test of the collision avoidance system.

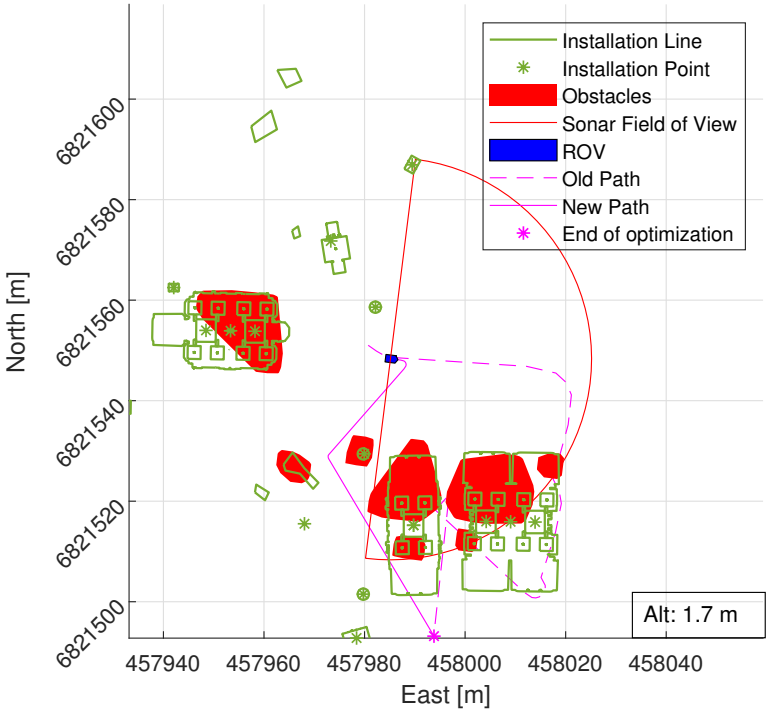



**Figure 7.12:** Excerpt from test of the collision avoidance system. This recalculation is done after the experiment, using a revised version of the recalculation method. The input data is the same as in Fig. 7.11.

Chapter 8

# Discussion

In this chapter the results presented in Ch. 6 and Ch. 7 will be discussed and compared. The chapter is divided into the same sections as the result chapters. In addition, Ch. 8.4 evaluates the runtime efficiency of the developed algorithms.

## 8.1 Object Detection

The results show that the object detection module is capable of efficiently and reliably detecting obstacles with some exceptions.

During the initial simulations, the three different methods described in Ch. 4.1.2 for identifying a hit from the sonar signal was tested. All three methods gave satisfactory results, with method two giving the most accurate results. During the first field trial, it was discovered that method 2 and 3 gave too many false positives. Method 1 requires more tuning than the other two, but due to the coupling with the sonar settings, it is easy to tune for a skilled ROV pilot.

Occupancy grids rely heavily on two assumptions, the *static world assumption* (SWA) and the *cell independence assumption* (CIA). These two assumptions are imperfect, which might cause some problems. The SWA assumes a static world, which is mostly true for a subsea production environment, but for other environments, there might be other moving and unknown objects, which violates the assumption. The CIA is violated due to the fact that each measurement gives information about a set of grid cells, not a single cell. The reliance on CIA is somewhat reduced as the dynamic inverse-sonar model is introduced as the probability of occupancy is in fact connected to the neighbouring cells. The SWA is not considered a problem in this thesis as fish and other small moving obstacles are filtered out, and larger moving obstacles normally have a known position.

A local occupancy grid has several benefits compared to global occupancy grid. A global occupancy grid has to account for the growth of the positional uncertainty, and thus the accuracy of the map will fall over time. With a local map, the obstacles will be correctly located with respect to the vehicle, which is sufficient for the purpose of collision avoidance. A global map will also require a lot of memory to retain all the information. As

an example: The discretized local grid has a resolution of $256\frac{\text{cells}}{m^2}$ when a sonar range of 50 is used, with double precision floating point which amounts to a storage space of 20.48 MB. Using the same parameters for the main subsea-area of the Snorre B field, which occupies approximately $20000m^2$. This amounts to 44 MB, while an area of $1km^2$ would take up 2 GB of memory.

On of the main problems with underwater-SLAM is the need for reliable features to use as landmarks. Reliable features can be easy to find in man-made environments, such as the subsea-production area considered in this thesis, but for other scenarios it can be difficult. One example of such a scenario is a vehicle moving between two sub-sea production areas, where unknown obstacles might recede.

During the field trials, it was discovered that the location of the sonar is of great importance. As the sonar is mounted more than a meter from the bottom of the ROV and the vertical beam-width is $40°$ when the sonar is at the high frequency, there will be a considerable blind-spot close to the ROV. This can be observed in Fig.8.1. This blind-spot causes the scans to have no echo of objects close to ROV, which in turn reduces the probability of occupation. Due to time constraints on the field trials, this problem was not addressed and further investigation is needed.

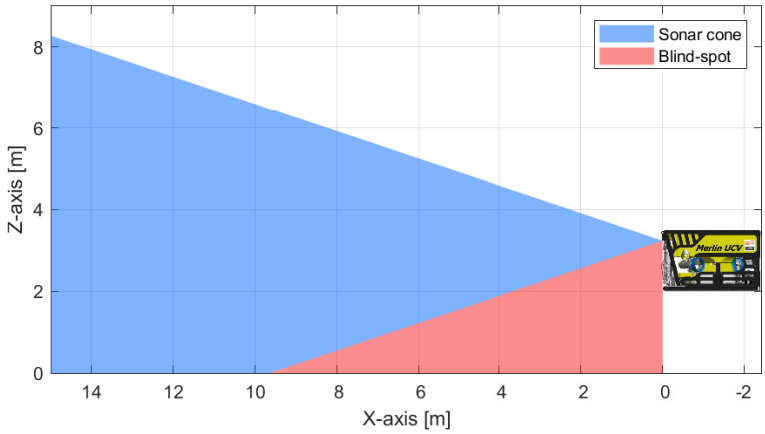



**Figure 8.1:** ROV with sonar cone and blind-spot at an altitude of two meters.

A major difference between the simulations and the field trials is the accuracy of the position and velocity measurements. During the simulations the correct position and velocity measurements where available noise free. During the field trials the position and velocity data were estimated by an INS system. This system experience drift over time and a steady-state error of approximately 90 meters was discovered. To account for this error an offset was added. To acquire this offset value, the ROV was flown to a fixed location and the difference in estimated position was compared to the position from the

map. There was also some drift during the trials, as noted in the object detection trial in Ch. 7.1. Even though this drift can be a problem for accurate localization of the ROV and subsequently the validation of the results, it is not a problem for the collision avoidance system. The ROV has an accurate compass and the velocity measurements from the DVL is accurate, which results in the change in translation and rotation of the grid being accurate.

The occupancy grid has some tunable parameters. The size of the discrete grid was chosen on the basis the of maximal number of bins generated by the sonar. During the earlier versions of the object detection module, the raw data had to be interpolated to fit the grid cells, and thus the same number of cells as bins was needed. The occupancy grid size was reduced from this, such that multiple hits has the possibility of affecting the same cell. With a smaller cell size, a single-grid cell would only be updated once during a $180^\circ$ scan, and no detection of obstacle would occur. Another important parameter is the initial, free and occupied probabilities. The initial probability was set to $p_{init} = 0.5$, since it is unknown.

The simulations were run with different values for free and occupied probabilities. It was found that $p_{free} = 0.7$ and $p_{occupied} = 0.3$ gave the best detection to false positive ratio. The threshold value used for the object detection was found to be effective at $0.6$. During the initial field experiments the hit threshold, $T = 170$ was found to bee accurate, in collaboration with the pilots.

## 8.2 Guidance System

The results in Ch. 6.2 and Ch. 7.2 indicates that the guidance system is performing sufficiently with respect to cross-track errors. The line-of-sight (LOS) guidance, with the variable lookahead distance is providing the control system with a desired heading, which makes the ROV converge to the chosen path. The cross-track errors are small during the execution of the paths, but there are some errors when a new path is started and the vehicle is not aligned with the path. The velocity dependent lookahead distance then makes the LOS-guidance to aggressive, which causes the convergence to the path to be more time consuming than necessary.

The region of acceptance (ROA) was implemented as a variable region, dependent on the proximity of the waypoints (WPs). This did not cause any problems when the ROA became small, but it should be considered for further work. A better solution would be to implement the WP-switching with respect to the along-track distance instead of a ROA, as suggested by [9, p. 265].

The surge-velocity guidance did not work as well as expected. The time allocated to reducing the velocity was not enough, and more time should have been spent on the tuning. Another problem with this algorithm was that the smoothing was done in the path planning and the velocity guidance was working against each other. The velocity guidance considers only the angle between two path-segments, which is small due to the smoothing. A possible solution to this problem is discussed in Ch. 9.2.2. It was discovered that the path following algorithms implemented on the Merlin UCV was not capable of the fast WP-switching needed by the collision avoidance system. To fix this problem the surge-velocity guidance algorithms were developed.

During the field experiments it was discovered that reference signal for both the surge-velocity and the heading was faster than the dynamics of the system. This caused some unnecessary oscillations, but due to time constraints during these experiments no additional tuning was performed.

## 8.3 Collision Avoidance

The complete collision avoidance system was tested both in simulations and in the field tests. In this section, the results from both of these trials will be discussed.

Both the simulations and the field tests showed similar results. The obstacles were detected in a timely manner and a new path was calculated.

The Voronoi diagram (VD) is an effective method for finding a set of feasible paths. From this set of feasible paths, the shortest one is found through the use of Dijkstra's algorithm. Dijkstra's algorithm provides the optimal solution from the feasible set, but not the overall optimal path. The algorithm handles multiple obstacles very well, and is able to calculate a good path through them.

The VD has several possible configurations of the generator points as discussed in Ch. 4.3.1. The runtime difference for these configurations are presented in Tab. 8.1. On the basis of this comparison the convex-contour method was chosen, even though the accuracy of the obstacle representation is lower. Due to the real time requirements of the system, a long runtime can cause problems. As stated in Ch. 8.1 the runtime comparison was done on a sample grid, but more complex grids can significantly increase the runtime. During the field trials the algorithm sometimes ran for up to 2 seconds, which with a normal AUV velocity of 4 knots would give a travel distance of approximately 4 m. To account for this a waypoint was inserted straight ahead of the vehicle as discussed in Ch. 4.5a. The insertion of this point also turned out to make it easier for the guidance system to transition between paths, as the first piece of the new path had the same orientation as the current piece of the old path. The other possibility of changing the generator points, is to use the mid-points between the corner-points, as described in Ch. 4.5a. During the trials, both possibilities were tested, and using the mid-points turned out to select negligible longer paths in most cases. In cases where the only path is around an obstacle at the outer corner of the grid, the mid-point method selected significantly shorter paths and was thus chosen as the final method.

The location of the destination point inserted into the VD has a large impact on how the final path will look. The initial algorithm used the first WP outside the occupancy grid as the destination point. This worked well in most cases. However; when the old path reentered the grid, the new path ended up in a loop. To solve this problem, a check to find the first WP after the last time the path exits the grid was implemented.

The path smoothing with the use of Fermat's spiral worked well, and a minimal curvature path was generated and subsequently discretized into the necessary number of waypoints.

A problem with the developed collision avoidance system is that a new path is only generated once the current path intersects a detected obstacle. This occasionally caused the vehicle to fly a longer path than necessary, as new information about the obstacles were gathered. A system where the path was recalculated more often was developed to solve

this problem, but the path was then changed more often than the guidance system could handle.

## 8.4 Performance

During the development of the algorithm several options for the contour approximations described in Ch. 2.2.5 were evaluated with regards to performance. A test script was constructed, where the input was an image of dilated contours. The test script then detected the contours, using the appropriate approximation. The collision avoidance algorithm was then run, until a new path was calculated. This script was run 10 times for each contour approximation. A runtime comparison for the data set used in Ch. 4.3 in can be seen in Tab. 8.1.

| Approximation | Mean time |
|---|---|
| No approximation | more than 6000 ms |
| Simple | 4344 ms |
| Teh-Chin | 2893 ms |
| Convex | 662 ms |

**Table 8.1:** Mean runtime for collision avoidance algorithm, with different contour approximations from dilated contours to a finished path. Each method was run 10 times, and a mean value was calculated. The simulation was stopped after one minute runtime. The dilated contours were generated from the same data as in Ch. 4.3.

The results indicate that the convex contour approximation is about four times as fast as the Teh-Chin approximation and 6.5 times as fast as the simple approximation. The Teh-Chin approximation has the benefit of accurately displaying convexity-defects, which could be helpful for large concave structures, such as the example shown in Fig. 8.2. A fast algorithm does, however, have more benefits than such special cases, and thus the convex contour approximations was selected for the algorithm.

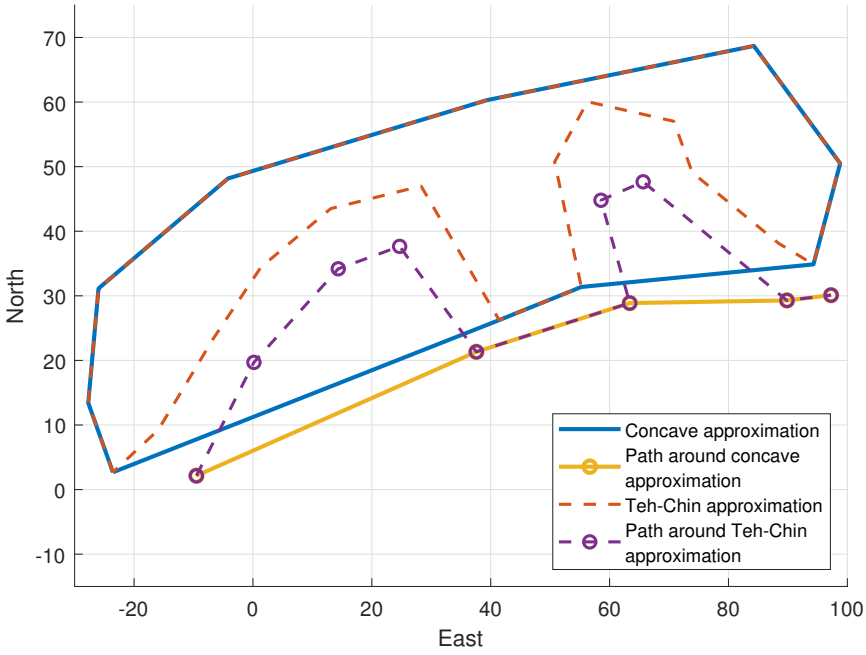



**Figure 8.2:** Example of a path following a concave structure, with a convex contour approximation and the Teh-Chin contour approximation.

When running the complete collision avoidance algorithm the CPU-utilization was below 30 %. Most of this CPU-usage is due to the graphic display of the program, which is only needed for the convenience of the user. This suggests that the algorithm is capable of running on on-board hardware on other underwater-vehicles such as AUVs.

Chapter 9

# Conclusions and Further Work

In this chapter a conclusion will be formulated based on the results in Ch. 6 and Ch. 7, and the discussion in Ch. 8. In the end recommendations for further work will be presented.

## 9.1 Conclusion

In this thesis, the problem of avoiding obstacles when following a pre-defined path was investigated. An effective approach using a single-beam sonar to populate a probabilistic vehicle-fixed occupancy grid, and subsequently detect obstacles was found. The detected obstacles were used to calculate a new collision free and smooth path adhering to the vehicles constraints. The complete collision avoidance system was simulated using a simulator called MORSE and implemented on IKM Subsea's Merlin UCV. Both the simulations and the field trials showed promising results, where the system was capable of efficiently detecting and avoiding obstacles.

The presented approach has several advantages over other methods such as SLAM-based methods. Using a vehicle-fixed grid greatly reduces the computational complexity as well as removing the reliance on landmarks, which can be hard to detect in a noisy environment. The vehicle-fixed grid also has the advantage of being independent of positional error-growth.

It was shown that a mechanically scanning single-beam sonar is sufficient as the only sensor for detecting obstacles in the vehicles planned path. The presented object detection procedure has some weaknesses, which were discussed.

The developed system for calculation of new paths is efficient, but the results are not always optimal. The usage of Voronoi diagrams and Dijkstra's algorithm always gives a collision-free path, but it might be longer than necessary. These aspects were thoroughly discussed, and suggestions for improvements will be presented in the next section.

## 9.2 Further Work

This section will present a proposal for further work, with aspects of the thesis that needs improvement and exploration.

### 9.2.1 Object Detection

With respect to the object detection system, several challenges have emerged during the work that should be addressed. The tuning of the sonar is a process that is easy for a skilled ROV pilot, but this takes a lot of practice, and thus further research should be done to develop either an algorithm that is independent of the sonar tuning, or tunes the sonar automatically. The current system uses operator controlled range and step settings, but the accuracy of the system could be increased by applying bat-inspired techniques for aiming and adjusting the sonar beam as described by Yamada et al. [42]. The main problem in testing such a system is that the ROV pilots rely heavily on the sonar for navigation, and are thus reluctant to release control over it. This could also be combined with a dynamically sized occupancy grid to fit the obstacle density.

The rotation and translation of the grid follow a deterministic manner, but the position and orientation are seldom perfectly known, and a probabilistic implementation of the rotation and translation, as done by Ganesan et al. [13] should be considered. The current translation of the grid relies on position updates, but to make the system more independent of inaccurate and unknown variables, this should be changed to use velocity measurements instead.

### 9.2.2 Guidance System

As described in Ch. 8.2, the velocity was occasionally too high trough sharp turns. A possible solution to this problem is to consider the curvature trough several waypoints, not just the first. It is also recommended to look into other guidance options, such as trajectory tracking.

### 9.2.3 Collision Avoidance

The collision avoidance system has some drawbacks that should be addressed. When an obstacle in the current path is detected it would be preferable if the guidance system is notified immediately, such that no collisions can happen during the time it takes to calculate a new path. Another drawback is that a path recalculation is only triggered by an obstacle on the current path. This can cause the vehicle to fly sub-optimal paths as new information is discovered. To solve this problem it is possible to try to recalculate the route more often, and change the route if the new route is better.

This system only considers paths in the two-dimensional space, but Voronoi diagrams can easily be extended to work in a 3D-space, but this will most likely not be possible without the use of extra sensors, such as a camera, an extra sonar or a 3D-sonar to extract information about the height of obstacles. Adding data from a camera will also improve the detection capabilities at a close range.

### 9.2.4 Simulations

The sonar model in MORSE has a few drawbacks as mentioned earlier. The simulator could be drastically improved by implementing a more realistic sonar model. To do this more research on underwater acoustics and comparison with real sonar data would be needed. The work done on accurate sonar simulators on GPUs done by Cerqueira et al. [3] is a good place to start.

# Bibliography


[1] Aakre, Ø. L., 2017. IKM Technology Autopilot Server User Manual. Tech. rep., IKM Technology.

[2] Braginsky, B., Guterman, H., 2016. Obstacle Avoidance Approaches for Autonomous Underwater Vehicle: Simulation and Experimental Results. IEEE Journal of Oceanic Engineering 41 (4), 882–892.

[3] Cerqueira, R., Trocoli, T., Neves, G., Joyeux, S., Albiez, J., Oliveira, L., 2017. A novel GPU-based sonar simulator for real-time applications. Computers and Graphics (Pergamon) 68, 66–76.

[4] Christ, R. D., Wernli Sr., R. L., 2014. Chapter 14 - Underwater Acoustics BT - The ROV Manual (Second Edition). In: The ROV Manual (Second Edition). Butterworth-Heinemann, Oxford, pp. 369–385.

[5] Christ, R. D., Wernli Sr., R. L., 2014. Chapter 15 - Sonar BT - The ROV Manual (Second Edition). In: The ROV Manual (Second Edition). Butterworth-Heinemann, Oxford, pp. 387–424.

[6] Council, N. R., 2005. Autonomous Vehicles in Support of Naval Operations. The National Academies Press, Washington, DC.

[7] Dijkstra, E., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1 (1), 269–271.

[8] Elfes, A., 1989. Using occupancy grids for mobile robot perception and navigation. Computer 22 (6), 46–57.

[9] Fossen, T. I., 2011. Handbook of Marine Craft Hydrodynamics and Motion Control. Wiley.

[10] Francois, R. E., Garrison, G. R., 1982. Sound absorption based on ocean measurements: Part I: Pure water and magnesium sulfate contributions. Journal of the Acoustical Society of America 72 (3), 896–907.

[11] Francois, R. E., Garrison, G. R., 1982. Sound absorption based on ocean measurements. Part II: Boric acid contribution and equation for total absorption. Journal of the Acoustical Society of America 72 (6), 1879–1890.

[12] Fredman, M., Tarjan, R., 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM) 34 (3), 596–615.

[13] Ganesan, V., Chitre, M., Brekke, E., 2016. Robust underwater obstacle detection and collision avoidance. Autonomous Robots 40 (7), 1165–1185.

[14] Goutsias, J., 2000. Mathematical Morphology and its Applications to Image and Signal Processing.

[15] Henriksen, E. H., Schjølberg, I., Gjersvik, T. B., 2016. UW MORSE: The underwater Modular Open Robot Simulation Engine. In: 2016 IEEE/OES Autonomous Underwater Vehicles (AUV). pp. 261–267.

[16] Hermansen, A., 2017. How we think UIDs will be used by Statoil in the future. Accessed 2018-06-25.
URL https://32zn56499nov99m251h4e9t8-wpengine.netdna-ssl.com/wp-content/uploads/2017/07/ID2017{_}Att09{_}Statoil{_}TomGlancy.pdf

[17] Hodges, R. P., 2010. Underwater Acoustics: Analysis, Design and Performance of Sonar. John Wiley and Sons, Ltd.

[18] Horton, J. W., 1959. Fundamentals of sonar. United States Naval Institute, Annapolis.

[19] Intel Corporation, 2018. Intel® Core i7-4720HQ Processor Specifications. Accessed 2018-06-23.
URL https://ark.intel.com/products/78934/Intel-Core-i7-4720HQ-Processor-6M-Cache-up-to-3{_}60-GHz

[20] Kartverk, S., 2017. Jordas rutenett. Accessed 2018-06-24.
URL https://www.kartverket.no/Kunnskap/Kart-og-kartlegging/Jordas-rutenett/

[21] Konolige, K., 1997. Improved Occupancy Grids for Map Building. Autonomous Robots 4 (4), 351–367.

[22] LAAS-CNRS, ISAE-SUPAERO, MORSE, 2016. What is MORSE? The MORSE Simulator Documentation. Accessed 2017-11-16.
URL https://www.openrobots.org/morse/doc/latest/what{_}is{_}morse.html

[23] Lekkas, A. M., 2014. Guidance and path-planning systems for autonomous vehicles. Ph.D. thesis, Trondheim.

[24] Ludvigsen, M., 2010. An ROV toolbox for optical and acoustical seabed investigations. Ph.D. thesis, Norges teknisk-naturvitenskapelige universitet, Fakultet for ingeniørvitenskap og teknologi, Institutt for marin teknikk.

[25] Milne, P. H., 1983. Underwater acoustic positioning systems. Spon, London.

[26] Newman, P., 2009. Under the Hood of the MOOS Communications API. Accessed 2017-11-17.
URL http://www.robots.ox.ac.uk/{~}pnewman/MOOSDocumentation/CommsArchitecture/latex/CommsArchitecture.pdfhttps://oceanai.mit.edu/svn/moos-ivp-aro/releases/moos-ivp-4.2.2/MOOS/Docs/CommsArchitecture/latex/CommsArchitecture.pdf

[27] NIMA, 2000. Department of Defense World Geodetic System 1984. Technical Report 8350.2, Third Edition, 175.

[28] OpenCV Development Team, 2018. The OpenCV Library. Accessed 2018-05-03.
URL https://opencv.org/about.html

[29] Otsu, N., 1979. A Threshold Selection Method from Gray-Level Histograms. Systems, Man and Cybernetics, IEEE Transactions on 9 (1), 62–66.

[30] Palomer, A., Ridao, P., Ribas, D., 2016. Multibeam 3D underwater SLAM with probabilistic registration. Sensors (Switzerland) 16 (4).

[31] Shvets, E. A., Shepelev, D. A., Nikolaev, D. P., 2016. Occupancy grid mapping with the use of a forward sonar model by gradient descent. Journal of Communications Technology and Electronics 61 (12), 1474–1480.

[32] Solari, F. J., Rozenfeld, A. F., Sebastian, V. A., Acosta, G. G., 2017. Artificial potential fields for the obstacles avoidance system of an AUV using a mechanical scanning sonar. In: 2016 3rd IEEE/OES South American International Symposium on Oceanic Engineering, SAISOE 2016. Institute of Electrical and Electronics Engineers Inc., Grupo INTELYMEC, Centro de Investigaciones en Física e Ingeniería Del Centro (CIFICEN), Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Facultad de Ingeniería, Universidad Nacional Del Centro de la Provincia de Buenos Aires (UNCPBA),.

[33] Statoil, 2018. Snorre. Accessed 2018-06-08.
URL https://www.equinor.com/no/hva-vi-gjoer/norwegian-continental-shelf-platforms/snorre.html

[34] Suzuki, S., Be, K., 1985. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing 30 (1), 32–46.

[35] Teh, C. H., Chin, R. T., 1989. On the detection of dominant points on digital curves. IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (8), 859–872.

[36] Thorp, W. H., 1967. Analytic Description of the Lowâ Frequency Attenuation Coefficient. Journal of the Acoustical Society of America 42 (1), 270.

[37] Thrun, S., 2003. Learning Occupancy Grid Maps with Forward Sensor Models. Autonomous Robots 15 (2), 111–127.

[38] Tritech, 2017. Tritech Knowledge Base. Accessed 2017-11-20.
URL http://www.tritech.co.uk/uploaded{_}files/WhatareCHIRPSonars.pdf

[39] Tritech International Limited, 2009. Software Notes for controlling and operating SeaKing / SeaPrince RS-232 Profiler Heads ( or Sonar Heads in Profiling Mode ). Tech. Rep. May 2005.

[40] Weisstein, E. W., n.d. Affine Transformation. Accessed 2018-06-23.
URL http://mathworld.wolfram.com/AffineTransformation.html

[41] Whitaker, J., 2018. Package pyproj. Accessed 2018-06-22.
URL http://jswhit.github.io/pyproj/

[42] Yamada, Y., Ito, K., Oka, A., Tateiwa, S., Ohta, T., Kobayashi, R., Hiryu, S., Watanabe, Y., 2015. Obstacle-avoidance navigation by an autonomous vehicle inspired by a bat biosonar strategy.

[43] Yen, J., 1972. Finding the lengths of all shortest paths in n-node nonnegative-distance complete networks using 1 2 n 3 additions and n 3 comparisons. Journal of the ACM (JACM) 19 (3), 423–424.

[44] Zhou, M., Bachmayer, R., Deyoung, B., 2016. Mapping for control in an underwater environment using a dynamic inverse-sonar model. In: 2016 OCEANS MTS/IEEE Monterey, OCE 2016. Institute of Electrical and Electronics Engineers Inc., Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL, Canada, pp. 1–8.

[45] Zhou, S., 2014. OFDM for underwater acoustic communications. Wiley.

Appendix A

# Abstract for Paper Contribution to *2018 IEEE OES Autonomous Underwater Vehicle Symposium*

# Navigation and collision avoidance of underwater vehicles using sonar data

Ørjan Grefstad
*Dept of Marine Technology*
*NTNU*
Trondheim, Norway
orjangr@stud.ntnu.no

Ingrid Schjølberg
*Dept of Marine Technology*
*NTNU*
Trondheim, Norway
Ingrid.Schjolberg@ntnu.no

***Abstract*— Collision avoidance is one of the main challenges in the field of autonomous underwater vehicles (AUV). In this paper a method for detecting obstacles, using a single-beam mechanically scanning sonar, including planning of an optimal path around the obstacles is proposed. The obstacle detection is archived with an inverse-sonar model updating a vehicle-fixed occupancy grid. A new and obstacle-free path is planned using Voronoi diagrams and Dijkstra's algorithm. The path is smoothed using Fermat's spiral and a LOS-guidance system with a time-varying lookahead-distance as guidance. The method is implemented and a full-scale test is performed from IKM's onshore control room on a remotely operated vehicle (ROV) operating at Statoil's Snorre B oil field. The technology is applicable to ROVs and AUVs in underwater operations.**

***Keywords—obstacle detection, collision avoidance, path planning, single-beam sonar***

## I. Introduction

Underwater vehicles and specifically remotely operated vehicles (ROVs) are commonly used for Inspection, Maintenance and Repair (IMR) missions in the oil and gas industry. This is a cost-driven industry and advances in automation is key-factor to reduce the mission expenses. One of the main difficulties during automated missions is the risk of collision. The collision avoidance challenge is often solved using multi-beam sonars, in a Simultaneous Localization And Mapping approach [1] or with the image recognition based techniques [2]. By using a single-beam sonar the costs can be significantly reduced. The object detection challenge is thougher with single-beam sonars , but can be solved with occupancy grids [3] or with a potential field method [4].

In this paper, occupancy grids are populated using the dynamic inverse-sonar model developed in [5]. The detected obstacles are then used as input to an online re-planning algorithm. This algorithm is motived by the work presented in [6]

## II. System Description

The system developed in this paper is tested on IKM's Merlin UCV, which is a work-class ROV permanently situated at Statoil's Snorre B oil field. The ROV is shown in Fig. 1. However, the method is applicable also to autonomous underwater vehicles (AUVs). The ROV has a Doppler-velocity log aided INS system, which together with a hydro-acoustic positioning system, situated at the rig provides accurate attitude and position information. This is a useful tool in the verification of the developed system.

The ROV is equipped with a Tritech Super SeaKing sonar. The mounting position of the sonar is highlighted in Fig. 1**Feil! Fant ikke referansekilden.**. The sonar is a single beam, mechanically scanning sonar, which utilizes CHIRP technology with frequencies centred at either 325 kHz or 675 kHz.

The ROV is operated from IKM's onshore control-room at Bryne, Norway, which makes it accessible for testing of new algorithms. The communication with the ROV's control system and the sonar is performed with UDP-messages following a binary protocol. The position updates are received as NMEA-messages.

METHODS AND IMPLEMENTATION

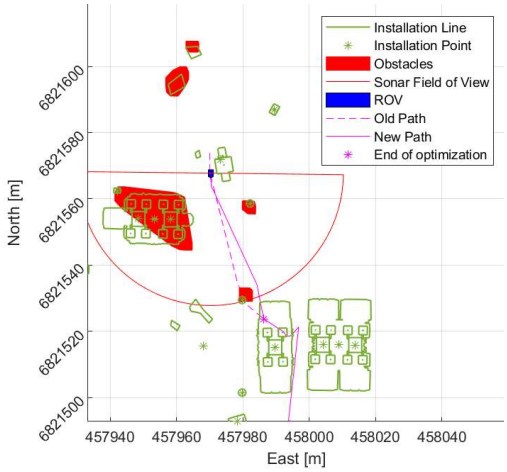



**Fig. 2: Recalculation of the path during collision**

The collision avoidance system is divided into three modules. First, the obstacles are detected in the object detection module. The detected obstacles are then passed on to the path planning module, which checks for potential collision threats, and calculates a new path if needed. The last module is the guidance system.

## III. RESULTS

The system is first tested for object detection capabilities, then the complete system is tested with pre-planned paths, going straight through obstacles.

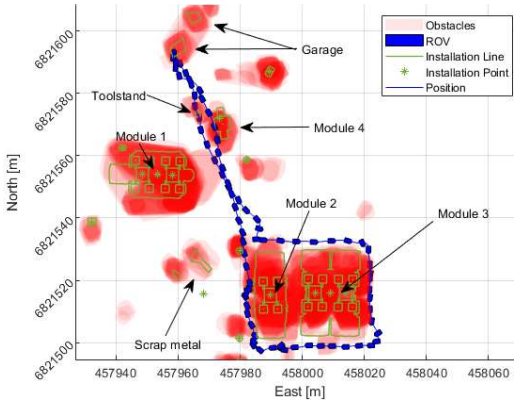



**Fig. 3: Obstacle detection test. Some obstacles on the map are labeled**

### *A. Obstacle detection*

In the first test the ROV was flown by a pilot, from the garage, around a subsea-module and back again. The results from this test can be observed in Fig. 3. A snapshot of the detected obstacles is taken every 10 seconds and then plotted in the same figure. The altitude of the ROV is controlled by the pilot and thus it appears to be flying straight through some obstacles, such as the toolstand and Module 4. In these cases, the ROV was flying above them. The ROV is flying at a mean altitude of approximately 2 meters until it reaches Module 2, then the mean altitude is increased to 3.5 meters for the remainder of the flight. It can be observed that all obstacles are clearly detected. It should be noted that there is some drift in the position, which can be observed by looking at the tool stand, which has changed location between the beginning and the end. The detected obstacles appear larger than the obstacles on the map, and this is due to a safety-margin of two meters. It should also be noted that the algorithm has some problems accurately detecting the south-side of Module 3. This is likely due to the high altitude and proximity to the module, which causes the sonar to miss it completely.

### *B. Collision Avoidance*

Several tests on collision avoidance are performed. All of the tests start close to the garage, with a preplanned path. The ROV was never able to reach the destination point due to constraints with the tether and ongoing operations.

Results from the first collision avoidance test can be observed in Fig. 2. It should be noted that most of the obstacles in the sonar's field of view are detected. The only undetected obstacle is Module 4 (see Fig. 3), which is due to proximity and altitude. The system is calculating a short deviation from the planned route, and is able to avoid the obstacle before the new path rejoins the old one. Right after the point where the optimization stops, between Module 2 and 3, there is a sharp bend in the path. This is the result of an earlier path recalculation. Since it is outside the optimized region, it is not smoothed away before it is closer to the optimized region. Fig. 5 shows a later time in the same test. The planned path on the east-side of Module 2 and 3 is longer than the one on the west-side, and thus the shorter one is selected. There is also an opening between Module 2 and 3, but due to the width of the ROV, this path is not feasible. It should be noted that the small obstacle south-east of Module 4 has disappeared from the grid, as the ROV flew at a high altitude close to it, and therefore several scans showed no obstacles.

The second test is made a few days later, with a different configuration of the sonar. The sonar configuration is decided by the pilot, and has implications for the effectiveness of the algorithm. When looking at **Feil! Fant ikke referansekilden.**Fig. 6 and Fig. 4 it can be observed that the obstacles appear smaller. This is due to the change in the sonar configuration causing less distinguishable return echoes. The system still manages to calculate a route around the obstacles.

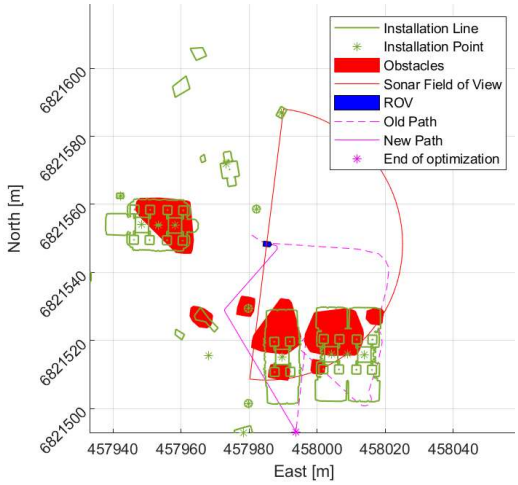



**Fig. 5: Recalculation of the path during collision avoidance test 1.**

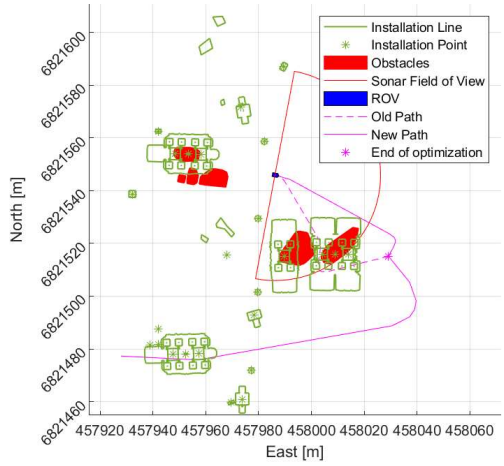



**Fig. 4: Recalculation of the path during collision avoidance test 2**

## IV. DISCUSSION AND CONCLUSIONS

This paper has presented an effective method for detecting obstacles using a single-beam sonar, as well as an effective way of calculating a new obstacle-free path. A vehicle-fixed local occupancy grid has several advantages over a global

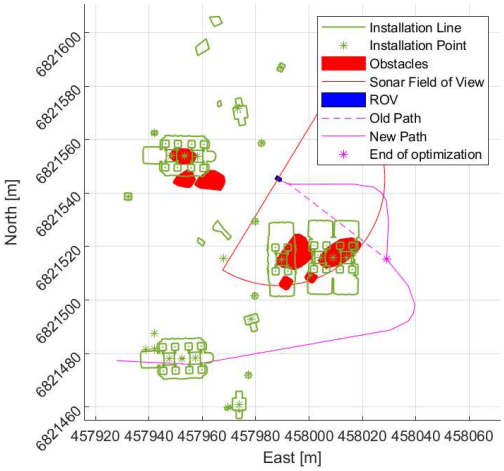



**Fig. 6: Recalculation of the path during collision avoidance test 2**

map. The complexity is reduced, and thus calculation time is significantly less. The major advantage is that the grid can be completely decoupled from global positions. The changes in position can then come from only a doppler-velocity log and a compass.

The full-scale test showed that the system is capable of detecting the obstacles in its path in an effective manner. A new and obstacle-free path is calculated and executed. The full-scale test showed that a mechanically-scanning single-beam sonar is adequate for detecting and avoiding obstacles.

The dependence on the sonar configuration is a problem that should be addressed, either through making the detection algorithm independent of the sonar configuration, or by making an algorithm that can automatically tune the sonar.

This system only considers paths in the two-dimensional space, but Voronoi diagrams can easily be extended to work in a 3D-space, but this will most likely not be possible without the use of extra sensors, such as a camera, an extra sonar or a 3D-sonar to extract information about the height of obstacles. Adding data from a camera will also improve the detection capabilities at a close range. The methods are applicable to any underwater vehicle equipped with a single beam sonar.

## REFERENCES

[1] A. Palomer, P. Ridao, and D. Ribas, "Multibeam 3D underwater SLAM with probabilistic registration," *Sensors (Switzerland)*, vol. 16, no. 4, 2016.

[2] B. Braginsky and H. Guterman, "Obstacle Avoidance Approaches for Autonomous Underwater Vehicle: Simulation and Experimental Results," *IEEE J. Ocean. Eng.*, vol. 41, no. 4, pp. 882–892, 2016.

[3] V. Ganesan, M. Chitre, and E. Brekke, "Robust underwater obstacle detection and collision avoidance," *Auton. Robots*, vol. 40, no. 7, pp. 1165–1185, 2016.

[4] F. J. Solari, A. F. Rozenfeld, V. A. Sebastian, and G. G. Acosta, "Artificial potential fields for the obstacles avoidance system of an AUV using a mechanical scanning sonar," in *3rd IEEE/OES South American International Symposium on Oceanic Engineering, SAISOE 2016*, 2017.

[5] M. Zhou, R. Bachmayer, and B. Deyoung, "Mapping for control in an underwater environment using a dynamic inverse-sonar model," in *2016 OCEANS MTS/IEEE Monterey, OCE 2016*, 2016, pp. 1–8.

[6] A. M. Lekkas, *Guidance and Path-Planning Systems for Autonomous Vehicles*, no. April. 2014.

[7] S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Auton. Robots*, vol. 15, no. 2, pp. 111–127, 2003.

[8] M. Candeloro, A. M. Lekkas, A. J. Sørensen, and T. I. Fossen, "Continuous Curvature Path Planning using Voronoi diagrams and Fermat's spirals," *IFAC Proc. Vol.*, vol. 46, no. 33, pp. 132–137, 2013.

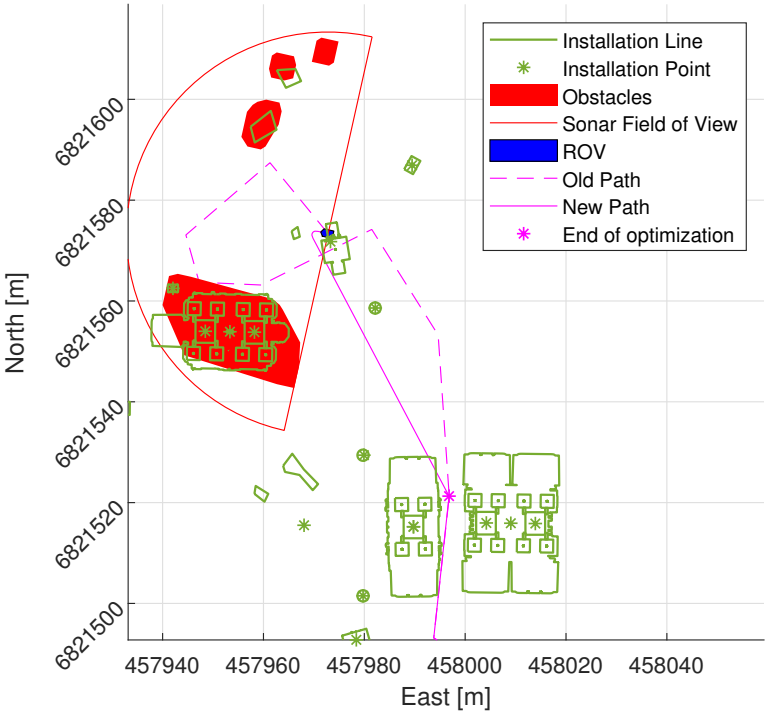
[9] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. 2011.

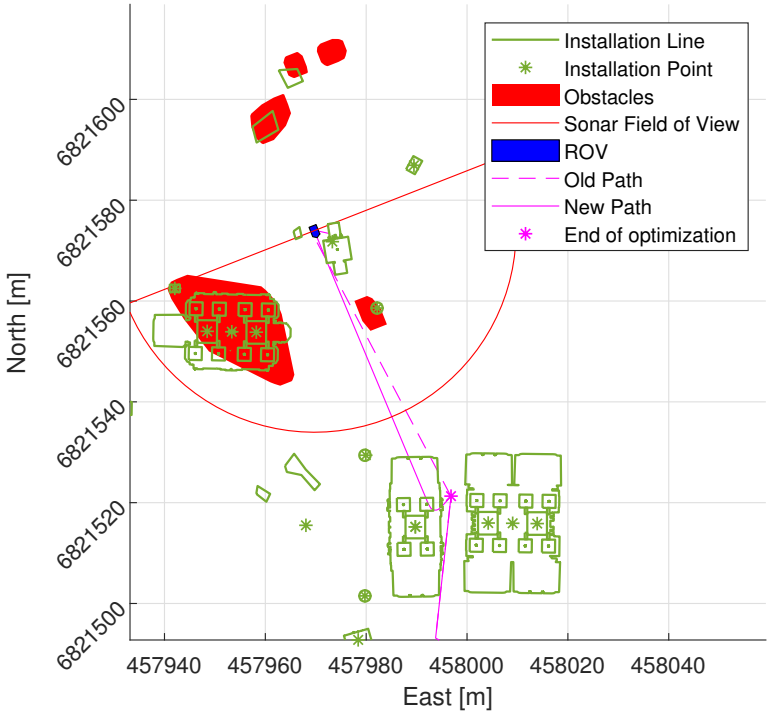
Appendix B

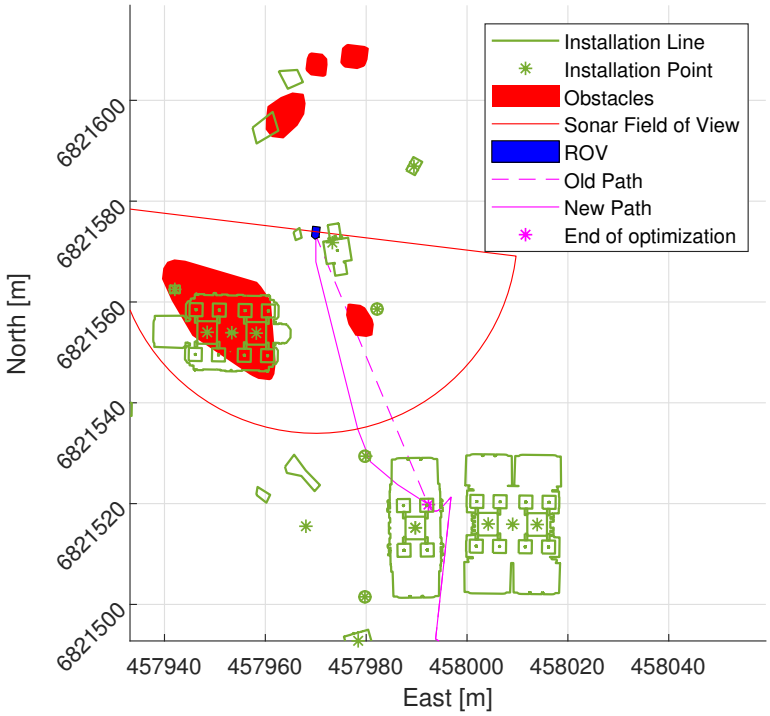
# Complete results for a selection of field trials

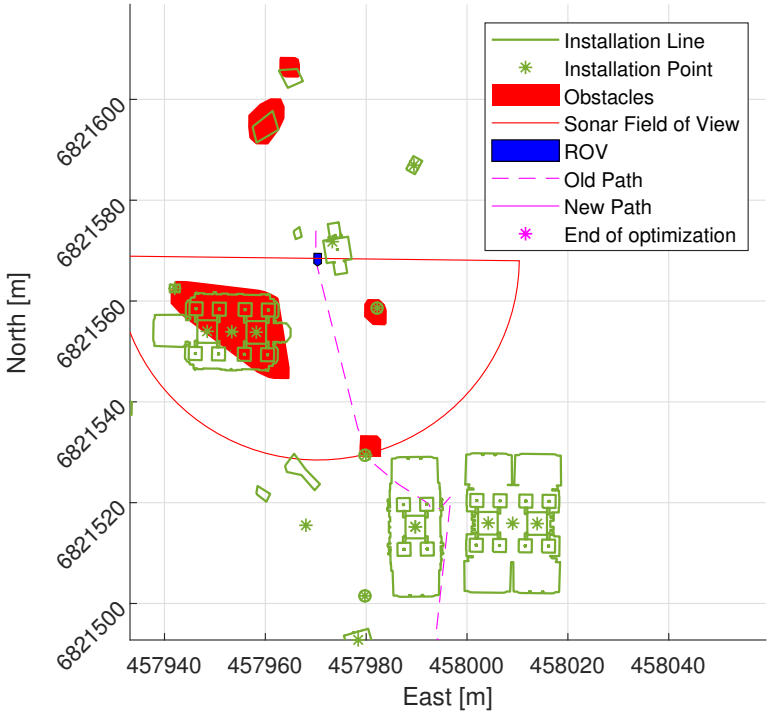
## B.1 Complete results for collision avoidance test at May 15th, 15:39

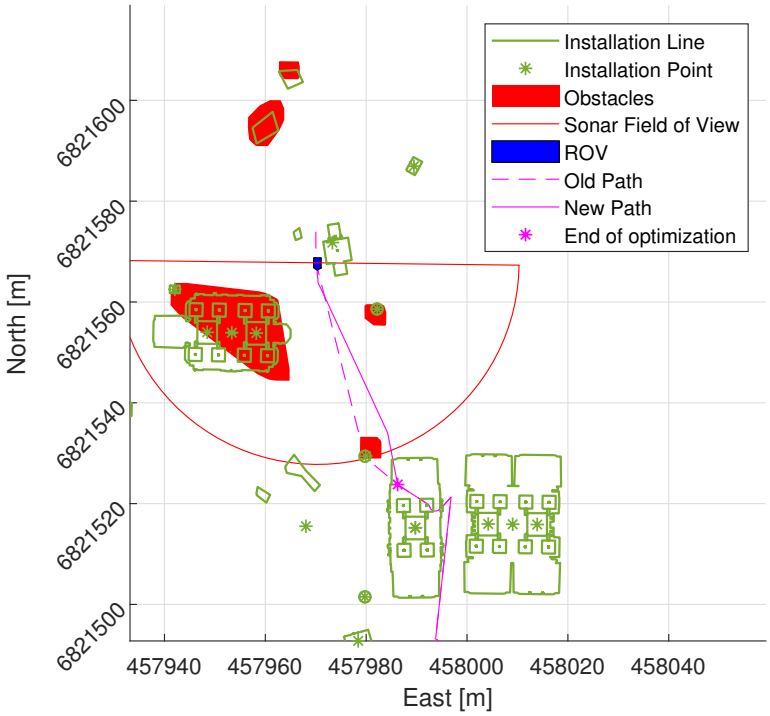
The collision avoidance results in this section are presented in chronological order.

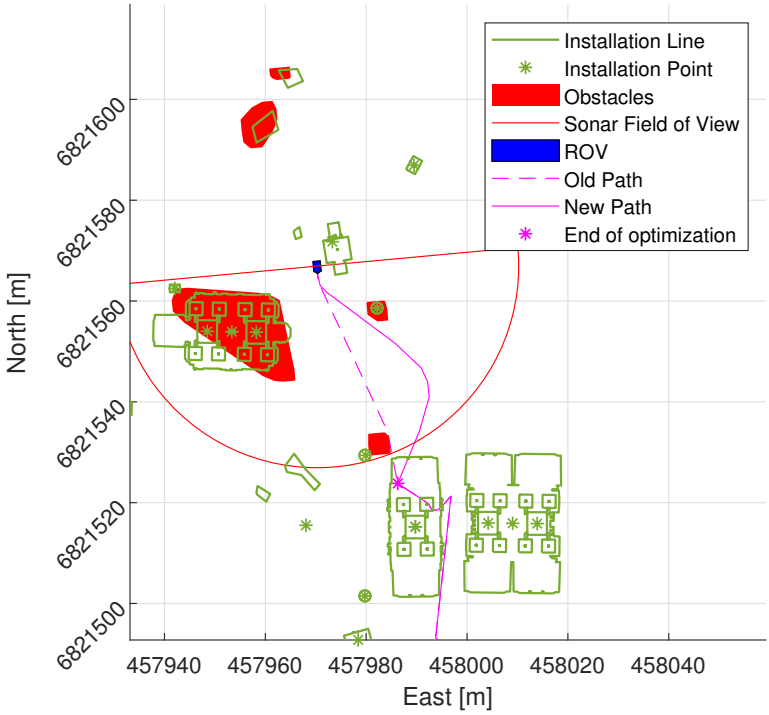


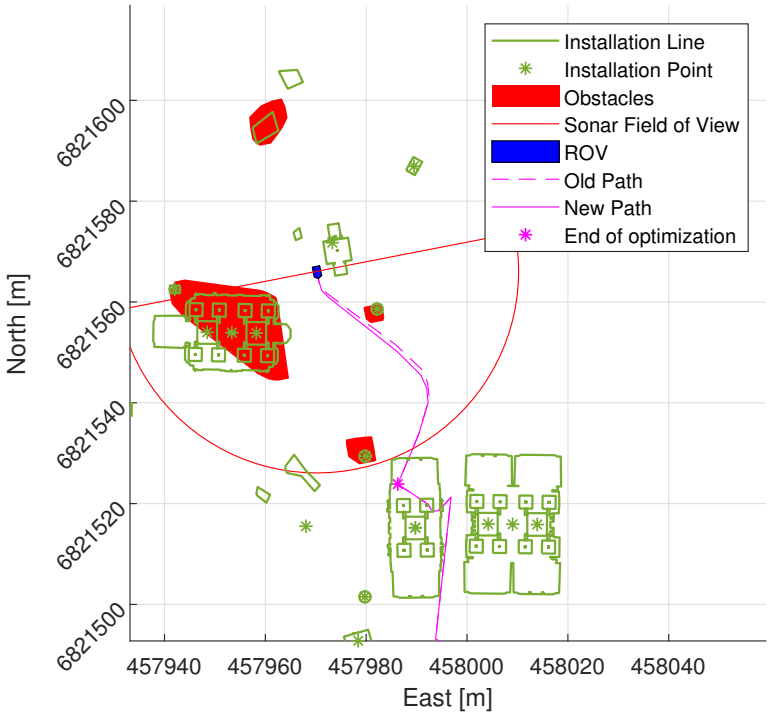

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

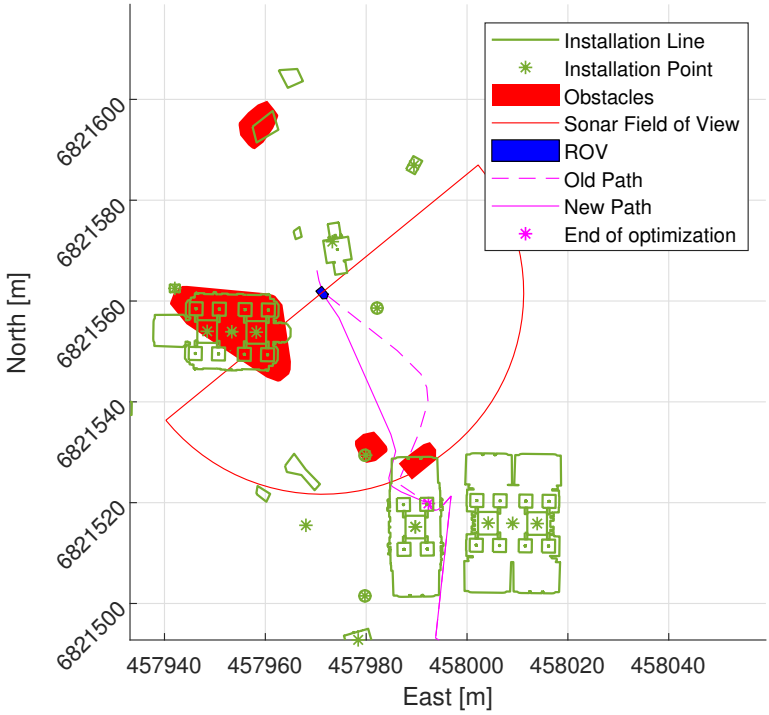

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

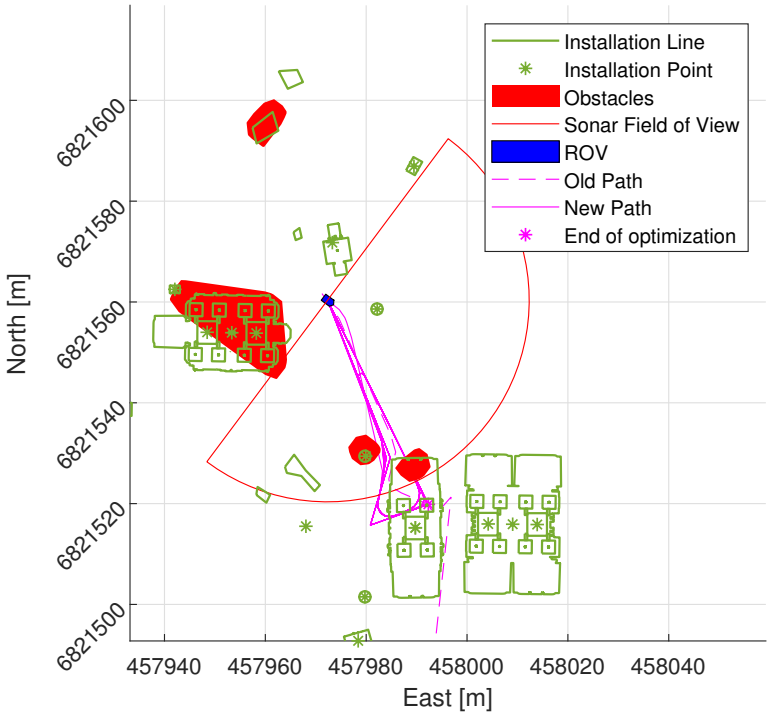

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

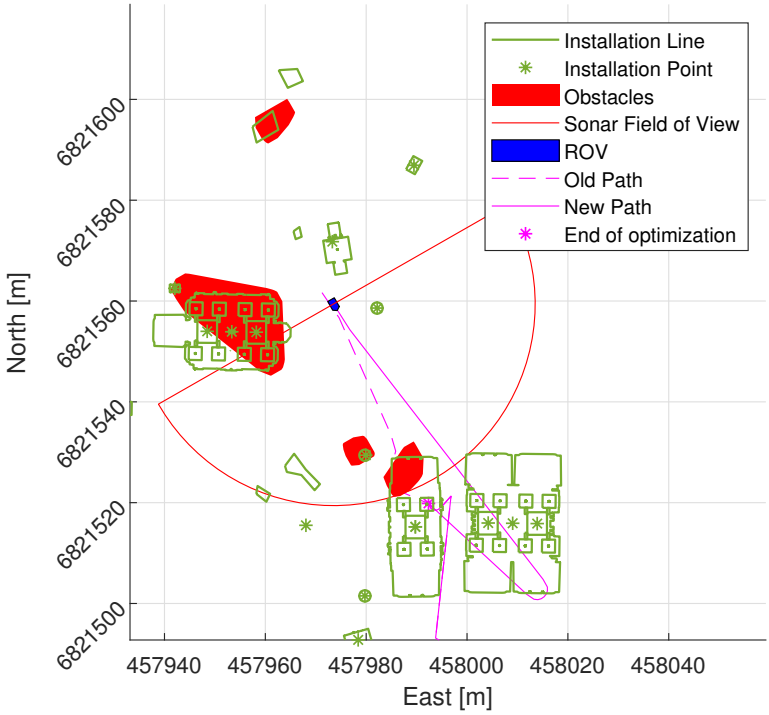

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

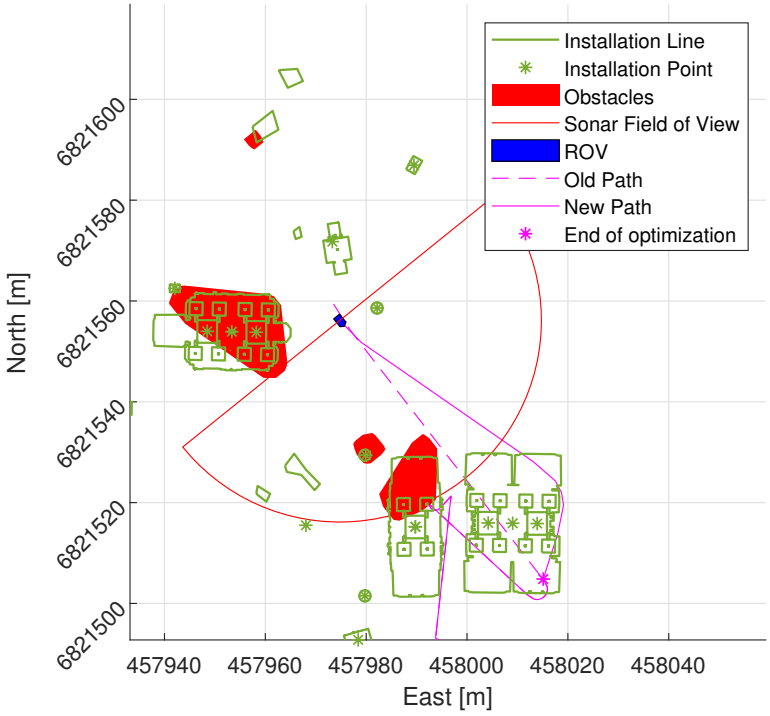

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

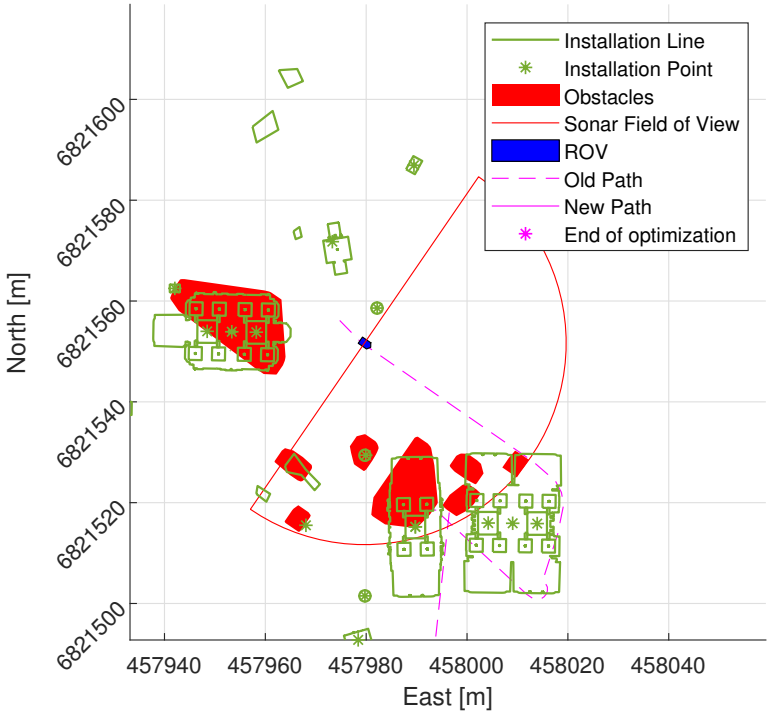

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

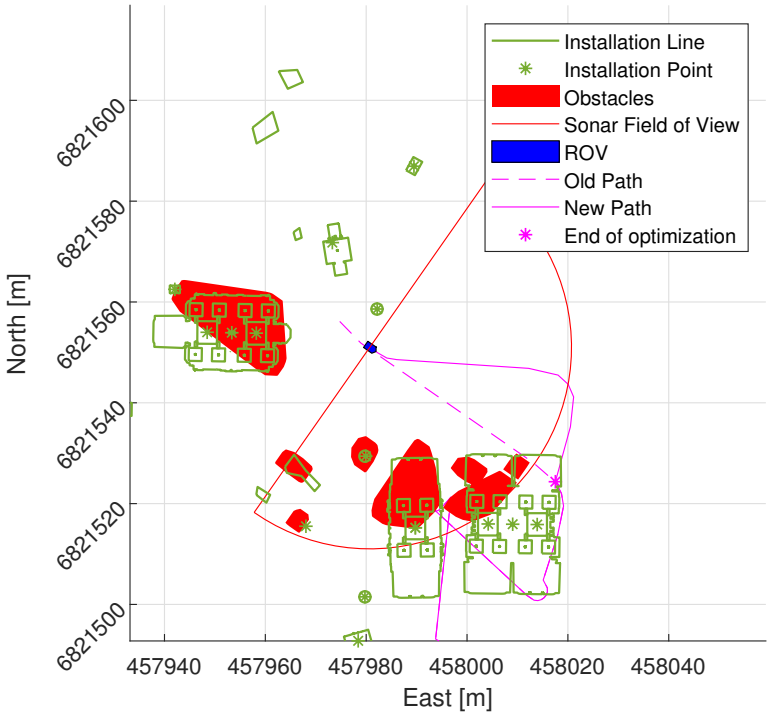

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

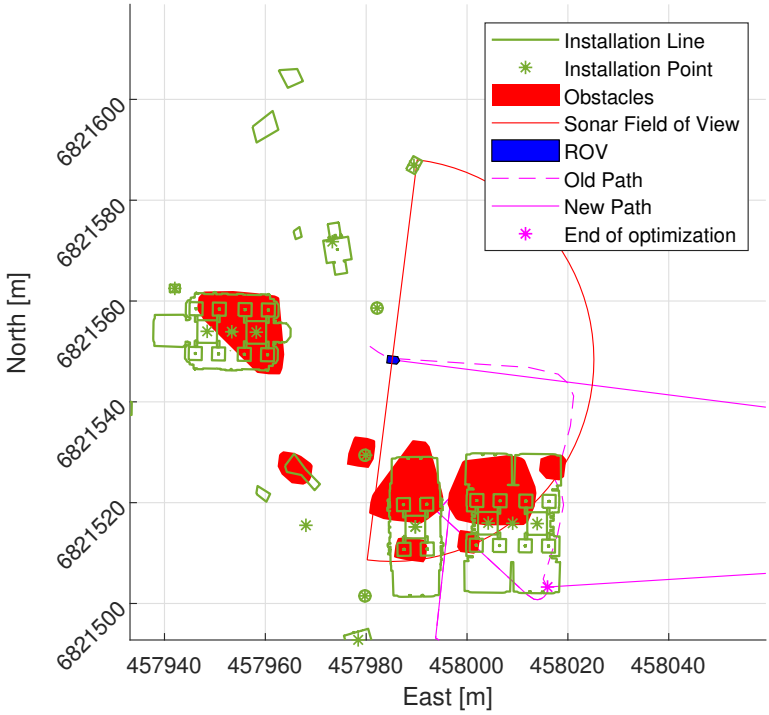

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

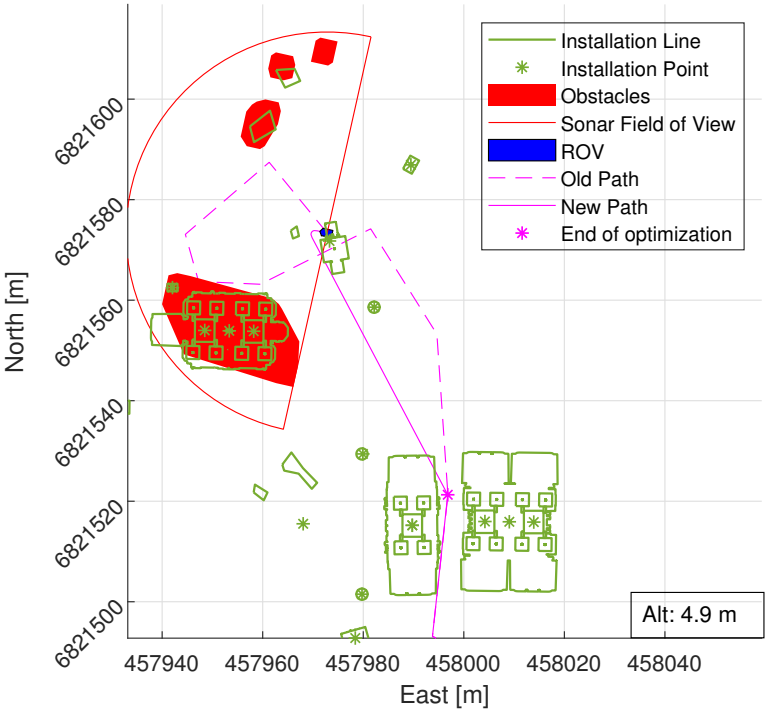

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

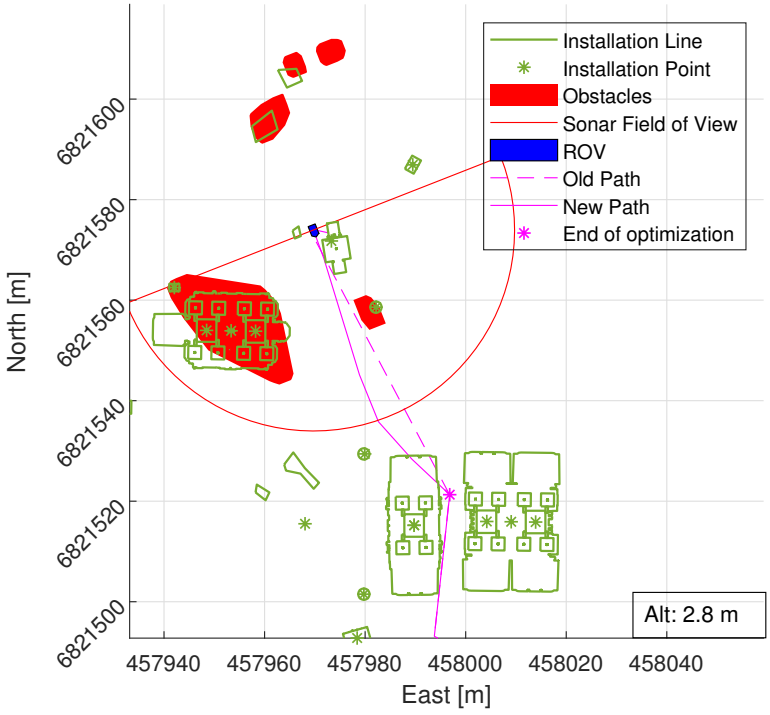

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

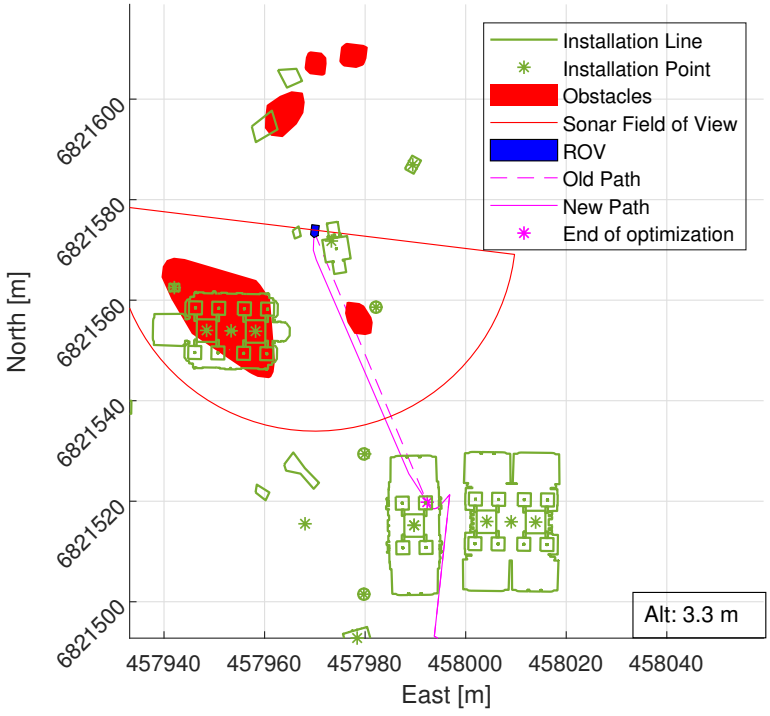

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

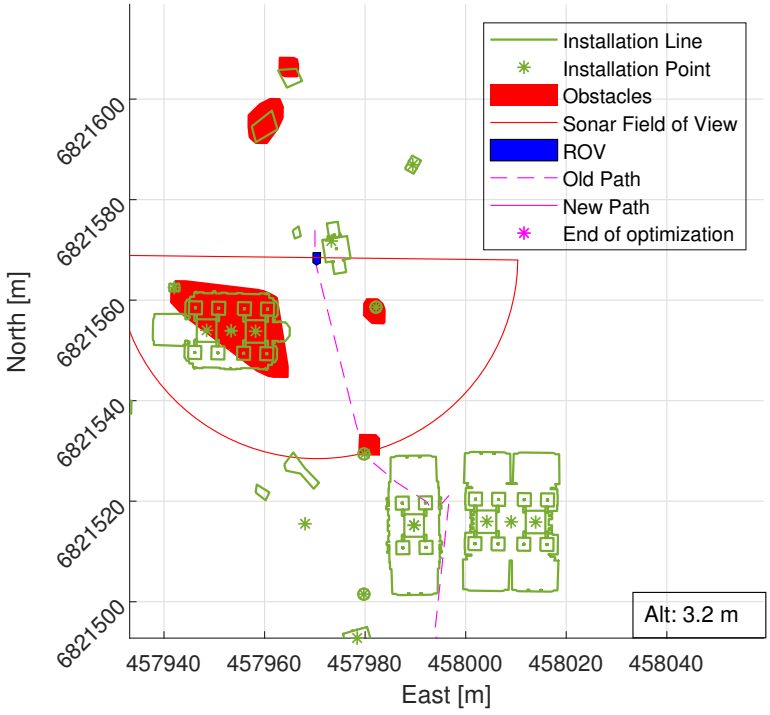
## B.2 Revised results for collision avoidance test at May 15th, 15:39

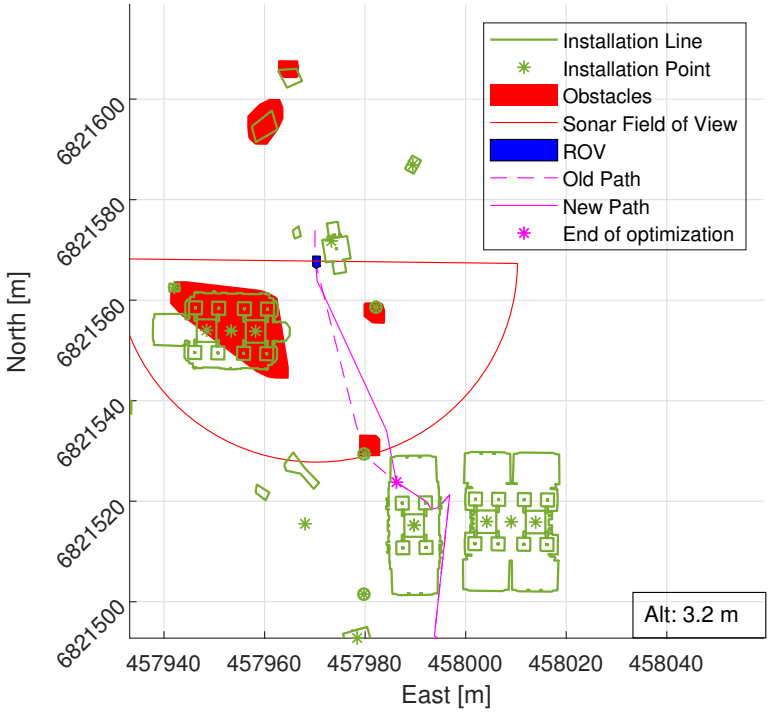
The revised collision avoidance results in this section are presented in chronological order.

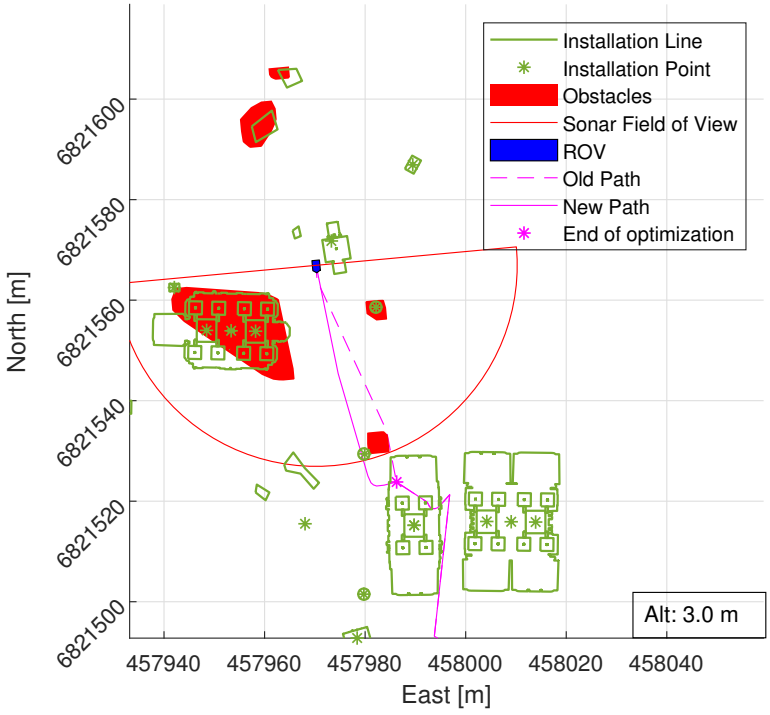


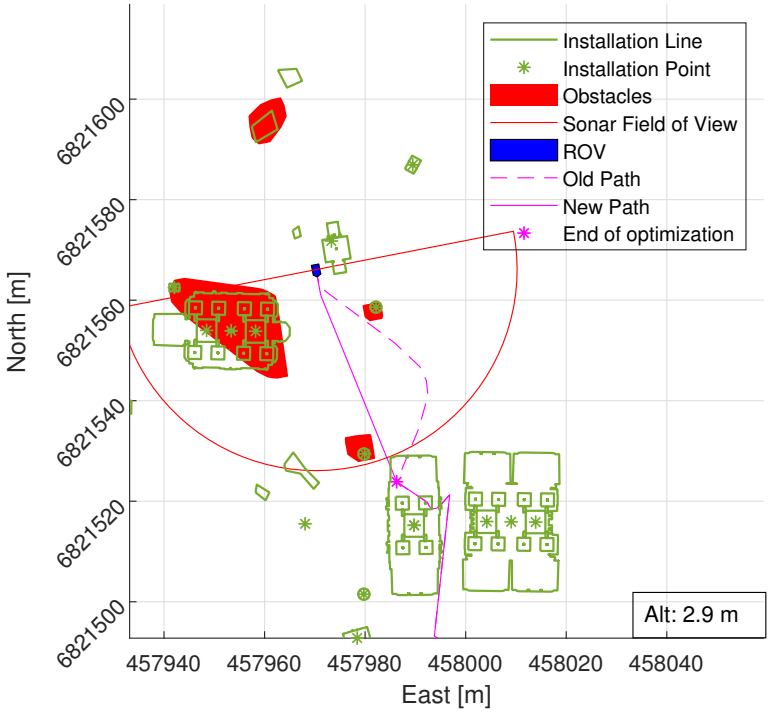

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.8 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

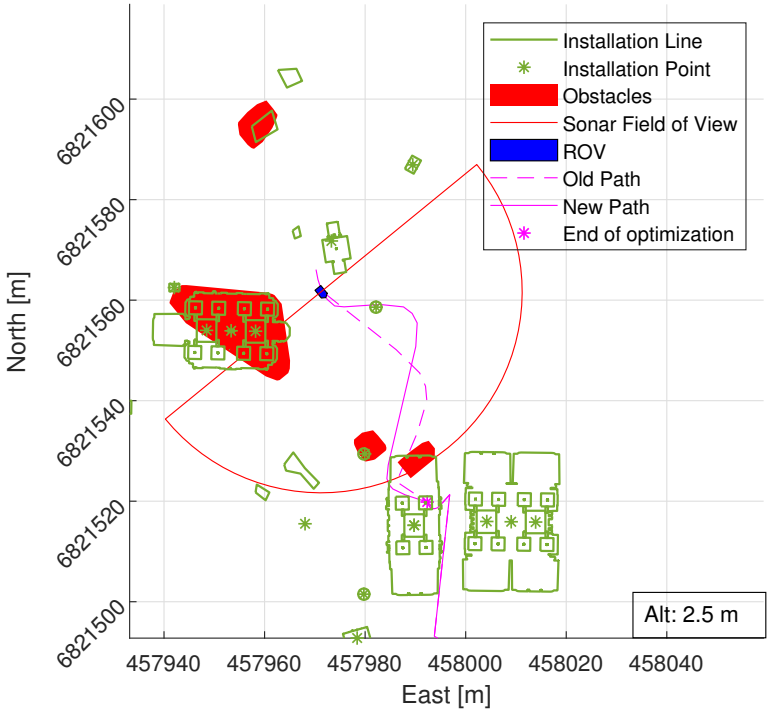

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 3.3 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

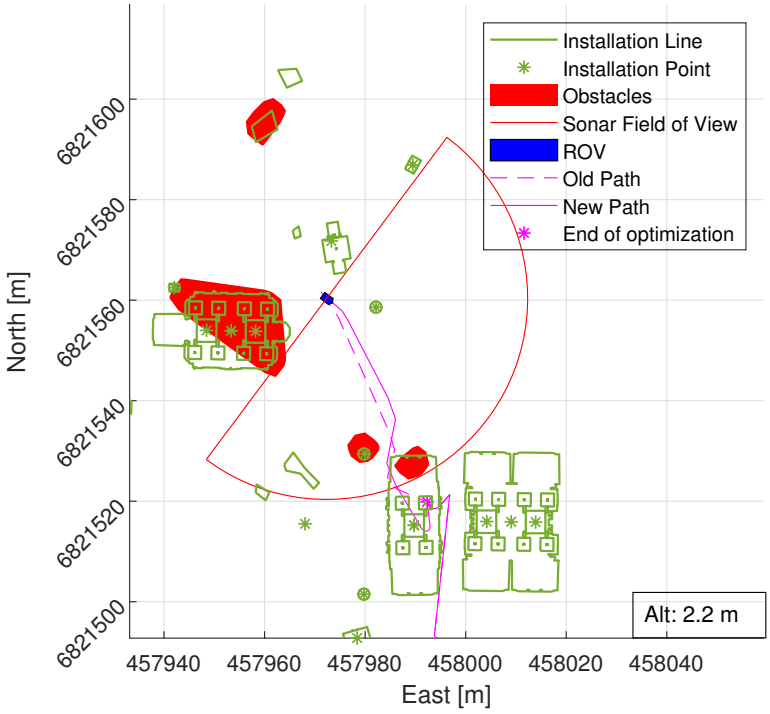

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 3.2 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

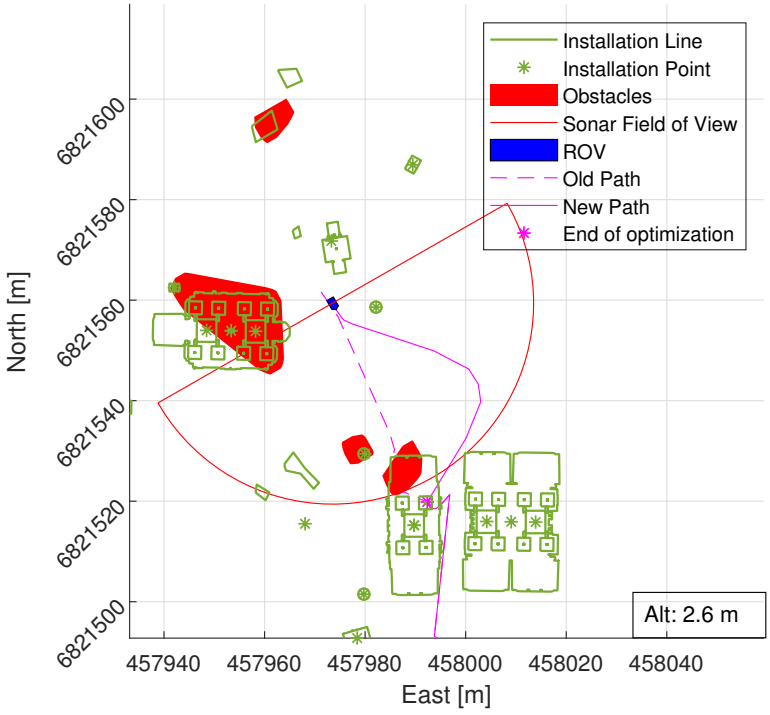

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 3.2 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

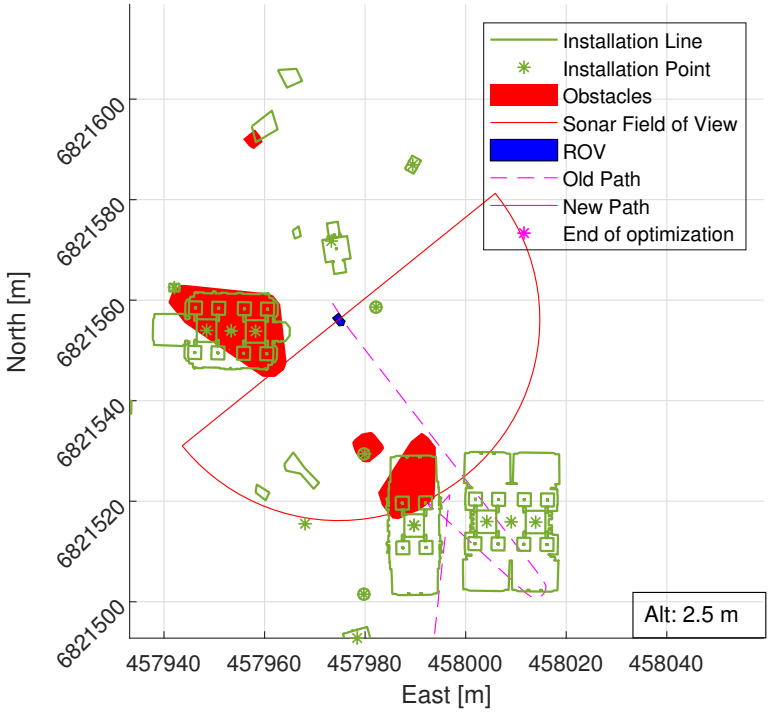

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040
Alt: 3.0 m

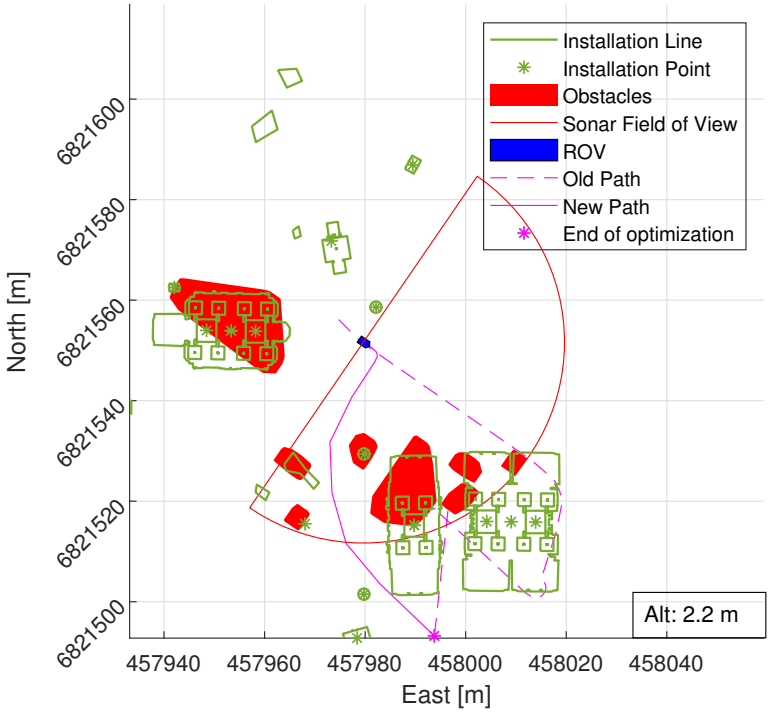

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040
Alt: 2.9 m

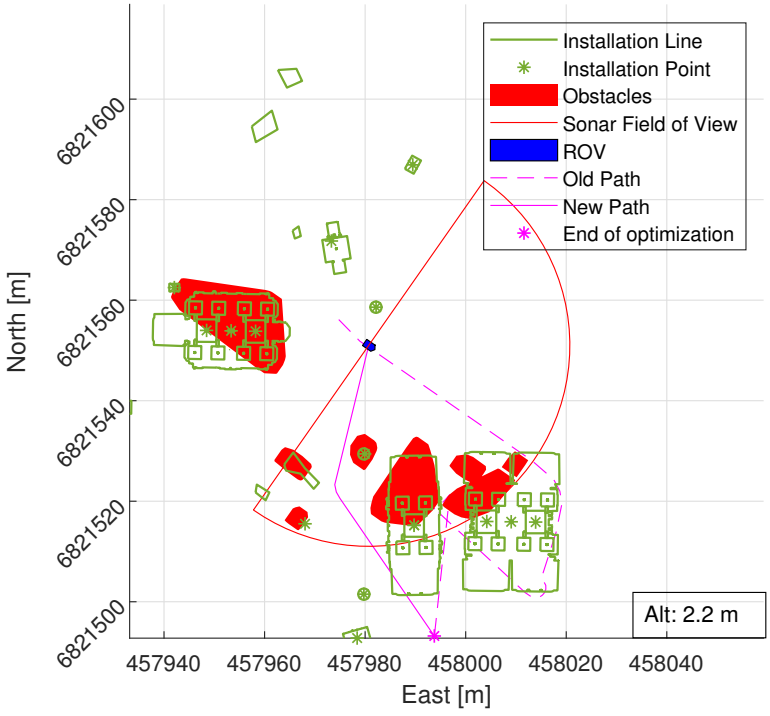

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.5 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

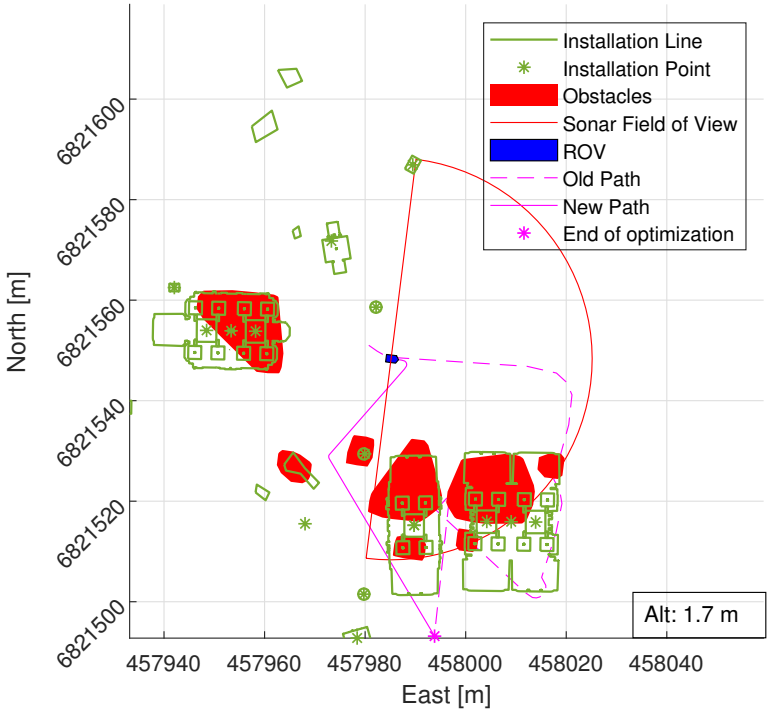

Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.2 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.6 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.5 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.2 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 2.2 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040


Installation Line
Installation Point
Obstacles
Sonar Field of View
ROV
Old Path
New Path
End of optimization
Alt: 1.7 m
North [m]
East [m]
6821600
6821580
6821560
6821540
6821520
6821500
457940
457960
457980
458000
458020
458040

Appendix C

# List of Electronic Attachments

## C.1 Attachments delivered in DAIM

The attached files delivered electronically in DAIM is listed here.

- A video describing the work done in the thesis was made as a mandatory assignment.
- PDF-version of abstract for paper contribution to *2018 IEEE OES Autonomous Underwater Vehicle Symposium*. The abstract is an A4-document, which means that it can be difficult to read in the B5-format.

## C.2 Source code

The source code developed in the project is available at `https://github.com/orjan1992/pySonar`.