Bathymetry SLAM using reduced rank Gaussian Processes and DVL range measurements

For real-time underwater position estimation

Danny Looman

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Bathymetry SLAM using reduced rank Gaussian Processes and DVL range measurements

For real-time underwater position estimation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Danny Looman

September 24, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Lobster Innovations BV. Their cooperation is hereby gratefully acknowledged.

Copyright © Delft Center for Systems and Control (DCSC)
All rights reserved.

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Bathymetry SLAM using reduced rank Gaussian Processes and DVL range measurements

by

Danny Looman

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: September 24, 2022

Supervisor(s): Dr. M. Kok

Ir. F.M. Viset

Reader(s): Dr. J.F.P. Kooij

Dr.ir. S.P. Mulders

Abstract

Underwater position estimation is challenging due to the absence of Global Navigation Satellite System (GNSS) signals. Underwater vehicles are typically equipped with a Doppler Velocity Log (DVL) that measures the velocity relative to the seafloor. Aside from the velocity, the DVL also measures the range of each of the four beams. When compared against a bathymetry height map, these measured ranges provide additional information enabling improving position estimates. Unfortunately, surveys often occur in areas where detailed bathymetry maps are unavailable [29]. In these cases, bathymetry Simultaneous Localization and Mapping (SLAM) could be used to improve the position estimates compared to the velocity integration position. With SLAM, the map is being estimated during the mission while at the same time using the map for position determination.

In this thesis, reduced rank Gaussian processes (GPs) are used as map representation for SLAM. The downside of regular GPs is a time complexity of $\mathcal{O}\left(n^3\right)$, reduced rank GPs improve the computation performance. GPs provide Gaussian distributions at any point, leading to a neat integration with probabilistic SLAM algorithms. To the best of the author's knowledge, this has not been used in bathymetry SLAM. This report investigates how reduced rank approximated GPs, representing the bathymetry, can be integrated into a SLAM algorithm to improve the position estimates of an underwater vehicle equipped with a DVL and low-quality gyroscopes.

A squared exponential kernel is used as GPs model of the bathymetry. The reduced rank approximation is vital for real-time SLAM performance. This map representation is integrated with a Rao Blackwellized particle filter (RBPF) that estimates both the underwater vehicle's trajectory and the bathymetry map.

The SLAM algorithm is evaluated using data from an underwater vehicle operated at the surface such that a GNSS reference position is available. Experiments of the SLAM algorithm show a reduced position error compared to the GNSS reference. The resulting algorithm has a computation time of up to 30 times faster than the Autonomous Underwater Vehicle (AUV) collects data while improving position estimates. This concludes that the RBPF using reduced rank GPs is capable of onboard improved position estimation on underwater vehicles.

Table of Contents

Preface

Before you lies the master thesis *“Bathymetry SLAM using reduced rank Gaussian Processes and DVL range measurements"*. It has been written to fulfill the graduation requirements of the Systems and Control program at the Delft University of Technology. I was engaged in researching and writing this thesis from September 2021 to September 2022.

I am grateful for the opportunity to execute a thesis of interest to my own company. However, while writing my thesis, I struggled with finding a balance between documenting the research of my thesis and future research related to Lobster. My broad interest in technological developments has led to some deviations from my graduation path. In the end, I reached a symbiosis in which I could use the knowledge of my thesis for Lobster and vice versa. A number of the results from this thesis will be applied in future missions that Lobster has scheduled executing. Other learning's of this thesis have found their way on the technical roadmap for the near future.

I would like to thank my supervisors, Manon Kok and Frida Viset, for their guidance and the detailed feedback you have provided me during the process. Finally, I want to thank my colleagues at Lobster Innovations for their time required to collect the datasets for this thesis, especially thanks to Daan for his help and guidance with programming in C^{++}. This has probably saved me days of time.

And for the reader: I wish you a pleasant reading.

Danny Looman

Delft, 21 September 2022

Chapter 1

Introduction

Oceans are ecosystems with an enormous impact on our lives. They determine the weather on our planet to a large extent, and many communities rely on the oceans as a food supply [31]. Yet humanity causes severe problems in the oceans, such as ocean warming, acidification, overfishing, and plastic pollution [31]. To mitigate the effects caused by these problem, it is vital to have a good understanding of the oceans' ecosystems to make rational decisions for a sustainable future.

Detailed underwater exploration is complex since the rapid attenuation of electromagnetic radiation limits both the range and field of view of high-resolution sensors [32]. The limited sensing range requires these surveys to be conducted in the vicinity of the seafloor to get detailed underwater information, such as visual data, detailed bathymetry data or magnetic field data. 90% of the oceans have a depth exceeding one kilometer as measured by surface area [6]. Reaching these deeper oceans is challenging due to hydrostatic pressures, severe communications constraints and the high costs of the required surface vessel to reach these remote areas. Due to these challenges, detailed deep ocean exploration is limited to some research sites of interest and industries with an economic interest, such as the oil & gas industry.

The technical challenges of the hydrostatic pressures and communications constraints are solved by manufacturing high-quality pressure chambers and the increased autonomy of underwater vehicles. However, as surveys have to be conducted near the seafloor, the mapped area per second is inherently low. At the same time, the associated costs of the surface vessel are high. Autonomous Underwater Vehicles (AUVs) are the preferred platform for surveys of large areas over remotely operated vehicles and towed sensors due to their autonomy. Also, AUVs are not subjected to the disturbances caused by a cable connected to the surface ship and enable the vessel to execute other tasks in the meantime [29]. To reduce the required costs of underwater surveys, a logical step forward would be to operate multiple underwater vehicles simultaneously from a single surface ship to increase the mapped area per second. One challenge is, that the current method of acoustic triangulation is not scalable to a fleet of AUVs as update rates decrease linearly with the fleet size.

For most use-cases, gathered data has to be geo-referenced to be of value. The required accuracy is application-dependent [5]. As Global Navigation Satellite System (GNSS) signals are fully attenuated by the seawater with a couple of centimeters, other solutions of underwater position estimation have been developed. Figure 1-1 shows the three main solutions for determining the position as identified by [5, 15, 29].

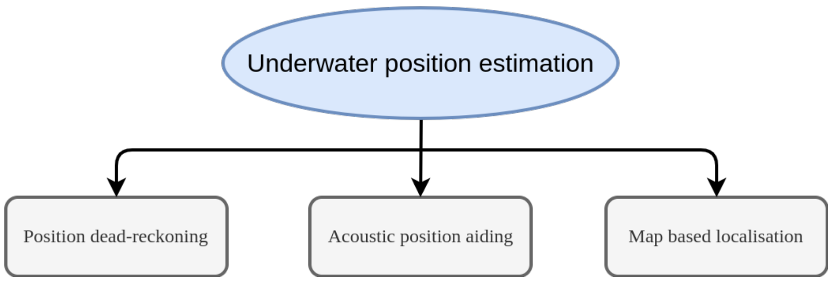


Figure 1-1: The three main strategies of underwater position estimation underwater [5, 15, 29].

The first strategy is position dead-reckoning, which only uses onboard sensors. The sensors used are a combination of inertial measurement units, a pressure sensor and a Doppler Velocity Log (DVL). The DVL measures the body velocity $\boldsymbol{v}^b$ relative to the seafloor. Together with the orientation estimate of the inertial measurement unit, the velocity can be integrated to a position estimate. All vehicle states are observable with this strategy, except the horizontal position of the underwater vehicle. Dependent on the accuracy of the sensors, a long term accuracy of less than 0.1% of distance traveled on the horizontal position estimate can be obtained [19].

The second approach is to use acoustic position aiding with a beacon setup. This is the state of the art approach to underwater position estimation as it is the most accurate due to the direct position measurements. One of such methods is Long Base Line (LBL) triangulation with 12 kHz acoustic beacons. This provides position accuracy of ± 10 meters and has an update rate of up to 20 seconds depending on the range [32]. Drawbacks of acoustic beacons include limited range, acoustic blind spots, multi-path propagation, reduced accuracy when used with multiple AUVs and setup time [23]. Prior to the underwater exploration mission, the speed of sound in the full water column has to be determined in order to calibrate the beacon system. These drawbacks limits the efficiency of offshore operations and motivate the research into map based navigation strategies.

The third approach is to use information from the environment and compare it against a reference map so one can localize itself on the map. Suitable environmental information are temperature, ocean currents, bathymetric variations or the magnetic field. Combination can be used as well, as long as the properties of interest has sufficient spatial variations. As these maps are often not available, or do not contain enough detail they must be generated beforehand with a surface vessel or an underwater vehicle aided by acoustic beacons. Another solution is to generate the reference map, and use the map for localization at the same time. This process is called Simultaneous Localization and Mapping (SLAM).

1-1 Research topic

Terrain relative navigation uses bathymetric variations to determine the position of an AUV and has been successfully used in practice [29]. Using terrain variability as information source is a elegant solution as most underwater vehicles are readily equipped with range sensors to operate safely near the seafloor. These sensors can be a single-point altimeter, a DVL, or a multi-beam echo sounder. Most survey grade underwater vehicles have a DVL as the velocity measurements are required for the accuracy of position dead-reckoning. As a byproduct of the velocity measurement, the DVL also measures the ranges of each beam by using time of flight calculations [4]. These range measurements are illustrated in Figure 1-2 in a birds-eye overview of a part of a mission. From the integration of velocities of the DVL and gyroscopes, we have a decent estimate about the traveled trajectory between any two measurements. Whenever the vehicle revisits a previously visited area, the collection of old range measurements combined with the trajectory estimate can be used to determine the most likely position based on the current measurements. This is the basic working principle of bathymetry SLAM.

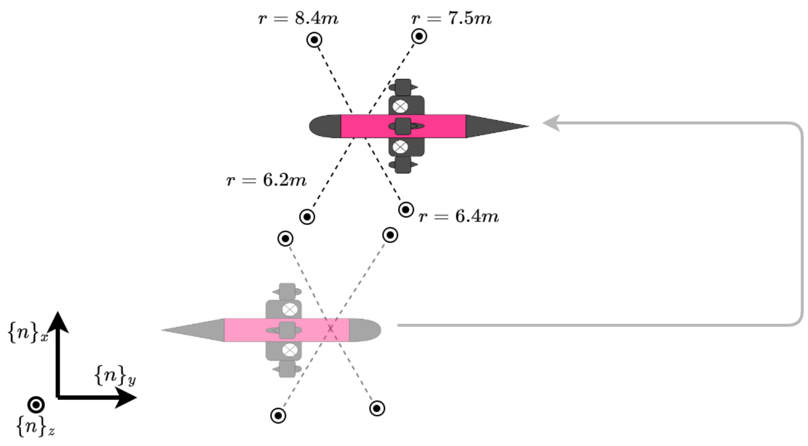


Figure 1-2: Birds eye overview of underwater vehicle performing a mission with the four DVL beams. The opaque underwater vehicle represents the position somewhere in the past. When revisiting positions, the old measurements can be used to evaluate the new measurements.

Bathymetric SLAM avoids the dependency on an available map by generating the map during the mission. If the AUV is able run SLAM onboard and accurately determine the position, no communication with the surface vessel or acoustic beacons is required. This makes the navigational performance independent of the number of AUVs used simultaneously. Due to these advantages, bathymetry SLAM could be a viable alternative to acoustic beacon systems for suitable missions.

One of the main challenges of a SLAM algorithm is how to store the bathymetry information effectively. Many SLAM algorithms uses a set of landmarks that are distinctive features in the environment. Automatic landmark recognition is challenging underwater as the terrain is typically sparse and simple geometric shapes cannot describe the variations accurately [29]. Therefore a featureless SLAM approach is preferred as landmark recognition is not required.

There are multiple ways to represent a featureless bathymetry map. One could use a full

3D occupancy gridmap representation, where each cube has a probability of being occupied. This has been used in [8]. A occupancy gridmap has as disadvantage that the model of the environment is discretized into cubes. As underwater vehicles operate in a 3D environment, the number of cubes and thus memory scale to the third power. This is not desirable for long range underwater missions as this can quickly become infeasible to store onboard.

An alternative map representation is to use a Gaussian process (GP) model, no discretization is needed and thus the seafloor height can be described at any point. [25] gives an extensive overview into GPs modeling and regression. A GP describes the probability over a distribution of functions. By defining the bathymetry function $b(\cdot)$ as

$$\boldsymbol{s}_z^n = b\left(\boldsymbol{s}_x^n,\ \boldsymbol{s}_y^n\right), \tag{1-1}$$

which relates the horizontal seafloor coordinates $\boldsymbol{s}_x^n$ and $\boldsymbol{s}_y^n$ to the corresponding depth value $\boldsymbol{s}_z^n$. If we model the function $b(\cdot)$ to be a realization of the GP, GP regression can be used as a map representation. Most of the seafloor are planes with some smooth height variations [6], this can be modeled as a 2D surface and is well described by Eq. (1-1). The fact that a GP map representation does not suffer from discretization errors, and that depth predictions from the GP model have a Gaussian distribution makes them interesting for SLAM.

The downside of the regular, dense GP implementation is that it is computationally expensive. The time complexity scales $\mathcal{O}(n^3)$, where n is the number of measurements, and the memory complexity scales $\mathcal{O}(n^2)$ [25]. Multiple approximations of GPs are developed to reduce computation and memory requirements, a selection of these methods are discussed in [25]. In [28], a reduced rank approximation was introduced that approximates the covariance function of the GP with a truncated series expansion. This approximation step leads to significant computational improvements that allow the processing of larger datasets. This method has not yet been used in the context of bathymetry SLAM.

Collaboration with Lobster Innovations

This thesis is performed in collaboration with the company Lobster Innovations. Lobster is a start-up company with as goal to simplify underwater data collections by rethinking underwater vehicles. As a result, Lobster has developed a vertically integrated AUV which is 10x lighter compared to competitors. This simplifies all operations with the AUV, leading to reduced handling time and making offshore operations more efficient. Currently Lobster is developing and testing an AUV for visual underwater surveys. This research is of interest to Lobster as it could allow for better positioning accuracy, without an upgrade to expensive navigation sensors.

Lobster's AUV is called the Scout, Figure 1-3 shows a picture of the Scout during one of the field tests. Datasets from those tests have been collected which allows for the independent research and validation of bathymetry based navigation methods. The sensors used in the Scout available to determine the position of the vehicle are:

- An inertial measurement unit consisting of a 3-axes accelerometer and gyroscope.
- An magnetometer for aiding the orientation estimation.
- A DVL for velocity aiding and range sensing.

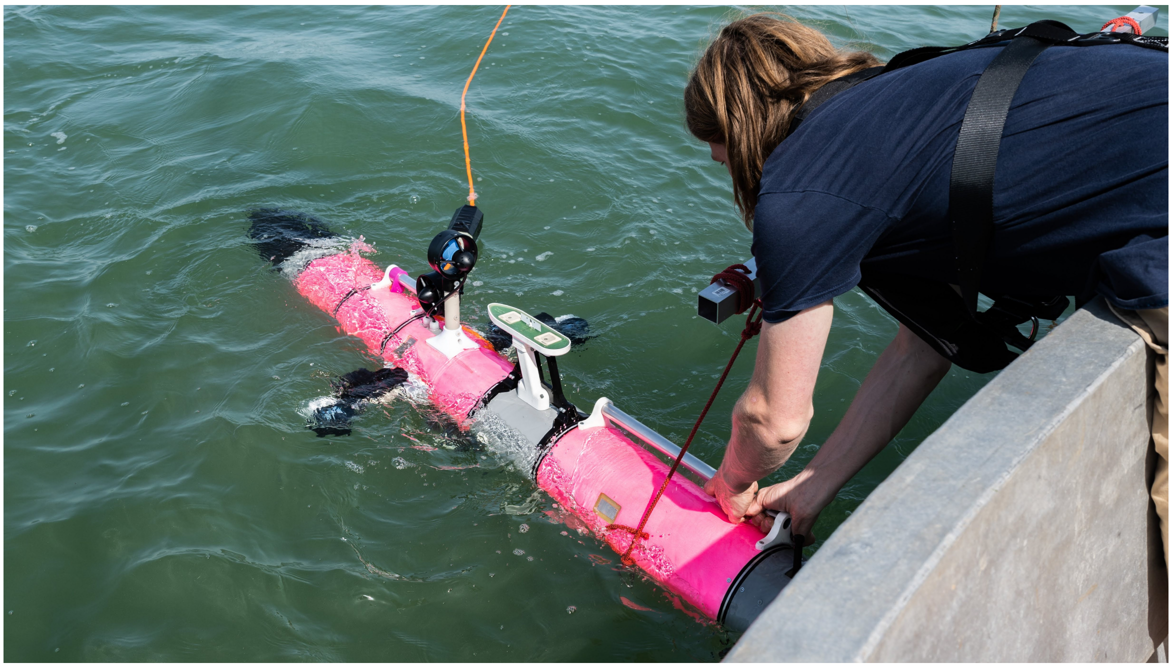


Figure 1-3: The Lobster Scout, the AUV used for experimental data collection for demonstration and validation of the proposed algorithms.

- A pressure sensor for determining the depth.
- A GNSS sensor which is used for geo-referencing the position when the AUV is floating at the surface.

Since the Scout employs low-cost micro-electro-mechanical systems (MEMS) gyroscopes, the heading estimate is magnetically aided. During field tests is observed that the magnetically determined heading suffers from distortions caused by systems in the vehicle. Based on other papers and the observations of the field tests, it is expected the inaccuracy in magnetically determined heading is the primary source of error in the position estimation filter [15, 18].

Research questions

To summarize, as part of making underwater data gathering more cost effective, the position estimation accuracy has to become independent of from the number of AUVs deployed. One potential solution is to use bathymetry SLAM. In order to verify the algorithm on experimental data, the Lobster Scout AUV is considered as platform. The key considerations are that the Scout has a DVL as range sensor and a magnetically determined heading which is not of the desired accuracy. This thesis aims to take a step towards accurate underwater position estimation capable of running in real-time by answering the main research question

> **How can reduced rank GP bathymetry SLAM with MEMS gyroscopes and DVL as main sensors be used to improve the position estimation of underwater vehicles?**

The following sub-questions support the answer to the main research question:

1. How can the bathymetry be estimated using reduced rank GP regression assumed with known position and which sources of errors are present in the map?
2. How can the bathymetry be used to improve the position estimate and how influential are the GP hyperparameters on the position estimation accuracy?
3. Is reduced rank GP SLAM able to run onboard of an underwater vehicle in real-time?

1-2 Organisation

The outline of the report is summarized in Table 1-1. Throughout this report, variable naming conventions are used consistently unless stated otherwise. Bold variables indicate that the variable is a vector $\boldsymbol{x} \in \mathbb{R}^n$. Capital letters indicate that the variable is a matrix $R \in \mathbb{R}^{n \times m}$. Scalar variables are mostly indicated with a lowercase letter such as $r \in \mathbb{R}$.

Chapter	Contents	Reading motivation
2	Relevant Work	Discusses literature in the field of bathymetry based navigation and other applications of reduced rank GPs.
3	Models	Introduces the underlying models of the SLAM algorithm including the dynamical and the measurement model.
4	Methodology	Explains the reduced rank approximation of the GP model and the integration with the SLAM algorithm.
5	Results	Reports the experimental test based on AUV data.
6	Discussion & Conclusions	Evaluates results and answers research questions

Table 1-1: Report contents and reading motivation

Chapter 2

Related Work

2-1 Position estimation based on bathymetry variations

The concept of using the seafloor depth profile as an information source for underwater vehicles is a well-known application [13, 29]. Two factors mainly determine the performance that can be obtained from any terrain navigation system, terrain variability and sensor capability [21]. Using a single or a multi-beam echo sounder for navigation given a bathymetry map is discussed in [13] where two different algorithms are suggested. First is the usage of a so-called robust, non-statistical minimum average distance between the map depth and the measured depth. The second, more sophisticated algorithm uses a point mass filter to estimate the position probability density function on a deterministic grid. A particle filter is closely related to the point mass filter and replaces the deterministic grid with a stochastic grid, which is a more efficient representation according to [12]. The particle filter has been used for bathymetry localization in [14]. All of these publications report an increase in navigation accuracy, which is the reason why these methods are widely accepted and used in practice. [29] even argues that localization on a known bathymetry map is the only suitable way for long-range beacon-less accurate underwater positioning.

Most of the literature found on bathymetry variations assumes a known bathymetry map that serves as reference. When narrowing the scope down to bathymetry Simultaneous Localization and Mapping (SLAM) algorithms, it turns out that almost all bathymetry SLAM algorithms use a featureless map representation. In [29], a reason for the featureless maps is presented, reliable feature extraction from range data is difficult in underwater environments due to the typically low resolution of sensors and amorphous shape of natural features.

In [30], a SLAM algorithm was tested based on feature locations extracted from sonar data. The location of these features was stored as additional states in the Extended Kalman Filter (EKF). To obtain the presented results, a side-scan sonar was used in this research that provides much more detail. Using a side-scan sonar, it is possible to observe small ripples in the sand, see Figure 2-2. Additionally, they manually checked all feature extractions for research reasons to eliminate any wrong classifications. After a data smoothing step, the maximum error is decreased by a factor of 10 compared to the smoothed dead-reckoned trajectory. The results presented in [30] are not comparable to the research question posed in this report. In this report, a Doppler Velocity Log (DVL) is considered as a range sensor. The DVL has limited sensing capabilities with just four beams. The type of features one can describe with four beams are very amorphous. Therefore, using a feature based SLAM method with a DVL seems unrealistic.

In [8], the focus is a full 3D SLAM algorithm with the goal of providing position estimates during a sinkhole inspection. Therefore a full 3D occupancy grid was chosen as map representation. A Bresenham ray tracing algorithm was used to estimate the expected range observation. Given the discretization of the voxel map, the usage of the ray tracing algorithm is a logical choice since it depends on the discretization of the voxel map. The downside of grid maps is the discretization of the volume into cubes, which causes a loss of detail, and the associated cubic memory requirements with the represented volume.

To the best of the author's knowledge, the work of Stephan Barkby [1] is the publication most related to this research. It is the only publication that uses a Gaussian process (GP) model for representing the bathymetry in a SLAM application. In [1], a Rao Blackwellized particle filter (RBPF) is used as an algorithm for SLAM, and GP regression was used for a local map reconstruction to obtain bathymetry predictions in overlapping areas. These predictions are used in turn to update the particle weights. The used sensor for perceiving the seafloor is a Reson 7125 400 kHz multi-beam sonar, which provides 480 beams uniformly distributed across 120°. Furthermore, the Autonomous Underwater Vehicle (AUV) is equipped with high-quality gyroscopes, providing an orientation accuracy of approximately 0.01° and a DVL for accurate velocity measurements. These sensors deviate significantly from the setup of this research, as this sensor setup allows for more accurate odometry information and more accurate sensing of the environment.

In order to work efficiently with GPs in [1], two important complexity reductions steps were made. Firstly, smart data storage was used that reduces duplicate storage between particles. Secondly, a covariance function that introduces sparsity in the covariance matrix as

$$k\left(\boldsymbol{s}, \boldsymbol{s}'\right) = \begin{cases} \sigma_0 \left[\frac{2+\cos\left(2\pi \frac{|\boldsymbol{s}-\boldsymbol{s}'|}{l}\right)}{3} \left(1 - \frac{|\boldsymbol{s}-\boldsymbol{s}'|}{l}\right) + \frac{1}{2\pi} \sin\left(2\pi \frac{|\boldsymbol{s}-\boldsymbol{s}'|}{l}\right) \right] & \text{if } |\boldsymbol{s}-\boldsymbol{s}'| < l, \\ 0 & \text{if } |\boldsymbol{s}-\boldsymbol{s}'| \geq l, \end{cases} \tag{2-1}$$

was used as GP model as this function transitions to precisely zero, where the regular squared exponential (SE) covariance function never reaches zero. As a consequence of using the sparse covariance function, only points within a certain distance have to be considered in the calculations. Both covariance functions are visualized in Figure 2-1 to highlight the difference. If the hyperparameters are adequately chosen, both functions will have similar shapes. As a consequence, the assumptions placed on the seafloor are similar. The SLAM simulations are run offline on experimental data and improve the dead-reckoned position estimate.

All of the mentioned publications used sensors of higher quality than the considered sensor suite of this report. The underwater vehicles are employed with a high-quality inertial measurement unit consisting of gyroscopes and accelerometers. Those systems are able to determine its orientation with an accuracy of 0.01°. This results in better odometry data, and thus the particle filter can be configured accordingly where the particle spread grows

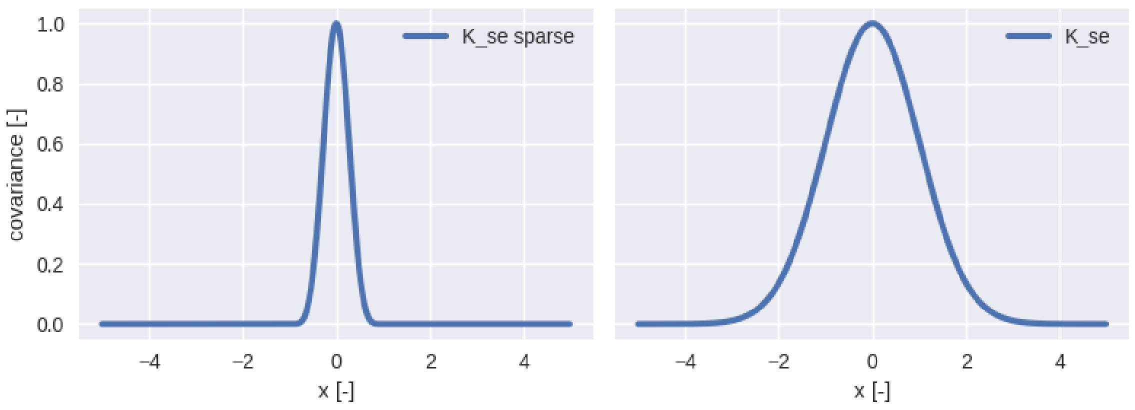
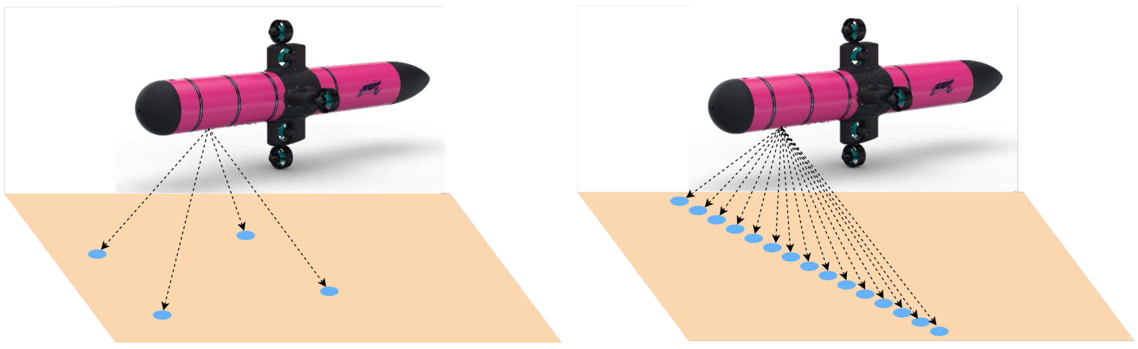


Figure 2-1: Comparison between: left, spare SE (2-1) covariance function; right, regular SE covariance function (3-18). All hyperparameters are set with a value of 1.

slowly over time. A low-noise dynamic model of the particle filter results in a SLAM algorithm that is more robust against slowly varying terrain. Apart from the high-quality dead-reckoning sensors, these publications also have used a multi-beam or side-scan sonar that senses 100 − 500 beams during each ping, providing much more detail in the bathymetry information compared to a DVL, as illustrated in Figure 2-2.



(a) Beam pattern of a DVL. **(b)** Beam pattern of a multi-beam sonar.

Figure 2-2: Visualization of beam patterns of the DVL in Figure 2-2a, and of a multi-beam sonar in Figure 2-2b. The number of beams used in a multi-beam echo sounder varies between 100 - 500, dependent on the model. Most of the DVL models have only four beams.

In [21], the trade-offs of using low-quality sensors were investigated on the performance of bathymetry localization given a known map. The summarized conclusions are that fewer beams lead to significantly longer convergence distance, and the increase in motion uncertainty mainly lowered the maximum achievable accuracy. Numerical values obtained from field tests of the convergence distance and achievable accuracy are reported in [21]. No convincing argument was made against SLAM using a DVL. However, as the sensor setup is of lower quality, the expected results should be lower than the mentioned SLAM studies.

2-2 Reduced rank GPs in magnetic field estimation

This section covers relevant literature for a reduced rank approximation of the covariance matrix. A known disadvantage of using Gaussian process regression is the poor scalability with increasing dataset size n. This is due to the inversion of the covariance matrix (3-17) when evaluating test points, The covariance matrix K has size $n \times n$ and the time complexity of matrix inversion is $\mathcal{O}(n^3)$ [25].

In [25], multiple approximations are introduced to make GPs scalable to larger datasets where $n > 10^4$. Reduced rank methods are approximating the covariance matrix K of the GP with $\tilde{K} \in \mathbb{R}^{m \times m}$, where $\tilde{K}$ is of lower rank $m < n$. An obvious approximation of K is to take the m largest eigenvalues of K and construct $\tilde{K} = \Phi \Lambda \Phi^\top$ where Λ is an diagonal matrix with the m largest eigenvalues and Φ are the corresponding eigenvectors. This decomposition has to be recalculated each time the K matrix expands, and computing this decomposition is also an $\mathcal{O}(n^3)$ operation. Therefore the eigendecomposition is not beneficial to speed up the calculations [25, 28].

One reduced rank approximation that can speed up the calculations is the reduced rank approximation presented in [28]. This approximation has a computational cost of $\mathcal{O}(nm^2)$ and memory requirements of $\mathcal{O}(nm)$, which becomes beneficial when $m < n$. This reduced rank approximation will be used in this report and will be covered in detail in Section 4-1. This approximation has been applied for magnetic field modeling and estimation, and those publications are discussed in the remainder of this chapter.

In [27], the reduced rank method was used for the modeling and interpolating of the measured magnetic field. This paper extends the batch processing methods with a sequential algorithm. Other methods only considered batch estimation, where the data is first collected and then bundled and processed afterward. GP interference is the solution to a linear Gaussian estimation problem. This type of problem has a commonly known sequential solution in the form of a Kalman filter. The sequential algorithm converts the estimation problem into Kalman filter equations and can be used for online applications such as SLAM.

In [17], the reduced rank approximation of [27] was used to represent the magnetic field in a SLAM algorithm. The SLAM algorithm uses particle filtering where the map state vector is the largest part of the state vector. The map states enter conditionally linearly in the measurement model, therefore a RBPF is used to reduce the sampling dimension. The map state estimate can be estimated by the previously described sequential GP format. The result of this formulation is a RBPF and a computationally tractable SLAM algorithm using GPs as map representation.

Chapter 3

Models

3-1 Scout Navigation System

The Scout is equipped with the sensor suite that includes a 3-axes accelerometer, 3-axes gyroscope, 3-axes magnetometer, Global Navigation Satellite System (GNSS) antenna, pressure sensor, and Doppler Velocity Log (DVL). These measurements are fused using an Extended Kalman Filter (EKF) with a kinematic dynamic model to estimate the vehicle state consisting of the pose and (angular) velocities. The EKF setup is similar to the filter described in [22]. The main deviation is that the orientation is estimated using quaternions instead of Euler angles, see Appendix A. The EKF outputs the vehicle's estimated state consistently at 300 Hz, these estimates are the inputs used for the control, mission planning, and mission execution of the underwater vehicle. The estimated state is described in two reference frames which are visualized in Figure 3-1. These frames are:

- The Earth fixed navigation frame $\{n\}$ in North-East-Down (NED) configuration. The x-axis points in the direction of the geodetic north, the y-axis points east, and the z-axis points downwards. The origin is chosen to be near the mission start at the surface, such that the horizontal plane is tangent with the Earth's surface. Due to this convention, one can approximate the local mission area as flat and use the $\{n\}$ frame for locally describing vehicle pose in the flat world. As Autonomous Underwater Vehicles (AUVs) are typically battery powered, the operation range is inherently limited, and the flat Earth approximation errors are negligible [10]. By saving the longitude and latitude of the $\{n\}$ frame origin, all positions described in the $\{n\}$ frame can be converted to longitude and latitude angles during post-processing, for example using the methods of [10].
- The body-fixed reference frame $\{b\}$ has its origin moving with the underwater vehicle and is chosen to coincide with the vehicle's center of mass. The x-axis points forwards, the z-axis points downwards, and the y-axis points to the right to complete a right-handed coordinate frame.

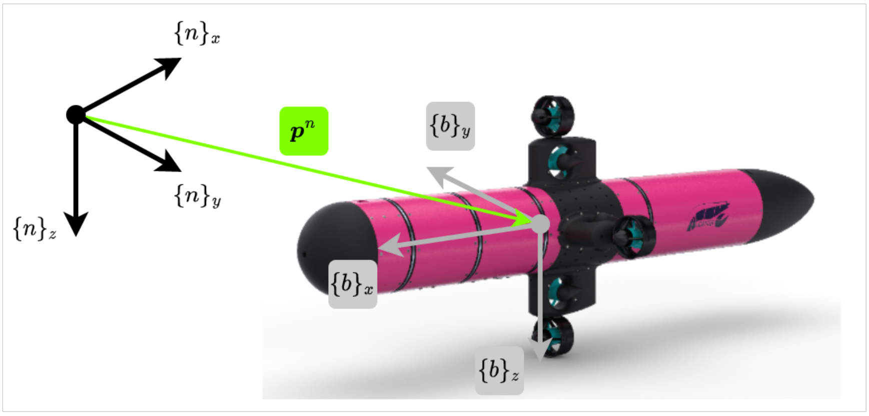


Figure 3-1: Illustration of the Earth fixed navigation $\{n\}$ frame and the underwater vehicle's body fixed $\{b\}$ frame.

The EKF's estimated state is

$$\boldsymbol{x}_k^{\text{EKF}} = \begin{bmatrix} \boldsymbol{p}_k^n & \phi_k^{nb} & \theta_k^{nb} & \psi_k^{nb} & \boldsymbol{v}_k^b & \boldsymbol{\omega}_k^b \end{bmatrix}^\top . \tag{3-1}$$

In Eq. (3-1), $\boldsymbol{p}_k^n \in \mathbb{R}^3$ is the position of origin of the $\{b\}$ frame. ϕ_k^{nb}, θ_k^{nb} and ψ_k^{nb} are the roll, pitch and yaw Euler angles, as introduced in Appendix A. $\boldsymbol{v}_k^b \in \mathbb{R}^3$ is the linear velocity and $\boldsymbol{\omega}_k^b \in \mathbb{R}^3$ is the angular velocity of the AUV. After descending underwater, the GNSS antenna no longer receives messages, and the horizontal position becomes unobservable. When relying on the inertial sensors and DVL only, the horizontal position is estimated by integrating the velocity, which results in the accumulation of integration errors. Note that the vertical position $\boldsymbol{p}_{k,z}^n$ is observable by measuring the hydrostatic pressure with a pressure sensor.

Since this research considers low-quality gyroscopes, these sensors cannot distinguish the Earth's rotation from the bias drift of the gyroscope due to the relatively high noise and drift levels. The EKF uses a magnetometer and accelerometer for aiding the orientation estimates as described in [16]. The attitude update of the accelerometer only considers the gravity vector, which is reasonable as the underwater environment is highly damped. The angles relative to the gravity vector can be estimated, which results in aiding the roll ϕ and pitch θ angles.

The magnetometer is modeled in the EKF only to measure the Earth's magnetic field. With this model, the orientation heading ψ angle is observable. However, the assumption that the magnetometer only measures the Earth's magnetic field is very restrictive. Ferromagnetic material in the neighborhood of the sensor violates this assumption. Therefore, the magnetic sensors have to be calibrated to reduce these unmodelled effects. In [24], multiple calibration algorithms are described and compared. However, AUVs also emits varying magnetic fields from high-power onboard equipment, such as thrusters and lights [3]. These are varying distortions in the magnetic which can not be calibrated for using one of the methods presented in [24]. The reason for using the magnetometer for heading estimation is practical. It is the best solution given a limited budget. The typical accuracy of magnetically determined heading is $\pm$ 5 degrees [18].

The velocity is accurately measured by the DVL. Therefore the heading inaccuracy can be the primary source of position integration errors due to the rotation of the $\{b\}$ to the $\{n\}$ frame.

As the aim is to improve the position estimates, the heading angle should also be considered in the estimation process. Although the ϕ and θ estimates of the EKF is influenced by magnetic field variations, this effect is small as the accelerometer also estimates these states. For the experiments in this report will be assumed that this influence is negligible and that the EKF estimated states are deterministic.

To summarize, the full pose of an underwater vehicle is described with 6 states as

$$\begin{bmatrix} \boldsymbol{p}^n_{k,x} & \boldsymbol{p}^n_{k,y} & \boldsymbol{p}^n_{k,z} & \phi^{nb}_k & \theta^{nb}_k & \psi^{nb}_k \end{bmatrix}^\top .$$

Part of these states are observable by the considered sensors and the EKF estimates $\boldsymbol{p}^n_{k,z}$, ϕ^{nb}_k and θ^{nb}_k with good accuracy. Therefore the estimation space can be reduced by considering these EKF estimates as deterministic inputs for the Simultaneous Localization and Mapping (SLAM) algorithm. The pose states that are not observable or inaccurate, are

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}^n_{k,x} \\ \boldsymbol{p}^n_{k,y} \\ \psi^{nb}_k \end{bmatrix} . \tag{3-2}$$

3-2 Dynamic model

The goal of the SLAM algorithm is to estimate the states of Eq. (3-2) using bathymetry information. We know that a state in the future $\boldsymbol{x}_{k+1}$ is related to the previous state. This section describes the describes the dynamical model describing the state evolution. The standard form of a dynamical state space model is

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\ \boldsymbol{u}^{\text{dyn}}_k,\ \boldsymbol{e}^{\text{dyn}}_k). \tag{3-3}$$

Where in Eq. (3-3), $\boldsymbol{u}^{\text{dyn}}_k$ is the input vector of deterministic variables, and $\boldsymbol{e}^{\text{dyn}}_k$ is the process noise input. A mathematical model never describes reality perfectly, the process noise is modeled as a way to represent these inaccuracies. In the underwater vehicle, there is a EKF estimating the observable vehicle states consistently at 300 Hz. In a loosely coupled approach, the estimates of the EKF are used as inputs to the dynamical model as visualized in Figure 3-2.

Based on the EKF estimates of Eq. (3-1), a change in position can be calculated by integrating the velocity, and the change in heading can calculated by integrating the angular velocity. The heading state is chosen to be modeled with a change in heading from the EKF. Since the delta times are short, this is approximately equivalent to integrating the angular velocity. The resulting dynamical model that describes the AUV's motion is

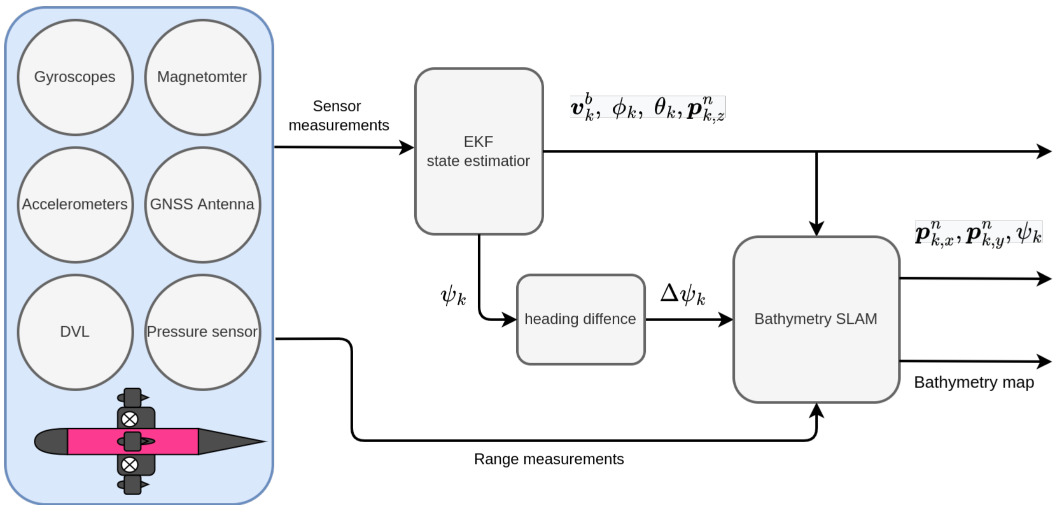


Figure 3-2: Schematic overview of the proposed implementation onboard an AUV. The SLAM algorithm will run parallel to the EKF.

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k,\ \boldsymbol{u}_k^{\text{dyn}},\ \boldsymbol{e}_k^{\text{dyn}}) \\
&= \begin{pmatrix} \begin{bmatrix} \boldsymbol{p}_{k,x}^n \\ \boldsymbol{p}_{k,y}^n \end{bmatrix} + \begin{bmatrix} T_k & 0 & 0 \\ 0 & T_k & 0 \end{bmatrix} R(\phi_k^{nb}, \theta_k^{nb}, \psi_k^{nb}) \left(\Delta \boldsymbol{v}_k^b + \boldsymbol{e}_{v,k}\right) \\ \psi_k^{nb} + \Delta\psi_k + e_{\psi,k} \end{pmatrix}, \\
\boldsymbol{e}_k^{\text{dyn}} &= \begin{bmatrix} \boldsymbol{e}_{v,k} \\ e_{\psi,k} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0},\ I_4 \begin{bmatrix} \sigma_v^2 \\ \sigma_v^2 \\ \sigma_v^2 \\ \sigma_\psi^2 \end{bmatrix}\right), \\
\boldsymbol{u}_k^{\text{dyn}} &= \begin{bmatrix} T_k & \phi_k^{nb} & \theta_k^{nb} & (\boldsymbol{v}_k^b)^\top & \Delta\psi_k \end{bmatrix}^\top .
\end{aligned} \tag{3-4}$$

Where

- T_k is the time difference $T_k = t_{k+1} - t_k$, which is not constant due to the range dependent update rates of the DVL.
- $\boldsymbol{e}_v$ is the velocity noise, as modeled by Gaussian noise with the covariance of $\sigma_v^2 I_3$.
- $\Delta\psi_k = \psi_{k+1} - \psi_k$ is the heading difference between two consecutive heading angles as stored in the dataset.
- $e_{\psi,k}$ is the heading uncertainty, which is modeled by a Gaussian distribution with the covariance of σ_ψ^2.

Eq. (3-4) is a zero-order hold model, it is assumed that the velocity is constant over the time period which is a simplification of reality. However, the proposed model results in nearly the same trajectory as the dead-reckoned EKF trajectory.

3-3 Preprocessing of range measurements

The DVL, or any other type of acoustic range sensor, measures the range $r \in \mathbb{R}$ by emitting a narrow sound pulse and measuring the time Δt_{echo} before an echo is detected. The range is calculated from a time of flight calculation as

$$r = \frac{\Delta t_{\text{echo}}}{2c}. \tag{3-5}$$

In Eq. (3-5), c is the speed of sound in the local water conditions, and the time is divided by two as the sound wave has traveled twice the distance in the measured Δt_{echo}. The speed of sound c is assumed to be known when using Eq. (3-5) and has to be configured within the DVL before use. The sound speed depends significantly on the surrounding water's salinity, depth, and temperature. In [9], an empirically determined algorithm for calculating c based on the environment's salinity, depth, and temperature measurements.

The range r is modeled as the length of a vector parallel to unit vector $\boldsymbol{\alpha}^b \in \mathbb{R}^3$. A DVL has typically four beams $j \in \{1, 2, 3, 4\}$ where each beam measures an independent range $r_{k,j}$ at time instance k. The range measurement is modeled to be normally distributed with a mean equal to the true range value $r_{k,j}^{\text{true}}$ and a standard deviation of σ_r as

$$r_{k,j} \sim \mathcal{N}(r_{k,j}^{\text{true}}, \sigma_r^2). \tag{3-6}$$

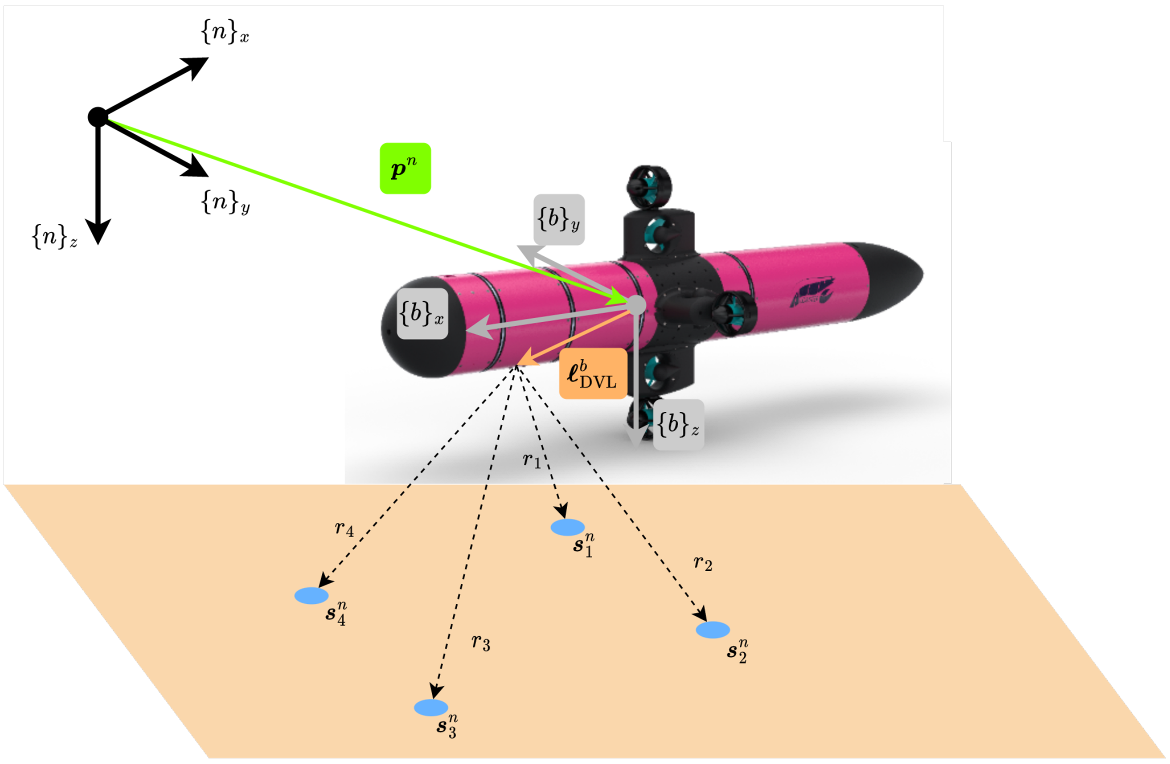


Figure 3-3: Visualization of range measurements and how they are related to seafloor points $\boldsymbol{s}$.

Figure 3-3 shows a schematic overview of the AUV and the modeled range vectors. The DVL is mounted at location $\boldsymbol{\ell}^b_{\text{DVL}} \in \mathbb{R}^3$, which is measured from the origin of the $\{b\}$ frame. By assumption that the sensor is rigidly mounted on the AUV, $\boldsymbol{\ell}^b_{\text{DVL}}$ and the beam direction vectors $\boldsymbol{\alpha}^b_j$ are constant and known in the $\{b\}$ frame. Table 3-1 lists the numeric values for these constant parameters.

Variable	Value	Units
$\boldsymbol{\ell}^b_{\text{DVL}}$	$\begin{bmatrix} -0.226 & 0.0 & 0.098 \end{bmatrix}^\top$	meters
$\boldsymbol{\alpha}^b_1$	$\begin{bmatrix} -0.271 & 0.271 & 0.924 \end{bmatrix}^\top$	-
$\boldsymbol{\alpha}^b_2$	$\begin{bmatrix} -0.271 & -0.271 & 0.924 \end{bmatrix}^\top$	-
$\boldsymbol{\alpha}^b_3$	$\begin{bmatrix} 0.271 & -0.271 & 0.924 \end{bmatrix}^\top$	-
$\boldsymbol{\alpha}^b_4$	$\begin{bmatrix} 0.271 & 0.271 & 0.924 \end{bmatrix}^\top$	-

Table 3-1: DVL mounting and direction vectors of the Scout with DVL-A50

The range vector points to the seafloor at locations $\boldsymbol{s}^n_{k,j} \in \mathbb{R}^3$, see Figure 3-3. The seafloor point can be expressed as a geometric conversion consisting of translations and rotations. The geometric conversion from a range $r_{k,j}$ to $\boldsymbol{s}^n_{k,j}$ is

$$\boldsymbol{s}^n_{k,j} = \boldsymbol{p}^n_k + R(\phi^{nb}_k, \theta^{nb}_k, \psi^{nb}_k)\left(\boldsymbol{\ell}^b_{\text{DVL}} + \boldsymbol{\alpha}^b_j r_{k,j}\right). \tag{3-7}$$

Note that Eq. (3-7) requires the knowledge of the vehicle states we wish to estimate from Eq. (3-2). Therefore these states can not be used during preprocessing, as the measurements should be independent of the state vector. However, due to the choice of the Euler multiplication order

$$R(\phi^{nb}_k, \theta^{nb}_k, \psi^{nb}_k) = R(\psi^{nb}_k)R(\theta^{nb}_k)R(\phi^{nb}_k), \tag{3-8}$$

the bathymetry depth $\boldsymbol{s}^n_{k,j,z}$ is independent of the AUVs horizontal position and heading angle. An element-wise expansion of Eq. (3-7) is shown in Appendix B, showing that this statement holds. Therefore, the range measurement can be preprocessed to a bathymetry depth, defined as measurement $y_{k,j}$ for the bathymetry SLAM algorithm. The preprocessing step of range measurement to $y_{k,j}$ is

$$y_{k,j} \equiv \boldsymbol{p}^n_{k,z} + \begin{bmatrix} \sin\theta^{nb}_k & \cos\theta^{nb}_k \sin\phi^{nb}_k & \cos\theta^{nb}_k \cos\phi^{nb}_k \end{bmatrix}\left(\boldsymbol{\ell}^b_{\text{DVL}} + \boldsymbol{\alpha}^b_j r_{k,j}\right), \tag{3-9}$$

where $r_{k,j} \sim \mathcal{N}(r^{\text{true}}_{k,j}, \sigma^2_r)$ and the range measurement is compensated for depth, offset and leveled to the horizontal plane. As $y_{k,j}$ is a function of the range, which is a random variable, $y_{k,j}$ will also be a random variable that is distributed as

$$y_{k,j} \sim \mathcal{N}\left(\boldsymbol{p}^n_{k,z} + \boldsymbol{d}_k\left(\boldsymbol{\ell}^b_{\text{DVL}} + \boldsymbol{\alpha}^b_j r^{\text{true}}_{k,j}\right),\ \left(\boldsymbol{d}_k \boldsymbol{\alpha}^b_j\right)^2 \sigma^2_r\right), \tag{3-10}$$

Where

$$\boldsymbol{d}_k = \begin{bmatrix} \sin\theta^{nb}_k & \cos\theta^{nb}_k \sin\phi^{nb}_k & \cos\theta^{nb}_k \cos\phi^{nb}_k \end{bmatrix}.$$

The measurement vector $\boldsymbol{y}_k$ is the column vector containing each of the four bathymetry measurements as

$$\boldsymbol{y}_k = \begin{bmatrix} y_{k,1} \\ y_{k,2} \\ y_{k,3} \\ y_{k,4} \end{bmatrix}. \tag{3-11}$$

3-4 Gaussian Process model for bathymetry

For a SLAM setting, the bathymetry is unknown and has to be estimated. We can define and assume the existence of a bathymetry function $b(\cdot)$ relating the horizontal position coordinate to a seafloor depth as

$$\begin{aligned} \boldsymbol{s}_{xy} &= \begin{bmatrix} \boldsymbol{s}_x^n & \boldsymbol{s}_y^n \end{bmatrix}, \\ \boldsymbol{s}_z^n &= b\left(\boldsymbol{s}_{xy}\right). \end{aligned} \tag{3-12}$$

where $\boldsymbol{s}^n$ are the seafloor points of Figure 3-3. A Gaussian process (GP) is a collection of random variables, where any finite selection of the random variables has a joint Gaussian distribution. A GP is a generalization of the Gaussian probability distribution. Where the Gaussian distribution describes random variables, the GP describes the properties of unknown functions. The function $b\left(\boldsymbol{s}_{xy}\right)$ is modeled to be a realization of a GP as

$$b\left(\boldsymbol{s}_{xy}\right) \sim \mathcal{GP}\left(0,\ k\left(\boldsymbol{s}_{xy},\ \boldsymbol{s}'_{xy}\right)\right). \tag{3-13}$$

In Eq. (3-13), $k(\boldsymbol{s}_{xy},\ \boldsymbol{s}'_{xy})$ is the covariance function and describes the spatial correlation between inputs $\boldsymbol{s}_{xy}$ and $\boldsymbol{s}'_{xy}$. The covariance function is chosen in advance and models the prior belief about the seafloor. The preprocessed measurement $y_{k,j}$ of Eq. (3-9) contains information about the bathymetry. With the chosen models, $y_{k,j}$ is the output of the Gaussian process.

As GPs describe relations between random variables where any selection is described by a (joint) Gaussian distribution, the assumption is made that the $m(\cdot)$ function is unique as each input will be described by a Gaussian that can only have a single mode. This assumption will make it very difficult to apply GP bathymetry mapping in areas underwater when there are caves or cliff overhangs. Since these scenes are assumed to be uncommon, the uniqueness property is not considered restrictive.

The bathymetry function Eq. (3-13) is chosen to be time-invariant, which is a reasonable assumption as the seafloor is not expected to change significantly over short time intervals. Therefore the measurement time index is not relevant and the notation will be simplified to $y_{k,j} = y_i$, where the relation between subscripts is $i = 4k + j$. The bathymetry measurement is modeled to be the output of the GP as

$$y_i = b\left(\boldsymbol{s}_{xy,i}\right) + e_i, \qquad e_i \sim \mathcal{N}\left(0, \left(\boldsymbol{d}_i \boldsymbol{\alpha}_i^b\right)^2 \sigma_r^2\right). \tag{3-14}$$

Given a set of n measurements and corresponding inputs, the set

$$\mathcal{D} = \{(\boldsymbol{s}_{xy,i},\ y_i),\ \forall i = 1, ..., n\} = \{S,\ \boldsymbol{y}\}$$

can be constructed containing all the available information about the bathymetry function. We want to predict unobserved inputs denoted by the asterisks $\boldsymbol{s}_*$ based on other inputs nearby. By definition of the GP the joint probability distribution between $\boldsymbol{y}$ and the unobserved $b(\boldsymbol{s}_*)$ is Gaussian. The joint Gaussian distribution is

$$\begin{bmatrix} \boldsymbol{y} \\ b(\boldsymbol{s}_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} K(S,\ S) + \Sigma_{\text{meas}} & K(S,\ \boldsymbol{s}_*) \\ K(\boldsymbol{s}_*,\ S) & K(\boldsymbol{s}_*,\ \boldsymbol{s}_*) \end{bmatrix} \right). \tag{3-15}$$

Where in Eq. (3-15), $K(S, S')$ is the covariance matrix obtained by evaluating the covariance function for each combination of the elements in S and S' as

$$K(S, S') = \begin{bmatrix} k(S_1,\ S'_1) & \cdots & k(S_n,\ S'_1) \\ \vdots & \ddots & \vdots \\ k(S_1,\ S'_{n'}) & \cdots & k(S_n,\ S'_{n'}) \end{bmatrix}, \tag{3-16}$$

and Σ_{meas} is a diagonal matrix with all the corresponding variances of each measurement on the diagonal.

In order to make the notation more compact, the following shorthand notation is used from now on $K = K(S, S)$, $K_* = K(S, S_*)$, and $K_{**} = K(S_*, S_*)$. Since (3-15) is jointly Gaussian, the predictive distribution of the unobserved outputs is expressed as [25, 26]

$$p\left(b\left(\boldsymbol{s}_*\right) \mid \boldsymbol{y}, S, \boldsymbol{s}_*\right) = \mathcal{N}\left(K_*^\top \left[K + \Sigma_{\text{meas}}\right]^{-1} \boldsymbol{y},\ K_{**} - K_*^\top \left[K + \Sigma_{\text{meas}}\right]^{-1} K_*\right). \tag{3-17}$$

Eq. (3-17) allows for predictions of the output of the unknown function, given dataset $\mathcal{D}$ and the covariance function $k(\boldsymbol{s}_{xy},\ \boldsymbol{s}'_{xy})$. In [1], the covariance function modeling prior bathymetry information was chosen to be a modified squared exponential (SE) with a non-zero mean function. The modified SE function was visualized in Figure 2-1 and results in sparse covariance matrices as it decays to exactly zero. This optimizes calculations as only non-zero components have to be evaluated.

As this research uses reduced rank GPs, there is no need for a sparse version of the SE covariance function. The results presented in [1] are promising; therefore we will use a similar prior for the GP, the addition between a constant covariance and a SE covariance function as

$$k(\boldsymbol{s}_{xy},\ \boldsymbol{s}'_{xy}) = \sigma^2_{\text{const}} + \sigma^2_{\text{SE}} \exp\left(-\frac{1}{2\ell^2_{\text{SE}}} \|\boldsymbol{s}_{xy} - \boldsymbol{s}'_{xy}\|^2\right). \tag{3-18}$$

The SE covariance function results in a high correlation between inputs $\boldsymbol{s}_{xy}$ and $\boldsymbol{s}'_{xy}$ that are close together. This leads that the type of functions that can be estimated is smooth and continuous. The constant covariance part allows for a non-zero mean of the bathymetry encoded in the kernel (see [25, 27]).

3-5 Measurement model

The standard form of a measurement model is

$$\hat{\boldsymbol{y}}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k^{\text{meas}}) + \boldsymbol{e}_k^{\text{meas}}. \tag{3-19}$$

Where $\hat{\boldsymbol{y}}_k$ is the prediction of the bathymetry measurements based on the current vehicle state $\boldsymbol{x}_k$ and input vector $\boldsymbol{u}_k$. The range measurement is the preprocessed bathymetry depth as described in Eq. (3-9). As we want to improve the vehicle states based on the bathymetry map, The measurement prediction is the expected value of the mapped bathymetry depth value as described in Eq. (3-14). This results in the basis of the measurement model as

$$\hat{y}_{k,j} = \mathbb{E}\left[b\left(\boldsymbol{s}_{xy,k,j}\right)\right] + e_{k,j}, \qquad e_{k,j} \sim \mathcal{N}\left(0,\ \left(\boldsymbol{d}_k \boldsymbol{\alpha}_j^b\right)^2 \sigma_r^2\right). \tag{3-20}$$

Taking the expectation of a GP map is presented in Eq. (3-17), however due to the implementation of the reduced rank GPs, the evaluation of the expected value is a approximation of Eq. (3-17). This is addressed in Chapter 4, however, the underlying model remains the same. The input $\boldsymbol{s}_{xy,k,j}$ of the map function is the location where the beam j intersects the seafloor at timestep k and is defined in Eq. (3-7), note that $\boldsymbol{s}_{xy,k,j}$ is a function of the state vector. The substitution of Eq. (3-7) in (3-20) leads to

$$\hat{y}_{k,j} = \mathbb{E}\left[b\left(\begin{bmatrix} \boldsymbol{p}_{k,x}^n \\ \boldsymbol{p}_{k,y}^n \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R(\phi_k^{nb}, \theta_k^{nb}, \psi_k^{nb}) \left(\boldsymbol{\ell}_{\text{DVL}}^b + \boldsymbol{\alpha}_j^b r_{k,j}\right]\right)\right] + e_{k,j}. \tag{3-21}$$

Note that, Eq. (3-21) is dependent on the range $r_{k,j}$ which is a stochastic variable. In order to define the measurement model in the standard form of Eq. (3-19), the problem has to be simplified by considering $r_{k,j}$ as deterministic input. By doing so, the input to the GP becomes deterministic, which significantly simplifies the problem. The standard implementation of GPs requires deterministic inputs for regression [25], GP modeling with noisy inputs requires solving integrals that have to be approximated [11]. This is not desired as it increases the computational requirements. Note that only approximately 27% of the range uncertainty is decoupled in the horizontal component. Since bathymetry surveys require downward-facing sensors, the roll and pitch angles are controlled to 0. Given the beam angles α_j^b which are stated in Table 3-1, most of the range uncertainty will be in the vertical component, which is taken into account in Eq. (3-9).

After simplifying the problem by assuming the ranges to be deterministic, Eq. (3-21) is in standard form when rewritten in vector format. By taking the expected bathymetry depth for each beam, the measurement model results in

$$
\begin{aligned}
\hat{\boldsymbol{y}}_k &= h\left(\boldsymbol{x}_k, \boldsymbol{u}_k^{\text{meas}}\right) + \boldsymbol{e}_k^{\text{meas}} \\
&= \begin{bmatrix} \mathbb{E}\left[b\left(\begin{bmatrix} \boldsymbol{p}_{k,x}^n \\ \boldsymbol{p}_{k,y}^n \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R(\phi_k^{nb}, \theta_k^{nb}, \psi_k^{nb}) \left(\boldsymbol{\ell}_{\text{DVL}}^b + \boldsymbol{\alpha}_1^b r_{k,1}\right)\right)\right] \\ \vdots \\ \mathbb{E}\left[b\left(\begin{bmatrix} \boldsymbol{p}_{k,x}^n \\ \boldsymbol{p}_{k,y}^n \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R(\phi_k^{nb}, \theta_k^{nb}, \psi_k^{nb}) \left(\boldsymbol{\ell}_{\text{DVL}}^b + \boldsymbol{\alpha}_4^b r_{k,4}\right)\right)\right] \end{bmatrix} + \boldsymbol{e}_k^{\text{meas}},
\end{aligned}
\tag{3-22}
$$

Chapter 4

Methodology

4-1 Reduced Rank approximation of GP model

A known disadvantage of using Gaussian process regression is the poor scalability with increasing dataset size n. This is due to the inversion of the covariance matrix (3-17) when evaluating test inputs, the covariance matrix K has size $n \times n$ and the time complexity of matrix inversion is $\mathcal{O}(n^3)$ [25]. This sections explains how the Gaussian process (GP) bathymetry model introduced in Section 3-4 can be approximated using the reduced rank algorithm presented in [28].

4-1-1 Covariance function approximation

For the approximation of [28] is assumed that the chosen covariance function $k(\cdot)$ is stationary, meaning it only depends on the distance between the inputs and not the absolute values of the inputs. If $k(\cdot)$ is stationary, it has an equivalent representation in terms of spectral density. This property is exploited in the reduced rank approximation of [28]. Note that the modeled covariance function in Eq. (3-18) only depends on the distance $\|\boldsymbol{s}_{xy} - \boldsymbol{s}'_{xy}\|^2$, therefore the stationary assumption holds.

The approximation method is performed on an eigenfunction expansion of the Laplace operator ∇^2 subjected to a Dirichlet boundary condition in a compact set $\Omega \subset \mathbb{R}^2$ of the input space as

$$\begin{cases} -\nabla^2 \phi_j(\boldsymbol{s}) = \lambda_j \phi_j(\boldsymbol{s}), & \boldsymbol{s} \in \Omega, \\ \phi_j(\boldsymbol{s}) = 0, & \boldsymbol{s} \in \partial\Omega. \end{cases} \tag{4-1}$$

Where in (4-1), λ_j the jth eigenvalue is and $\phi_j(\cdot)$ the eigenfunction of the Laplace operator within the given domain Ω. Using this eigendecomposition, the covariance function can be approximated as

$$k\left(\boldsymbol{s}, \boldsymbol{s}'\right) \approx \sum_{j}^{m} S\left(\lambda_j\right) \phi_j(\boldsymbol{s}) \phi_j\left(\boldsymbol{s}'\right), \tag{4-2}$$

where $S(\cdot)$ is the spectral density function of the covariance function. By truncating the series of (4-2) at $m < n$, the rank of the problem is reduced. This leads to the desired effect of improving time complexity and memory complexity. The advantages of the reduced rank approximation of [28] is that Eq. (4-2) can often be solved by evaluating closed-form expressions, and that the eigenfunctions $\phi(\cdot)$ are independent of the hyperparameters [28]. This reduced rank method has a computational cost of $\mathcal{O}(nm^2)$ and memory requirements of $\mathcal{O}(nm)$.

In order to implement the reduced rank approximation, a compact domain of the input space has to be chosen. This domain is also referred to as the tile. For the bathymetry GP model, a square tile formal is chosen centered at zero $\boldsymbol{s}_{\text{center}}^{\text{tile}} = \boldsymbol{0}$ with size of $2\ell_{\text{tile}}$ as

$$\Omega = [-\ell_{\text{tile}},\ \ell_{\text{tile}}] \cup [-\ell_{\text{tile}}, \ell_{\text{tile}}]. \tag{4-3}$$

Both input axes have the same prior covariance function modeled. Therefore a square tile format is chosen. Also, square tiles allow for fast implementation when aligned with the axes of the $\{n\}$ frame as the tiles corresponding to an input $\boldsymbol{s}_i$ can be looked up without evaluating trigonometric functions.

On the square domain, the analytical expressions for the basis functions $\phi(\cdot)$ and eigenvalues λ that solve (4-1) are [27]:

$$\begin{aligned} \phi_j(\boldsymbol{s}) &= \prod_{d=1}^{2} \frac{1}{\sqrt{\ell_{\text{tile}}}} \sin\left(\frac{\pi Z_{j,d}\left(\boldsymbol{s}_d + \ell_{\text{tile}}\right)}{2\ell_{\text{tile}}}\right), \\ \lambda_j^2 &= \sum_{d=1}^{2} \left(\frac{\pi Z_{j,d}}{2\ell_{\text{tile}}}\right)^2. \end{aligned} \tag{4-4}$$

where $Z \in \mathbb{R}^{m\times 2}$ is a matrix containing permutation integers [27] which correspond to the eigenvalues with the largest spectral densities. Note that the eigenvalues are independent of the GP inputs. Therefore they remain constant throughout the estimation. The spectral density of a two-dimensional input squared exponential (SE) covariance function is [25, 27]

$$S_{\text{SE}}(\lambda) = 2\pi\sigma_{\text{SE}}^2\ell_{\text{SE}}^2 \exp\left(-\frac{1}{2}\lambda^2\ell_{\text{SE}}^2\right). \tag{4-5}$$

In order to have an efficient approximation, $q = 5m$ eigenvalues can be selected, and the spectral density can be evaluated. Then the m largest spectral densities of the eigenvalues have to be selected of $S_{\text{SE}}(\lambda_j)$. Finally, the corresponding indices from Z that were used to obtain the m largest spectral densities have to be stored to evaluate the basis functions $\phi_j(\boldsymbol{s})$ during the approximation. In Appendix C-1 sample python code is listed for determining the m largest spectral densities and corresponding indices.

The modeled covariance function for bathymetry mapping Eq. (3-18) can be approximated as

$$\begin{aligned} k\left(\boldsymbol{s}, \boldsymbol{s}'\right) &= \sigma_{\text{const}}^2 + k_{\text{SE}}\left(\boldsymbol{s}, \boldsymbol{s}'\right) \\ &\approx \sigma_{\text{const}}^2 + \sum_{j=1}^{m} S_{\text{SE}}\left(\lambda_j\right)\phi_j(\boldsymbol{s})\phi_j\left(\boldsymbol{s}'\right). \end{aligned} \tag{4-6}$$

The approximation of the covariance function Eq. (4-6) is only valid within the chosen domain Ω Eq. (4-3). At the edge of the domain, the approximation will revert to the boundary condition Eq. (4-1). Therefore even if $m \to \infty$ Eq. (4-6) remains an approximation of the true GP. The corresponding approximation of the covariance matrix is $K(S, S') \approx \Phi(S)\Lambda\Phi(S')^\top$. Where $\Lambda \in \mathbb{R}^{m+1 \times m+1}$ is a diagonal matrix with the σ^2_{const} and m largest spectral densities of the eigenvalues on the diagonal. $\Phi \in \mathbb{R}^{n \times m+1}$ is a matrix containing the stacked eigenvectors of the points in $\mathcal{D}$ as

$$\begin{aligned}\Phi(S) &= \begin{bmatrix} 1 & \phi_1(\boldsymbol{s}_1) & \dots & \phi_m(\boldsymbol{s}_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(\boldsymbol{s}_n) & \dots & \phi_m(\boldsymbol{s}_n) \end{bmatrix}, \\ \Lambda &= \begin{bmatrix} \sigma^2_{\text{const}} & 0 & \dots & 0 \\ 0 & S_{\text{SE}}(\lambda_1) & \dots & 0 \\ \vdots & \vdots & \ddots & \dots \\ 0 & 0 & \dots & S_{\text{SE}}(\lambda_m) \end{bmatrix}. \end{aligned} \tag{4-7}$$

Using the matrix inversion lemma, the standard GP interpolation equations of the (3-17) can be rewritten to [28]

$$\begin{aligned}\mathbb{E}\left[b(\boldsymbol{s}_*)\right] &\approx \boldsymbol{\phi}_*^\top \left(\Phi^\top\Phi + \sigma_r^2\Lambda^{-1}\right)^{-1}\Phi^\top \boldsymbol{y}, \\ \mathbb{V}\left[b(\boldsymbol{s}_*)\right] &\approx \sigma_r^2\boldsymbol{\phi}_*^\top \left(\Phi^\top\Phi + \sigma_r^2\Lambda^{-1}\right)^{-1}\boldsymbol{\phi}_*. \end{aligned} \tag{4-8}$$

Where in Eq. (4-8) $\boldsymbol{\phi}_* = \phi(\boldsymbol{s}_*)$ and σ_r^2 the modeled output variance. The interpolation equations of Eq. (4-8) are well suited for the batch estimation. However, the goal is to use the reduced rank approximation in a Simultaneous Localization and Mapping (SLAM) algorithm. The map has to be updated with every new measurement, and the interpolation is part of the measurement model defined in Eq. (3-22). Therefore Eq. (4-8) is not the most efficient implementation for real-time applications.

In [27] a sequential format for real-time applications of Eq. (4-8) is presented that is similar to the Kalman filter notation where the map state vector is defined as $\boldsymbol{\mu}_i$ and the corresponding covariance defined as Σ_i. When initializing $\boldsymbol{\mu}_0 = \boldsymbol{0}$ and $\Sigma_0 = \Lambda$, the map estimated can be updated with each new observation pair of $\boldsymbol{s}_i$ and y_i as

$$\begin{aligned} q_i &= \phi\left(\boldsymbol{s}_i\right)\Sigma_{i-1}\phi\left(\boldsymbol{s}_i\right)^\top + \sigma_i^2 \\ K_i &= \frac{\Sigma_{i-1}\phi\left(\boldsymbol{s}_i\right)^\top}{q_i} \\ \boldsymbol{\mu}_i &= \boldsymbol{\mu}_{i-1} + K_i\left(\boldsymbol{y}_i - \phi\left(\boldsymbol{s}_i\right)\boldsymbol{\mu}_{i-1}\right) \\ \Sigma_i &= \Sigma_{i-1} - q_i K_i K_i^\top \end{aligned} \tag{4-9}$$

After the sequential update, predictions for unseen input $\boldsymbol{s}_*$ can be made by taking into account all observations up to measurement i, and the approximate predictive distribution with the sequential implementation is

$$\begin{aligned}\mathbb{E}\left[b\left(\boldsymbol{s}_*\right)\right] &\approx \phi(\boldsymbol{s}_*)\boldsymbol{\mu}_i, \\ \mathbb{V}\left[b\left(\boldsymbol{s}_*\right)\right] &\approx \phi(\boldsymbol{s}_*)\Sigma_i\phi(\boldsymbol{s}_*)^\top. \end{aligned} \tag{4-10}$$

The computational complexity of the sequential implementation is $\mathcal{O}(m^3)$ for each update. This has to be updated with each measurement. Therefore, the total computational complexity is $\mathcal{O}(nm^3)$. The memory requirements are reduced to $\mathcal{O}(m^2)$ as only the Σ_i matrix and $\boldsymbol{\mu}_i$ vector has to be stored. The resulting expected bathymetry function value of Eq. (4-10) can be directly substituted into the measurement model defined in Section 3-5 which leads to

$$\begin{aligned}\hat{\boldsymbol{y}}_k &= h\left(\boldsymbol{x}_k, \boldsymbol{u}_k^{\text{meas}}\right) + \boldsymbol{e}_k^{\text{meas}} \\ &= \begin{bmatrix} \phi(\left(\begin{bmatrix} \boldsymbol{p}_{k,x}^n \\ \boldsymbol{p}_{k,y}^n \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R(\phi_k^{nb}, \theta_k^{nb}, \psi_k^{nb}) \left(\boldsymbol{\ell}_{\text{DVL}}^b + \boldsymbol{\alpha}_1^b r_{k,1}\right)\right) \boldsymbol{\mu}_k \\ \vdots \\ \phi\left(\begin{bmatrix} \boldsymbol{p}_{k,x}^n \\ \boldsymbol{p}_{k,y}^n \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R(\phi_k^{nb}, \theta_k^{nb}, \psi_k^{nb}) \left(\boldsymbol{\ell}_{\text{DVL}}^b + \boldsymbol{\alpha}_4^b r_{k,4}\right)\right) \boldsymbol{\mu}_k \end{bmatrix} + \boldsymbol{e}_k^{\text{meas}}. \end{aligned} \tag{4-11}$$

Where $\boldsymbol{\mu}_k$ in Eq. (4-11) is the map state vector at time instance k that will be estimated in SLAM simultaneously with the vehicle states. Eq. (4-11) can be efficiently evaluated since it makes use of reduced rank GPs and the $\phi(\cdot)$ is a function known in closed form as defined in Eq. (4-4).

4-1-2 Tile layout over input domain

The GP input domain of interest can be large in underwater applications and unknown at the start of the algorithm. Note that the reduced rank approximation is defined on a square subset Eq. (4-1) of the GP input domain and is zero outside the domain. One solution would be to define a single large tile covering the entire seafloor space which the Autonomous Underwater Vehicle (AUV) is expected to enter. However, this is not desirable as a large domain requires many basis functions m in order to get good approximations of the GP model, which defeats the purpose of using an approximation in the first place. Another solution is to divide the seafloor input domain into multiple independent tiles with $\boldsymbol{s}_{\text{center}}^{\text{tile}} \neq \boldsymbol{0}$ that each estimate the bathymetry. Since the used covariance function is stationary, meaning that $k(\cdot,\cdot)$ only depends on the distance between inputs, the origin of the inputs may be freely translated as the distance remains the same. Therefore each tile has it's own GP approximation with the corresponding $\boldsymbol{\mu}_k$ and Σ_k, however the Λ matrix and the $\phi(\cdot)$ function are the same. This means we can divide the whole input space into independent tiles, which can be initialized if required.

For a tile to approximate the dense GP, the effects caused by the zero boundary condition Eq. (4-1) are undesirable as the dense GP is not subjected to these effects. To reduce these effects, adjacent tiles overlap, which is visualized in Figure 4-1.

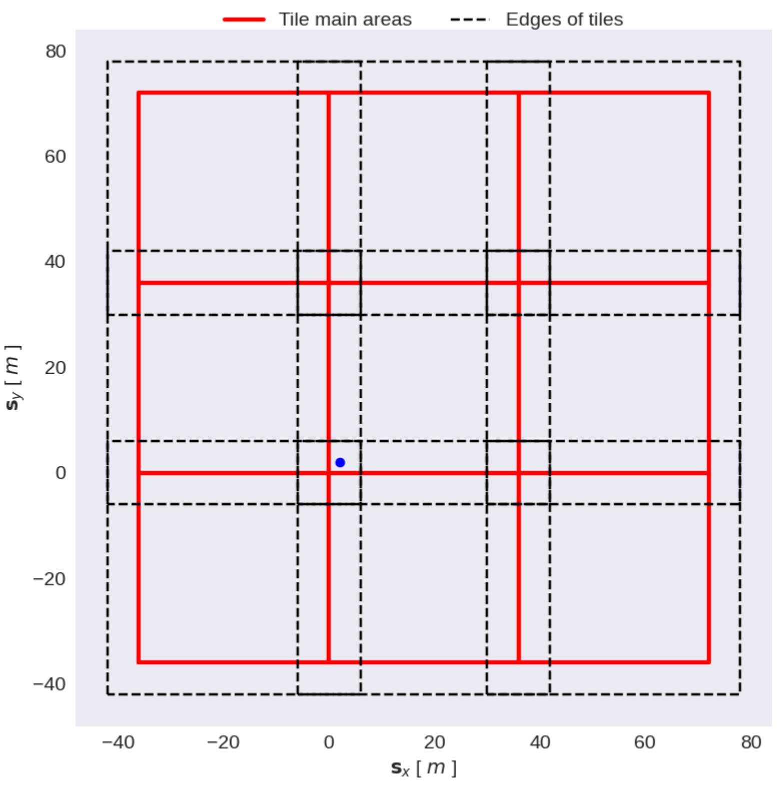


Figure 4-1: Illustration of how 9 tiles are placed in such a way to have no overlap between the main tile areas. The blue point represents a location of an observation.

The red main tile areas in Figure 4-1 indicate the area where the tile has a good approximation of the dense GP. The overlapping distance is called ℓ_{overlap} and is the distance between the edge of the tile and the edge of the main tile area. In order to produce a good transition, $\frac{2}{3}$ of the ℓ_{overlap} is being used for updating the tile if a measurement is observed in this area. The last $\frac{1}{3}$ of the ℓ_{overlap} is used as transition area from the GP approximation to the boundary condition. If the blue point in Figure 4-1 would represent the seafloor point $\boldsymbol{s}_{xy}$ where a measurement intersects with the seafloor, 4 tiles would have to be updated using Eq. (4-9). When the estimated bathymetry value of the blue point is desired, only the corresponding main tile has to be evaluated.

Variable	Size	Description
$\boldsymbol{s}^{\text{tile}}_{\text{center}}$	$\mathbb{R}^2$	Center location of tile for shifting inputs.
$\boldsymbol{\mu}$	$\mathbb{R}^{m+1}$	State vector of GP in tile.
Σ	$\mathbb{R}^{m+1 \times m+1}$	Covariance matrix of GP in tile.

Table 4-1: Variables that are initialized with each new tile.

Each tile has only to keep track of a few properties, which are listed in Table 4-1. The tiles can share all other parameters, such as tile dimensions and the initial Λ matrix. The required memory for each unique tile is, therefore only a function of m as

$$\text{RequiredBytes} = 4 \cdot \left(m^2 + 3m + 5\right). \tag{4-12}$$

Where in Eq. (4-12) the number of variables is multiplied by 4 since there are 4 bytes in a single precision floating point number. Note that Eq. (4-12) is not optimized to take into account the symmetry of the Σ matrix. Therefore the memory requirements could potentially be halved. For example, by choosing $m = 256$ each extra tile requires 0.27 Mb of memory.

Algorithm 1 contains the pseudo-code required for generating a bathymetry map using the reduced rank approximated GPs.

Algorithm 1 Bathymetry map estimation pseudo code

```
Given dataset $\mathcal{D}$, $\ell_{\text{tile}}$, $\ell_{\text{overlap}}$, $m$, $\ell_{\text{SE}}$, $\sigma_{\text{SE}}$ and $\sigma_{\text{const}}$.

 1: Calculate $\Lambda_0$ and $Z$ from Eq. (4-1) and (4-5) using $\ell_{\text{tile}}$, $m$, $\ell_{\text{SE}}$, $\sigma_{\text{SE}}$ and $\sigma_{\text{const}}$; for code
    example see Appendix C-1.

 2: Set of initialized tiles: $tiles = \emptyset$
 3: for each element of $\{\boldsymbol{s}_i, y_i, \sigma_i\} \in \mathcal{D}$ do
 4:     Find $activeTiles$ based on input location $\boldsymbol{s}_i$, $\ell_{\text{tile}}$, and $\ell_{\text{overlap}}$
 5:     for $tile \in activeTiles$ do
 6:         if $tile \notin tiles$ then
 7:             Initialize $tile$ with $\boldsymbol{\mu}_0 = \mathbf{0}$, $\Sigma_0 = \Lambda$.
 8:             Extend $tiles = \{tiles,\ tile\}$
 9:         end if
10:         Update local GP estimate of $tile$ using sequential estimation of Eq. (4-9) as
                $\boldsymbol{\phi} = \phi\left(\boldsymbol{s}_i - \boldsymbol{s}^{\text{tile}}_{\text{center}}\right)$
                $q_i = \boldsymbol{\phi}\Sigma_{i-1}\boldsymbol{\phi}^\top + \sigma_i^2$
                $K_i = \frac{\Sigma_{i-1}\boldsymbol{\phi}^\top}{q_i}$
                $\boldsymbol{\mu}_i = \boldsymbol{\mu}_{i-1} + K_i\left(y_i - \boldsymbol{\phi}\boldsymbol{\mu}_{i-1}\right)$
                $\Sigma_i = \Sigma_{i-1} - q_i K_i K_i^\top$
11:     end for
12: end for

13: for $tile \in tiles$ do
14:     Interpolate main area of $tile$ using Eq. (4-10)
15: end for
16: Return joined interpolation values of each tile to create a bathymetry map.
```

4-2 Bathymetry GP SLAM algorithm

In order to truly improve the operational capabilities of underwater vehicles, we want to estimate the map and the vehicle's position simultaneously. The previous section described how a seafloor map using reduced rank GPs could be made. The algorithm in the previous section assumed known $\boldsymbol{s}_i$ GP input points for each measurement. This requires knowledge about the vehicle's state. This section describes the methodology of the SLAM algorithm.

As the name suggest, SLAM estimates both the vehicle state and the bathymetry map state. In Section 3-1 the uncertain pose of the vehicle was denoted by $\boldsymbol{x}_k$ and contained the horizontal position and the heading angle. In the previous section, we saw how the GP model is estimated by a Kalman filter with state vector $\boldsymbol{\mu}_k$ for each tile d. The joint SLAM state vector is thus

$$\boldsymbol{x}_k^{\text{SLAM}} = \begin{bmatrix} \boldsymbol{x}_k \\ \{\boldsymbol{\mu}_k\}_{d=1}^{N_{\text{tile},k}} \end{bmatrix}. \tag{4-13}$$

Where in Eq. (4-13), $N_{\text{tile},k}$ is the number of tiles initialized up to time instance k. In Section 3-2, the dynamic model of the vehicle states was introduced, and recall that in Section 3-4 we placed the assumption that the bathymetry has no time-related dynamics. Only the estimates of μ_k change each time step as more observations are available. The $\boldsymbol{x}_k^{\text{SLAM}}$ state vector will be split into the map state vector and the vehicle state vector for the remainder of this report as

$$\boldsymbol{x}_k = \begin{bmatrix} p_{x,0:k}^n \\ p_{y,0:k}^n \\ \psi_{0:k}^{nb} \end{bmatrix}, \qquad \boldsymbol{\eta}_k = \{\boldsymbol{\mu}_k\}_{d=1}^{N_{\text{tile},k}}. \tag{4-14}$$

Where $\boldsymbol{\eta}_k$ is defined for notational brevity as the growing in size, combined tile state vector. The joint probability distribution that we aim to estimate is the probability distribution of the current position and the distribution of the bathymetry as

$$p\left(\boldsymbol{x}_k,\ \boldsymbol{\eta}_k \mid Y_{0:k},\ U_{0:k},\ \boldsymbol{x}_0\right). \tag{4-15}$$

There are two main methods of estimating $Eq.\ (4-15)$ [7], EKF based SLAM and particle filter SLAM. The Extended Kalman Filter (EKF) approximates the joint state $\boldsymbol{x}_k^{\text{SLAM}}$ with a Gaussian distribution. Therefore it cannot accurately represent the joint state distribution in situations where the terrain is locally similar at different locations. This leads to multi-modal probability distributions, which can not be represented accurately with a Gaussian distribution. For this reason, SLAM based on particle filtering will be used. A particle filter has a stochastic adaptive grid of N_p evaluation points, also called the particles. The particles are chosen in such a way that they represent the relevant part of the state space. Each particle i has an corresponding state variable $\boldsymbol{x}_k^{\text{SLAM},i}$, and an associated importance weight w_k^i that indicate the relative accuracy of $\boldsymbol{x}_k^{\text{SLAM},i}$ compared to the other particles states.

The joint SLAM state vector contains the bathymetry state $\boldsymbol{\eta}_k$ that has the dimension of $mN_{\text{tile},k}$. As m is of high dimension, typically $m > 50$, the joint state vector is of high dimension. It is computationally unfeasible to sample the joint state $\boldsymbol{x}_k^{\text{SLAM}}$ as the required particles to accurately represent the state space scale exponential with the state dimension [7]. Rao-Blackwellization of the join SLAM state vector reduced the sample space by rewriting a joint probability distribution into two components using the conditioning rule as

$$p(x_1, x_2) = p(x_2 \mid x_1)p(x_1). \tag{4-16}$$

In case that $p(x_2 \mid x_1)$ can be solved analytically, the joint sample space of $p(x_1, x_2)$ can be reduced by only sampling $p(x_1)$ and using the analytical relation to represent the joint distribution. This process is called Rao-Blackwellization of the sample space [7]. This sample space reduction is applied in the Rao Blackwellized particle filter (RBPF), which is also known as FastSLAM and is key for computational viability [7].

Eq. (4-15) can also be conditioned into two components. Note that in Section 4-1, an analytical algorithm was introduced which estimates $\boldsymbol{\eta}_k$ and requires as an input $\boldsymbol{s}_i$ which is the point where the beam intersects the seafloor in the $\{n\}$ frame. This depends on the vehicle state, and this relation is described in Eq. (3-7). Due to the the analytical formulas of Eq. (4-9), it is possible to partition the probability distribution of Eq. (4-15) as

$$p\left(X_{0:k},\ \boldsymbol{\eta}_k \mid Y_{0:k}, U_{0:k}, \boldsymbol{x}_0\right) = p\left(\boldsymbol{\eta}_k \mid X_{0:k}, Y_{0:k}, U_{0:k}\right) p\left(X_{0:k} \mid Y_{0:k}, U_{0:k}, \boldsymbol{x}_0\right). \tag{4-17}$$

Where in Eq. (4-17), the estimation has to be extended to depend on the whole vehicle trajectory $X_{0:k} = \{\boldsymbol{x}_0, ..., \boldsymbol{x}_k\}$ instead of the single pose $\boldsymbol{x}_k$. The conditional probability $p\left(\boldsymbol{\mu}_k \mid X_{0:k} Y_{0:k}, U_{0:k}\right)$ represents the bathymetry map given the AUV states, this estimation problem is extensively discussed in Chapter 4-1 and can be estimated analytically using the Kalman filter updates of Eq. (4-9). When using the partitioning of Eq. (4-17), the whole trajectory $X_{0:k}$ has to be sampled. This can be sampled recursively as

$$\begin{aligned} p\left(X_{0:k} \mid Y_{0:k}, U_{0:k}, \boldsymbol{x}_0\right) &= p\left(X_{0:k} \mid \cdot\right) \\ &= p\left(\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \mid \cdot\right) \\ &= \boldsymbol{x}_0 p\left(\boldsymbol{x}_1, \mid \boldsymbol{x}_0, \cdot\right) \ldots p\left(\boldsymbol{x}_k, \mid X_{0:k-1}, \cdot\right) \end{aligned} \tag{4-18}$$

At each time step, the samples are drawn from a proposal distribution $q(\cdot)$ which approximates the probability distribution $p\left(\boldsymbol{x}_k, \mid X_{0:k-1}, Y_{0:k}, U_{0:k}, \boldsymbol{x}_0\right)$. An AUV typically has accurate state predictions due to the accurate velocity measurements. Therefore a prior sampling particle filter is chosen as it performs well for the case where the state prediction is accurate, as stated in [12]. The proposal distribution of the prior sampling particle filter is the dynamical model presented in Eq. (3-4) as

$$\begin{aligned} q(\boldsymbol{x}_k \mid X_{0:k-1}, Y_{0:k}, U_{0:k}, \boldsymbol{x}_0) &= p(\boldsymbol{x}_k \mid \boldsymbol{x}^i_{k-1}, \boldsymbol{u}_k), \\ &\sim f(\boldsymbol{x}^i_{k-1},\ \boldsymbol{u}^{\text{dyn}}_{k-1},\ \boldsymbol{e}^{\text{dyn}}_{k-1}). \end{aligned} \tag{4-19}$$

The sampling of the dynamic model results in the random behavior of the particle states. Given the proposal distribution of (4-19), the importance weight is then updated with the probability of the current measurement given the state and map as

$$\tilde{w}^i_k \propto w^i_{k-1} p(\boldsymbol{y}_k \mid \boldsymbol{x}^i_k, \boldsymbol{\eta}^i_k). \tag{4-20}$$

Where in Eq. (4-20), the $\tilde{w}_k^i$ represents the unnormalized weights, and have to be normalized in order for the samples to approximate the true probability distribution of Eq. (4-15). Weight normalization is performed as

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{i=1}^{N_p} \tilde{w}_k^i}, \qquad i = 1, \ldots, N_p. \tag{4-21}$$

The probability $p(\boldsymbol{y}_k \mid \boldsymbol{x}_k^i, \boldsymbol{\eta}_k^i)$ indicates the likelihood of a bathymetry measurement. The measurement model of Eq. (4-11) calculates an expected measurement $\hat{\boldsymbol{y}}_k$ given the map for each Doppler Velocity Log (DVL) beam as $\hat{\boldsymbol{y}}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k^{\text{meas}}) + \boldsymbol{e}_k^{\text{meas}}$. Since each of the beams is assumed to be statistically independent and subjected to Gaussian noise, the probability of a measurement given the bathymetry can be evaluated as [1]

$$\begin{aligned} p(\boldsymbol{y}_k \mid \boldsymbol{x}_k^i, \boldsymbol{\eta}_k) &= p\left(\boldsymbol{y}_k - h\left(\boldsymbol{x}_k^i, \boldsymbol{u}_k^{\text{meas}}\right) = 0\right), \\ &= \prod_{j=1}^{4} \frac{1}{\sqrt{2\pi\left(\sigma_r^2 + \sigma_{\text{map},j}^2\right)}} \exp\left(-\frac{\left(y_{k,j} - \hat{y}_{k,j}\right)^2}{2\left(\sigma_r^2 + \sigma_{\text{map},j}^2\right)}\right). \end{aligned} \tag{4-22}$$

In Eq. (4-22), $\sigma_{\text{map},j}^2 = \mathbb{V}\left[b\left(\boldsymbol{s}_{k,j}\right)\right]$ obtained as part of the map interpolation in Eq. (4-10). As a validation step, a deterministic grid of particles with evenly spaced positions was created. The probability for each particle was calculated using Eq. (4-22) for a certain bathymetry measurement $\boldsymbol{y}_k$ and bathymetry map $\boldsymbol{\eta}_N$ and is shown in Figure 4-2. The bathymetry map $\boldsymbol{\eta}_N$ is created using Algorithm 1 using the integration of the dynamical model up to $k = N$. The selected measurement and the map are taken from the same dataset. Therefore the true position $p_{k,\text{true}}^n$ of the measurement is known. $p_{k,\text{true}}^n$ is indicated in Figure 4-2 as the red dot on the bathymetry map. The area around $p_{k,\text{true}}^n$ also has a high value evaluated probability of Eq. (4-22).

In Figure 4-2, it can be seen that there is a whole area of locations where the corresponding probability is high. This is caused since the DVL has limited beams and sensing accuracy, and the seafloor has similar characteristics in different areas of the map. Therefore a Gaussian distribution would be a poor approximation of Figure 4-2. This is the motivation for using the particle filters as it does not rely on Gaussian assumptions of the probability distributions.

A crucial part of the particle filter is to resample the current particle set for a new particle set. This is because the likely particles have a large weight and a higher chance of being selected than the particles with low relative weight. The resample step results in good coverage of particles in the state space in the most probable region. During the resampling step, copies are created from particles with a uniform weight of $w_k^i = \frac{1}{N_p}$. This also involves copying the bathymetry map and other variables, which are stated in Table 4-2. The resampling can be implemented in numerous ways. In Section C-2, a code example of how to implement the resampling process is presented. The particles are resampled when the number of effective particles $N_{p,\text{eff}}$ is below a threshold. An effective particle is a particle whose weight is significantly above zero. $N_{p,\text{eff}}$ can be approximated as [12]

$$N_{p,\text{eff}} \approx \frac{1}{\sum_{i=1}^{N_p} (w_k^i)^2}. \tag{4-23}$$

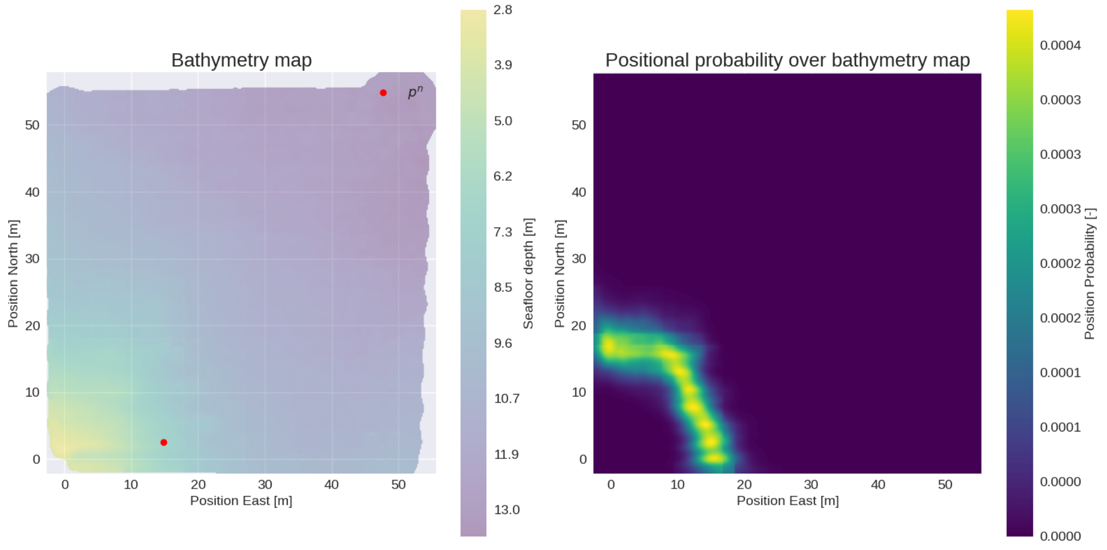


Figure 4-2: Evaluation of probability metric over whole bathymetry map. On the left-hand side the corresponding true position $p^n_{k,\text{true}}$ is indicated with the red dot.

Resampling takes place whenever the following condition is true

$$N_{p,\text{eff}} < c_{\text{resample}} N_p, \tag{4-24}$$

where $0 < c_{\text{resample}} < 1$ is the resample threshold.

Variable	Size	Description
w^i_k	$\mathbb{R}$	Relative importance weight
$X_{0:k}$	$\mathbb{R}^{3\times k+1}$	Current and past vehicle trajectory
$\{\boldsymbol{\mu}^i_k\}_{d=1}^{N_{\text{tile},k}}$	$\mathbb{R}^{(m+1)\times N_{\text{tile},k}}$	State of bathymetry for each tile
$\{\Sigma^i_k\}_{d=1}^{N_{\text{tile},k}}$	$\mathbb{R}^{(m+1)\times(m+1)\times N_{\text{tile},k}}$	Covariance matrix of bathymetry for each tile

Table 4-2: Local variables for each particle.

In Algorithm 2, the reduced rank bathymetry RBPF is described in detail. The RBPF can is summarized in 4 main steps given a particle set [7, 12]:

1. Propagate particle states x^i_k trough dynamic model of Eq. (3-4) and sample the motion noise to introduce randomness.
2. Update each particle's weights using Eq. (4-11), based on current measurement $\boldsymbol{y}_k$.
3. Perform resampling if $N_{p,\text{eff}} < c_{\text{resample}} N_p$.
4. Update each particle's map estimate by executing Eq. (4-9) for each tile in each map.

Algorithm 2 Bathymetry SLAM algorithm

```
    Given $N_p$, $\boldsymbol{x}_0$, $c_{\text{resample}}$, $\sigma_v$, $\sigma_\psi$, $\sigma_r$, $\ell_{\text{tile}}$, $\ell_{\text{overlap}}$, $m$, $\Lambda$, $Z$
 1: Initialize $N_p$ particles with $\boldsymbol{x}_0^i = \boldsymbol{x}_0$ and $w_0^i = \frac{1}{N_p}$

 2: $X_0^* = \{\}$                                   ▷ Best state estimates at each time step
 3: $M_0^* = \{\}$                                   ▷ Best map estimates at each time step
 4: $k = 0$
 5: while $k < N$ do
 6:     $k = k + 1$
 7:     for each particle $i \in N_p$ do
 8:         Propagate $\boldsymbol{x}_{k-1}^i$ trough the motion model to obtain $\boldsymbol{x}_k^i$ (3-4).
 9:         Store $\boldsymbol{x}_k^i$ in the particles state history $X_k^i = \{X_{k-1}^i, \boldsymbol{x}_k^i\}$.

10:         for each beam $j \in \{1, 2, 3, 4\}$ do
11:             Calculate $\boldsymbol{s}_{k,j,*}$ based on $\boldsymbol{u}_k^{\text{meas}}$ and $\boldsymbol{x}_k^i$ using Eq. (3-7).
12:             Find tile based on $\boldsymbol{s}_{k,j,*}$ and extract $\boldsymbol{\mu}_k^i$ and $\Sigma_k^i$.
13:             Calculate $\hat{y}_{k,j} = \phi(\boldsymbol{s}_{k,j,*})\boldsymbol{\mu}_k^i$ and $\sigma^2_{\text{map},j} = \phi(\boldsymbol{s}_{k,j,*})\Sigma_k^i\phi(\boldsymbol{s}_{k,j,*})^\top$
14:         end for
15:         Update $w_k^i$ by using Eq. (4-20) and (4-22).
16:     end for

17:     Normalize particle weight for each particle (4-21).
18:     Resample if $N_{p,\text{eff}} < c_{\text{resample}}N_p$ based on weight of each particle.

19:     for each particle $i \in N_p$ do
20:         Update bathymetry estimate $\boldsymbol{\eta}_k^i$ and $\{\Sigma_k^i\}_d^{N_{\text{tile},k}}$ using Algorithm 1
21:     end for

22:     Find the particle with the largest weight $i^* = \text{argmax}_i \left(\left\{w_N^0, ..., w_N^i, ...w_N^{Np}\right\}\right)$.
23:     Append current best state and map as $X_k^* = \{X_{k-1}^*,\ \boldsymbol{x}_k^{i^*}\}$, $M_k^* = \{M_{k-1}^*,\ \boldsymbol{\eta}_k^{i^*}\}$
24: end while

25: Return SLAM state trajectory $X_N^*$ and the observed bathymetry $M_N^*$ as output.
```

Chapter 5

Results

5-1 Used datasets

During each mission of the Scout Autonomous Underwater Vehicle (AUV), all raw sensor data and the Extended Kalman Filter (EKF) fused state estimates are logged in a dataset. Figure 5-1 shows a diagram of the state estimation and control of the AUV. The Doppler Velocity Log (DVL) measurement message contains the body velocity and range measurements of each beam. The velocity measurement is used in the EKF, while the range information is not taken into account.

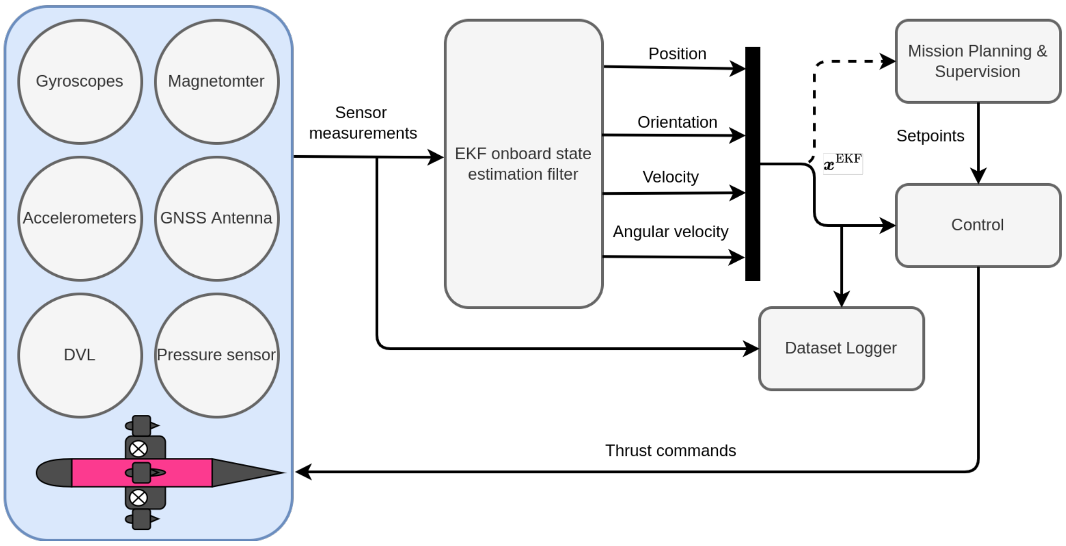


Figure 5-1: Data flow diagram during operation of Scout missions.

Two specially designed missions were conducted for the experiments in this report. The first is called Dataset 1 and is a larger lawnmower mission of 50 × 50 meters at a constant altitude of 3 meters. The lawn velocity was $0.5\frac{m}{s}$ and the total mission took around an hour to complete. As the Scout has been underwater, there are no independent position measurements from the Global Navigation Satellite System (GNSS) antenna for this mission. The second mission is referred to as Dataset 2 and is a smaller lawnmower mission at the surface with GNSS reference measurements. This mission was challenging to execute since the Scout is less reliable when operating at the surface as air gets sucked into the thrusters. As a consequence, the thrusters produce unreliable thrusts. Dataset 2 is the best dataset with GNSS reference measurements.

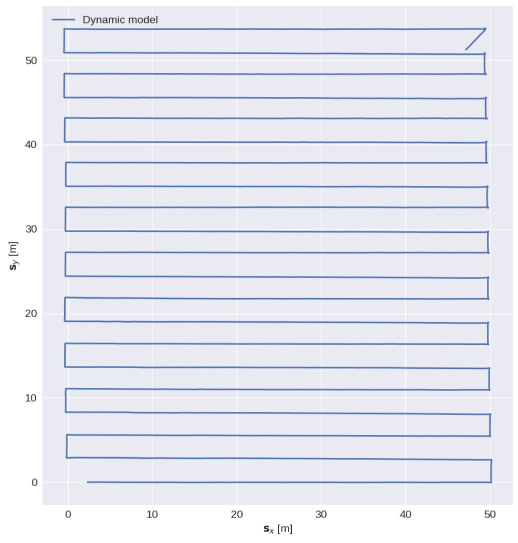
Apart from the Scout datasets, the open-source AURORA dataset is used to test the reduced rank mapping algorithm. This dataset is two orders of magnitude larger in terms of range measurements than the other datasets [2]. In the Aurora dataset, the range measurements were captured with a multi-beam echo sounder. The multi-beam's range measurements are similar to range measurements of the DVL only the multi-beam captures with each ping up to 520 measurements. Unfortunately, this dataset was unsuitable for Simultaneous Localization and Mapping (SLAM) experiments since the beams suffer from some significant inaccuracies due to ray bending in the overlap area. When ray bending occurs, the assumption that each acoustic beam can be modeled with a straight vector no longer holds. Specific data post-processing is required to eliminate the effects, which was not possible during this research. Key properties of the datasets are summarized in Table 5-1, and the simulation of the dynamical model of dataset 1 and dataset 2 are visualized in Figure 5-2.

Dataset name	**Distance traveled**	**Number of range measurements**	**Other relevant information**
Dataset 1	1.1 km	$6.9 \cdot 10^4$	Large lawnmower pattern
Dataset 2	0.17 km	$1.5 \cdot 10^4$	Surface mission with GNSS measurements as reference
AURORA	25 km	$7.9 \cdot 10^6$	Range measurements collected with a multi-beam

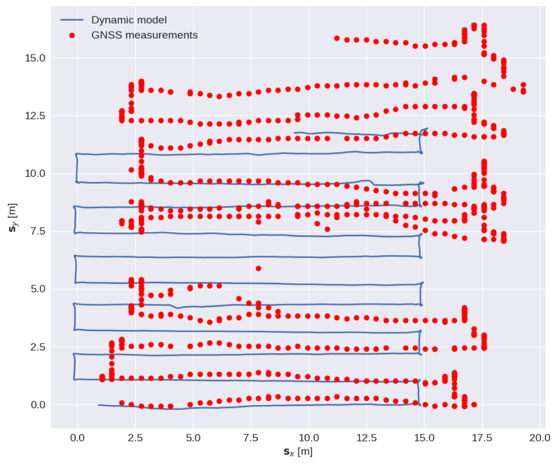
Table 5-1: The different datasets used in this report with some key properties of each dataset.

The simulated trajectory in Figure 5-2 is obtained using the dynamical model Eq. (3-4), which approaches the $\boldsymbol{x}^{\text{EKF}}$ estimates. The simulated trajectory of both datasets looks structured and does not seems to contain any drift. However, note that $\boldsymbol{x}^{\text{EKF}}$ is being used as input to the control algorithm, and the robot is controlled to follow the mission plan (which is a typical lawnmower pattern). Due to this closed-loop setup, the estimated states $\boldsymbol{x}^{\text{EKF}}$ are similar to the mission plan. However, in reality the AUV drifts away from the mission plan. This can be seen in Figure 5-2b where the GNSS points are measurements from the actually traveled trajectory. Since the used GNSS antenna is a basic model, the obtained measurements contain some noticeable noise. Also, there is one large jump in the measured position, likely caused by the loss of connection with one satellite. Nevertheless, the GNSS measurements more accurately indicate the actual trajectory and can be treated as a reference.

From the simulations of the dynamical model in Figure 5-2, one can see there is no overlap in the trajectory of the dynamic model. However, due to the combination of DVL beam angles, lawn spacing, and altitude, there is an overlap in the points $\boldsymbol{s}_{xy}$ where the beams intersect the seafloor.



(a) Dataset 1



(b) Dataset 2

Figure 5-2: Visualization of the traveled trajectory of the two datasets, dataset 2 is executed at the surface with GNSS measurements.

5-2 Reduced rank mapping algorithm

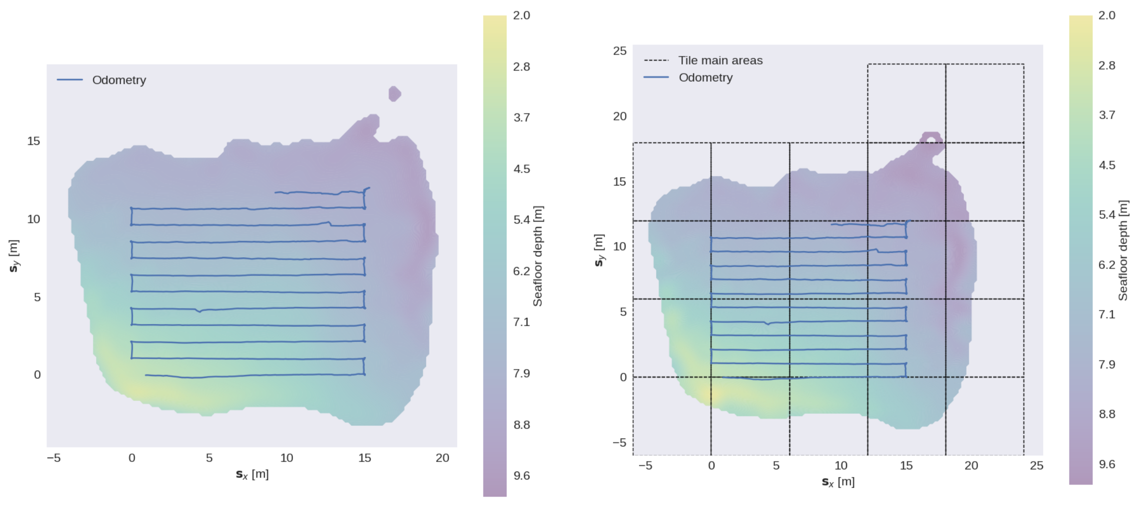
In bathymetry mapping algorithm using reduced rank Gaussian processes (GPs) was described in Algorithm 1. The underlying GP bathymetry model is discussed in Section 3-4. In this section, results of the mapping algorithm are demonstrated and compared against the dense version of the GP model. Throughout this section, the EKF state $\boldsymbol{x}_k^{\text{EKF}}$ of the log file will be assumed to be deterministic, including the position. This is needed to calculate the corresponding intersection point $\boldsymbol{s}_{k,j}$ for each range measurement.

When running algorithm 1 with fixed hyperparameters of $\ell_{\text{SE}} = 1.0$, $\sigma_{\text{SE}} = 1.0$ and $\sigma_{\text{const}} = 10.0$. The resulting bathymetry plots are visualized in Figure 5-3. Full page copies of these images are placed in Section D.

Algorithm	Computation time	Total memory required
Dense GP	420 sec	3.8 Gb
Reduced rank GP $m = 256$	2.0 sec	70 Mb

Table 5-2: Computational improvements of using approximate GPs over dense GP using dataset 2 with $1.5 \cdot 10^4$ bathymetry measurements.

In Table 5-2, the computational improvements are stated of using the reduced rank approximation over the dense GP implementation. In the SLAM algorithm, each particle estimates a local version of the bathymetry map. A particle filter can require 100 or more particles to represent the posterior distribution accurately. Therefore the map estimation will quickly become a bottleneck for achieving real-time performance. Apart from computational performance, the accuracy of the approximation is as least as important. To



(a) Dense GP implementation

(b) Reduced rank GP implementation with $m = 256$ basis functions

Figure 5-3: Comparison of the dense GP bathymetry map and the reduced rank approximation. The map is generated using the odometry trajectory as deterministic input to the mapping algorithm.

measure the accuracy of the reduced rank GP map, the map predictions are compared to the map predictions of the dense GP, and a root mean squared error (RMSE) value is calculated. The RMSE values are visualized over a range of number of basis functions m in Figure 5-4, and we see that the approximation approaches the dense GP model. These results indicate that the reduced rank GP approximation of the model presented in Section 3-4, significantly improves computational performance while having only a slightly worse prediction accuracy.

The mapping algorithm scales well to larger datasets such as Dataset 1 and the Aurora dataset. As a dense GP algorithm is incapable of processing these datasets due to the number of measurements, no reference map is available for comparison. As the resulting bathymetry maps are primarily an illustration of the algorithm, the figures are placed in Section D. The used constants stated in Table 5-3. The parameters of Table 5-3 are chosen based on the visual consistency of the produced bathymetry map. The tile size of the Aurora dataset is chosen to be larger because of memory constraints. As the surveyed area is roughly 800×5000 meters, the required amount of tiles became problematic.

Variable	ℓ_{SE}	σ_{SE}	σ_{const}	m	σ_r	ℓ_{tile}	ℓ_{overlap}
Dataset 1	1.0	1.0	10.0	256	0.2	6.25 m	2.5 m
Aurora dataset	3.0	1.0	10.0	256	0.2	22.5 m	7.5 m

Table 5-3: Constants used for the mapping of the other datasets.

Although the obtained computational performance depends significantly on the implementation, it is the primary motivation for using reduced rank GP approximation. The results presented in Table 5-4 are obtained with an implementation in C++ of the reduced rank GPs run on a single core of a modern notebook. In the next section, the

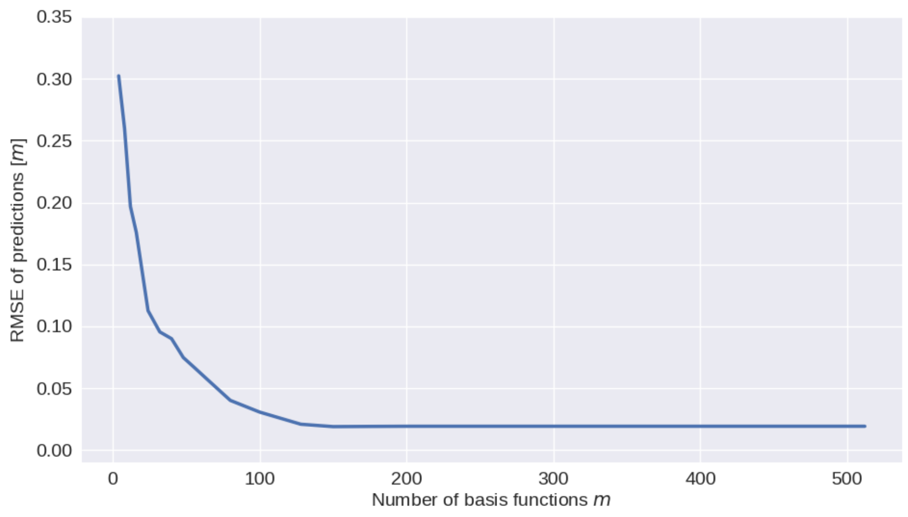


Figure 5-4: RMSE in meters between dense GP predictions and reduced rank GP over varying number of basis functions m.

influence of some of these parameters on the localization performance is evaluated.

Dataset	Number of tiles	Computation time	Total memory required	Figure number
Dataset 1	120	7 sec	102 Mb	D-3
Aurora	5116	801 sec	2.0 Gb	D-5

Table 5-4: Computational costs of Algorithm 1, run on a notebook single core C^{++} implementation.

5-3 Position improvement using a known map

In this section, results are reported of using the SLAM Algorithm 2 to estimate the vehicle trajectory and using a previously determined bathymetry map as input to the algorithm. When the bathymetry map is used as input, each particle uses this as a map, and the map estimation step is skipped. This results in an Algorithm which is also known as localization. The reason for wanting a bathymetry map as input is that when noisy odometry data is simulated as input to the particle filter, the output should converge to the odometry trajectory used for generating the map. Therefore the original odometry trajectory can be used as a reference trajectory and can be used to make conclusions about the filter's performance. The experimental setup is visualized in Figure 5-5, where the map generation uses the reference trajectory for determining the beam intersection points and running Algorithm 1.

The noisy odometry data $\tilde{\boldsymbol{v}}^b_{1:N}$ and $\Delta\tilde{\psi}_{1:N}$, is created by corrupting the dataset's odometry

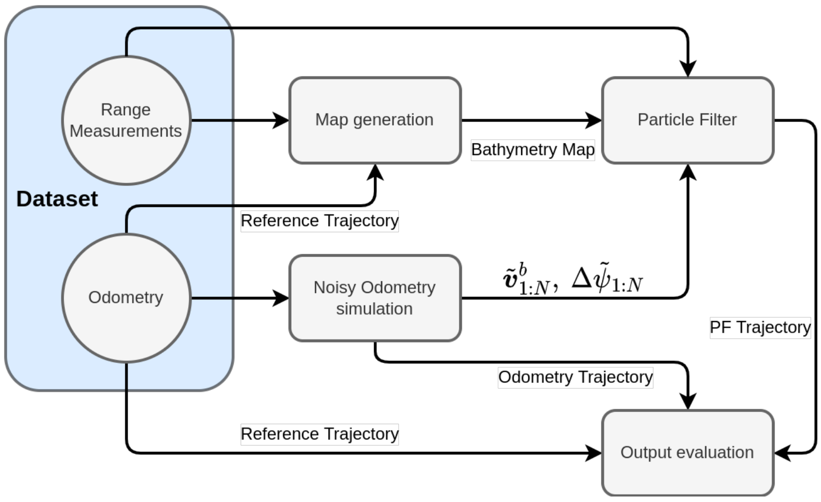


Figure 5-5: Simulation setup localization experiments from the logged dataset.

data with additional additive Gaussian noise, as

$$\begin{aligned} \tilde{\boldsymbol{v}}^b_k &= \boldsymbol{v}^b_k + \tilde{\boldsymbol{e}}_{\boldsymbol{v},k}, & \tilde{\boldsymbol{e}}_{\boldsymbol{v},k} &\sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{\tilde{v}} I_3), \\ \Delta\tilde{\psi}_k &= \Delta\psi_k + \tilde{e}_{\Delta\psi,k}, & \tilde{e}_{\Delta\psi,k} &\sim \mathcal{N}(0, \sigma^2_{\Delta\tilde{\psi}}). \end{aligned} \tag{5-1}$$

Based on observations from missions, the odometry uncertainty that results in a realistic simulated odometry errors are chosen to be $\sigma_{\tilde{v}} = 0.05 \ \frac{m}{s}$ and $\sigma_{\Delta\tilde{\psi}} = 0.002$ rad. All the simulations of the particle filter in this section are performed using the same noisy odometry data as input. Figure 5-6 shows the noisy odometry trajectory obtained by simulating the motion model using the noisy odometry inputs. The noisy odometry is subsequently used as an input to the dynamical model in the particle filter. The only knowledge the particle filter gets about the reference trajectory is via the bathymetry map, as the reference trajectory is used to generate the input bathymetry map.

In practical applications, localization on a known map is functional whenever bathymetry data is available. A reduced rank GP bathymetry map can be estimated from the available data. We may therefore assume that whenever the AUV starts a mission, it has a GNSS lock before submerging underwater. The initial particle spread can thus be limited to the area around the GNSS lock, and the initial heading may be determined with the magnetometer as is done in the EKF filter. Note that this heading estimate is subjected to magnetic field distortions of the AUV itself. To take this uncertainty into account, the initial heading distribution is modeled as a uniform distribution centered around ψ_0^{EKF}. The initial particle state distribution is chosen as:

$$\begin{aligned} p(\boldsymbol{p}^n_{0,x}) &= \mathcal{N}(\boldsymbol{p}^{n,\text{EKF}}_{0,x},\ 4) \\ p(\boldsymbol{p}^n_{0,y}) &= \mathcal{N}(\boldsymbol{p}^{n,\text{EKF}}_{0,y},\ 4) \\ p(\psi_0) &= \mathcal{U}(\psi_0^{\text{EKF}} - 10^\circ,\ \psi_0^{\text{EKF}} + 10^\circ) \end{aligned} \tag{5-2}$$

The initial position uncertainty values of (5-2) are chosen based on the accuracy of the GNSS signals. The initial heading uncertainty is based on tests' observations and accuracy reporting in the publication of [18]. Due to computing time constraints, all simulations are done with 100 particles unless otherwise specified.

5-3-1 Demonstration of particle filter

This section investigates the resulting trajectory of the particle filter for a specific set of parameter choices. The choice of these parameters is to a large extent arbitrary. The chosen parameters are listed in Table 5-5. In later experiments, these variables are varied and there will be shown that the particle filter converges for a large range of values.

Variable	σ_r	c_{resample}	ℓ_{SE}	σ_{SE}	σ_{const}	m	ℓ_{tile}	ℓ_{overlap}	σ_ψ	σ_v
Value	0.4	0.6	1.5	1.0	10.0	256	9.0	3.0	0.004	0.1

Table 5-5: Parameters used for demonstration of the particle filter.

The particle filter, as described in Algorithm 2, uses the information from the bathymetry map to evaluate the incoming measurements based on map predictions. This way, the position estimates are improved over the simulated odometry. Figure 5-6 shows an overview with the bathymetry map, the simulated odometry trajectory (indicated as odometry), the reference trajectory, and the particle filter estimated trajectory. The main difference between the odometry and the particle filter is, that the particle filter evaluates the incoming range measurements against the bathymetry map. The estimated trajectory from the particle filter is close to the reference trajectory. This indicates that the bathymetry data contains helpful information for underwater position estimation.

In Figure 5-7, the error of the position and heading estimates with respect to the reference trajectory are shown over time. It can be seen that the filter is also able to correct the drift in the heading estimates of the odometry. However, the heading estimates are noisy, which can be improved by lowering the motion noise in the particle filter. Nevertheless, the heading estimates correspond better to the reference than the odometry. Only at the beginning of the mission the odometry has a better position and heading estimates. This is due to similar terrain in the initial spread of the particles Eq. (5-2), after the initial wrong belief, the particle filter output does converge towards the reference trajectory. This shows that by using bathymetry as an information source, both the position and heading estimates of underwater vehicles can be improved.

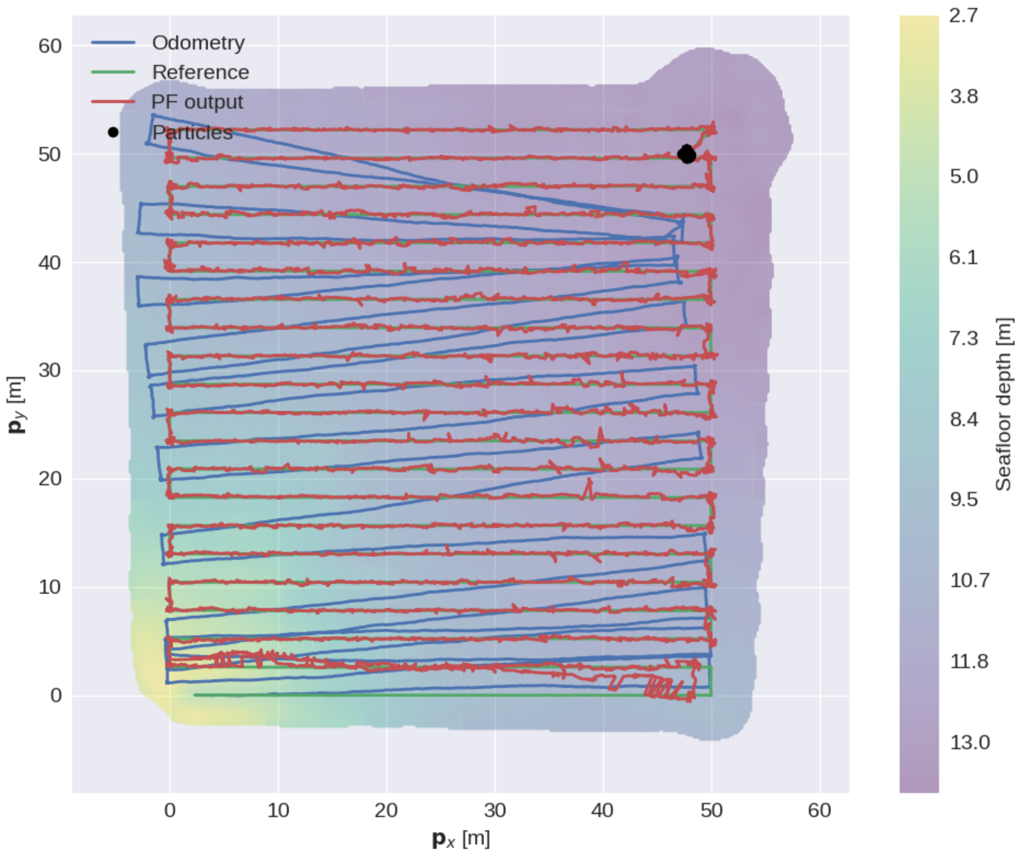


Figure 5-6: Output trajectory of the particle filter, odometry, and reference trajectory. After an initial error due to locally similar terrain, the trajectory converged towards the reference trajectory.

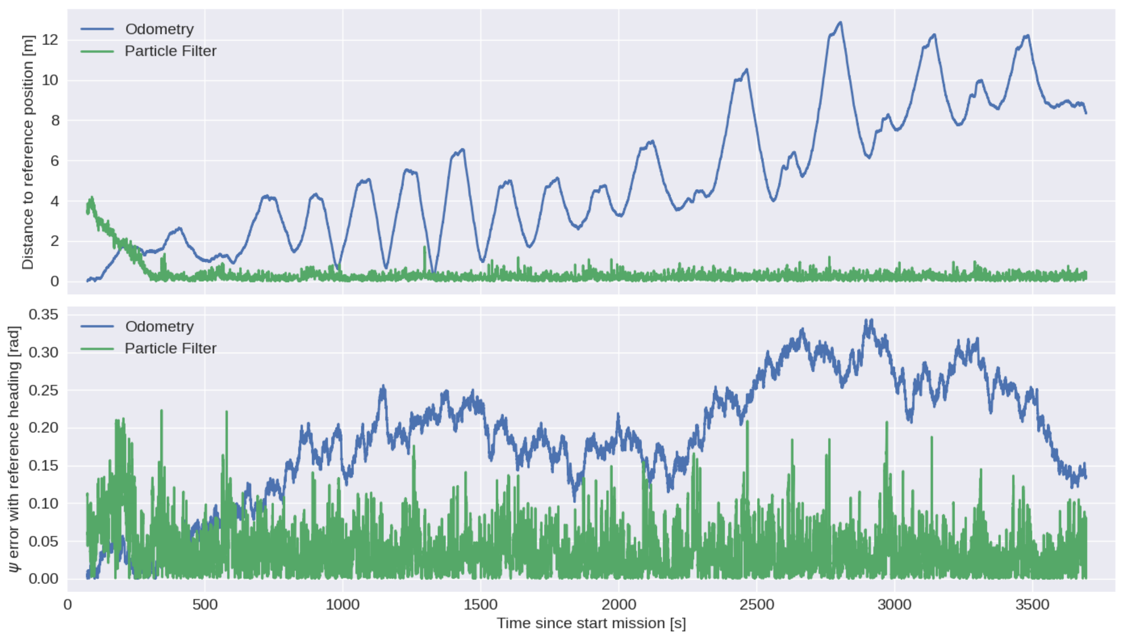


Figure 5-7: Distance of odometry and particle filter trajectories compared to the reference trajectory.

5-3-2 Varying the motion noise

In this section, the motion model noise levels of the particle filter are varied to see the influence on the filter's performance. The positions of the output trajectory from the particle are consequently evaluated against the reference positions, and the resulting RMSE of the last half of the trajectory is chosen as an evaluation metric. Figure 5-6 showed there could be an initial wrong belief about the AUV's position. Therefore, only the last half is taken into account; otherwise, the initial wrong belief will dominate the RMSE values. The initial belief depends on the initial particle distribution. The initial particle spread is a random sample over quite a large area, many simulations would be required to get a meaningful average. Except for the motion noise parameters, all other used parameters are the same as stated in Table 5-5. Table 5-6 shows the resulting last half RMSE values of the particle filters trajectory averages over 5 independent runs.

	RMSE LH				
σ_ψ [rad]	$\sigma_v = 0.01\ [\frac{m}{s}]$	$\sigma_v = 0.02\ [\frac{m}{s}]$	$\sigma_v = 0.04\ [\frac{m}{s}]$	$\sigma_v = 0.08\ [\frac{m}{s}]$	$\sigma_v = 0.16\ [\frac{m}{s}]$
0.000	4.00	3.17	1.94	2.56	1.82
0.001	0.428	0.576	0.144	0.168	0.220
0.002	0.554	0.186	0.142	0.170	0.218
0.004	0.157	0.142	0.146	0.173	0.224
0.008	0.158	0.158	0.156	0.185	0.229
0.016	0.187	0.173	2.354	0.200	0.245

Table 5-6: RMSE of the last half of the particle filter's output position, compared to the reference position.

Due to the lack of high-quality gyroscopes, the odometry heading angle has to be determined using the magnetic field. This introduces heading errors due to magnetic field distortions of the AUV. As these errors can be dominant in the position estimation, in Chapter 3 was argued that the heading should be added to the sampled states. From the results in Table 5-6 it is clear that sampling the heading ($\sigma_\psi \neq 0$) improves the estimation accuracy significantly when such a heading uncertainty is present, as is the case in the simulated odometry.

More generally, there seems to be an optimum present for the used motion uncertainties. A too low uncertainty will restrict the particles from deviating from the odometry, and a too high uncertainty will cause the sample spread to be larger, resulting in a lower probability that a sample is near the reference trajectory. Large uncertainties can even cause the filter to diverge, as happens once for the run configuration of $\sigma_\psi = 0.016$ and $\sigma_v = 0.04$, this has caused the RMSE value to be significantly larger other runs.

5-3-3 Varying the map parameters

In this section, the simulations of the previous section are repeated, only now the map hyperparameters and number of basis functions are varied while the remaining parameters are equal to the reported values in Table 5-5. As this thesis aims to investigate how reduced rank Gaussian processes that represent the bathymetry can be used for improved position estimation of underwater vehicles, the relation between hyperparameters and filter performance is particularly interesting.

The map parameters that are varied are the hyperparameters ℓ_{SE}, σ_{SE} and σ_{const}, number of basis functions m and range expected standard deviation level σ_r. The resulting RMSE of the last half of the mission are stated in Table 5-7.

hyperparameters values				**RMSE LH**		
l_{SE}	σ_{SE}	σ_{const}	σ_r	$m = 128$	$m = 256$	$m = 512$
1.0	1.0	1.0	0.2	0.169	0.124	0.119
1.0	1.0	1.0	0.4	0.161	0.124	0.122
1.0	1.0	1.0	0.8	0.159	0.139	0.138
1.0	1.0	10.0	0.2	0.165	0.123	0.117
1.0	1.0	10.0	0.4	0.159	0.125	0.120
1.0	1.0	10.0	0.8	0.159	0.137	0.139
1.0	2.0	1.0	0.2	0.162	0.119	0.110
1.0	2.0	1.0	0.4	0.156	0.118	0.114
1.0	2.0	1.0	0.8	0.160	0.131	0.129
1.0	2.0	10.0	0.2	0.165	0.117	0.109
1.0	2.0	10.0	0.4	0.157	0.117	0.115
1.0	2.0	10.0	0.8	0.157	0.131	0.129
2.0	1.0	1.0	0.2	0.265	0.258	0.266
2.0	1.0	1.0	0.4	0.275	0.273	0.270
2.0	1.0	1.0	0.8	0.280	0.278	0.277
2.0	1.0	10.0	0.2	0.259	0.262	0.264
2.0	1.0	10.0	0.4	0.278	0.270	0.271
2.0	1.0	10.0	0.8	0.272	0.273	0.273
2.0	2.0	1.0	0.2	0.241	0.240	0.248
2.0	2.0	1.0	0.4	0.241	0.242	0.241
2.0	2.0	1.0	0.8	0.250	0.251	0.250
2.0	2.0	10.0	0.2	0.242	0.238	0.241
2.0	2.0	10.0	0.4	0.242	0.247	0.243
2.0	2.0	10.0	0.8	0.240	0.244	0.245

Table 5-7: RMSE values of the last half for the given hyperparameters and number of basis functions. RMSE values indicate the accuracy of the best particle trajectory compared to the reference trajectory. All RMSE values are averaged over five runs.

From the results presented in Table 5-7 some interesting observations can be made. Firstly, the hyperparameter σ_{const} seems to have little influence on the position accuracy performance. Secondly, the length scale seems to be the most influential on the filter performance, the RMSE approximately halves when choosing $\ell_{\text{SE}} = 1.0$ relative to $\ell_{\text{SE}} = 2.0$. However, we must be careful about this observation as both the particle filter and map generation are based on the same range measurements, as illustrated in Figure 5-5. Therefore the particle filter could be over-fitted to the measurements. A third observation is that the number of basis functions m used in the reduced rank approximation has more influence on the filter performance for the lower length scale. This could be explained since a smaller length scale allows for a more detailed map. Thus, more basis functions are required to represent the map and provide improved interpolation values. The increase in basis functions could have reduced the GP approximation errors, leading to better interpolation values that improve the positioning. For $\ell_{\text{SE}} = 2.0$ the increase in basis functions does not lead to better positioning performance for the given tile size. This is an important observation as the number of basis functions is one of the main drivers of the required computation time and memory.

5-3-4 Varying particle filter tuning

Finally, the particle filters main tunings parameters c_{resample} and σ_r are varied over multiple values. c_{resample} determines how often the filter resamples are based on the spread between particle weights. A higher value results in more often resampling. σ_r determines the extent to which the current observation is considered relative to the weight at the previous time. A higher value of σ_r means that the particle weights change more gradually and thus that the spread of particles is more extensive before resampling.

Simulations are run with varying values of c_{resample} and σ_r. As the particle filter is stochastic, each parameter configuration is repeated 10 times. The average of 10 runs of RMSE values over varying particle filter tuning parameters are stated in Table 5-8.

	RMSE LH			
σ_r $[m]$	$c_{\text{resample}} = 0.5$	$c_{\text{resample}} = 0.6$	$c_{\text{resample}} = 0.7$	$c_{\text{resample}} = 0.8$
0.1	0.178	0.182	0.182	0.182
0.2	0.175	0.180	0.185	0.190
0.4	0.182	0.185	0.195	0.203
0.8	0.191	0.196	0.204	0.213
1.6	0.262	0.272	0.268	0.292
3.2	0.507	0.470	0.478	0.524
6.4	0.756	0.804	0.945	0.943

Table 5-8: Root Mean Square Values of the last half (LH) of the particle filters trajectory compared to the reference trajectory. Values are averaged over 10 simulations.

The primary observed trend over different tuning parameters is that an increased noise level leads to worse position estimation, given a known map. The resample threshold does not seem to affect positional accuracy significantly.

5-4 SLAM results

Dataset 2 is collected by operating the AUV at the surface of a lake. Therefore GNSS measurements are collected by the AUV are measurements of its actual trajectory throughout the mission. These GNSS measurements serve as reference trajectory. The trajectory estimated by the SLAM algorithm can thus be compared against the reference trajectory. The GNSS measurements have an assumed standard deviation of $\sigma_textGNSS = 2$ meter when the measurements are transformed to the $\{n\}$ frame using the methods in [10]. Both for the odometry trajectory and the SLAM estimated trajectory, a

RMSE can be calculated compared to the GNSS measurements. A decrease in RMSE is expected for SLAM estimated trajectory as it considers the bathymetry information.

The SLAM algorithm presented in Section 4-2 is set up with the initial position and heading from the dataset. For online algorithm implementations, the initial position can be determined from an initial GNSS position and the heading from the fusion of the gyroscope and magnetometer. The other parameters used are stated in Table 5-9. A high noise value σ_r was found to produce the best results as the amount of resamples was reduced, and thus more candidate trajectories were kept. Based on the results of Table 5-7, the number of basis functions was set to 256 to ensure sufficient detail in the map.

Variable	N_p	c_{resample}	σ_v	σ_ψ	σ_r	ℓ_{tile}	ℓ_{overlap}	m	ℓ_{SE}	σ_{SE}	σ_{const}
Value	200	0.7	0.02	0.001	2.0	9.0	3.0	256	1.0	1.0	10.0

Table 5-9: Used parameters for running the SLAM algorithm.

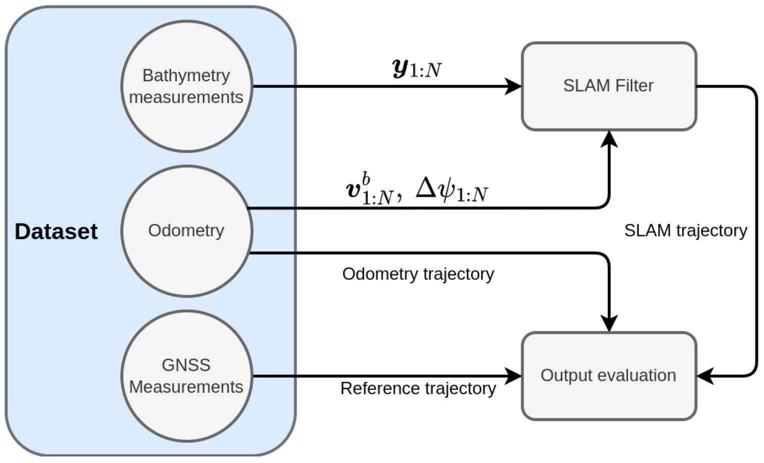


Figure 5-8: Experimental setup using real AUV data SLAM experiments.

Algorithm 2 is simulated on Dataset 2, the mission stored in Dataset 2 took 900 seconds to execute and consists of 4700 DVL messages of 4 beams each. The average sample rate is thus 5 Hz. The resulting final trajectory is shown in Figure 5-9, and the evaluation RMSE metric is stated in Table 5-10 for the SLAM estimated trajectory and the odometry integrated trajectory. The RMSE metric has been reduced by 31% compared to the odometry trajectory.

	Odometry trajectory	**SLAM trajectory**
RMSE	1.9 m	1.3 m

Table 5-10: SLAM output trajectory compared to odometry as measured by the evaluation metric.

The required computation time and memory of the SLAM algorithm is stated in Table 5-11. These experiments have been conducted on a modern notebook using the setup of Table 5-9. The SLAM algorithm runs approximately 5 times faster than real-time using this setup. When running the algorithm with $m = 128$, the algorithm is 29 times faster than real-time. Besides the number of basis functions, another critical factor is the number of particles N_p, which

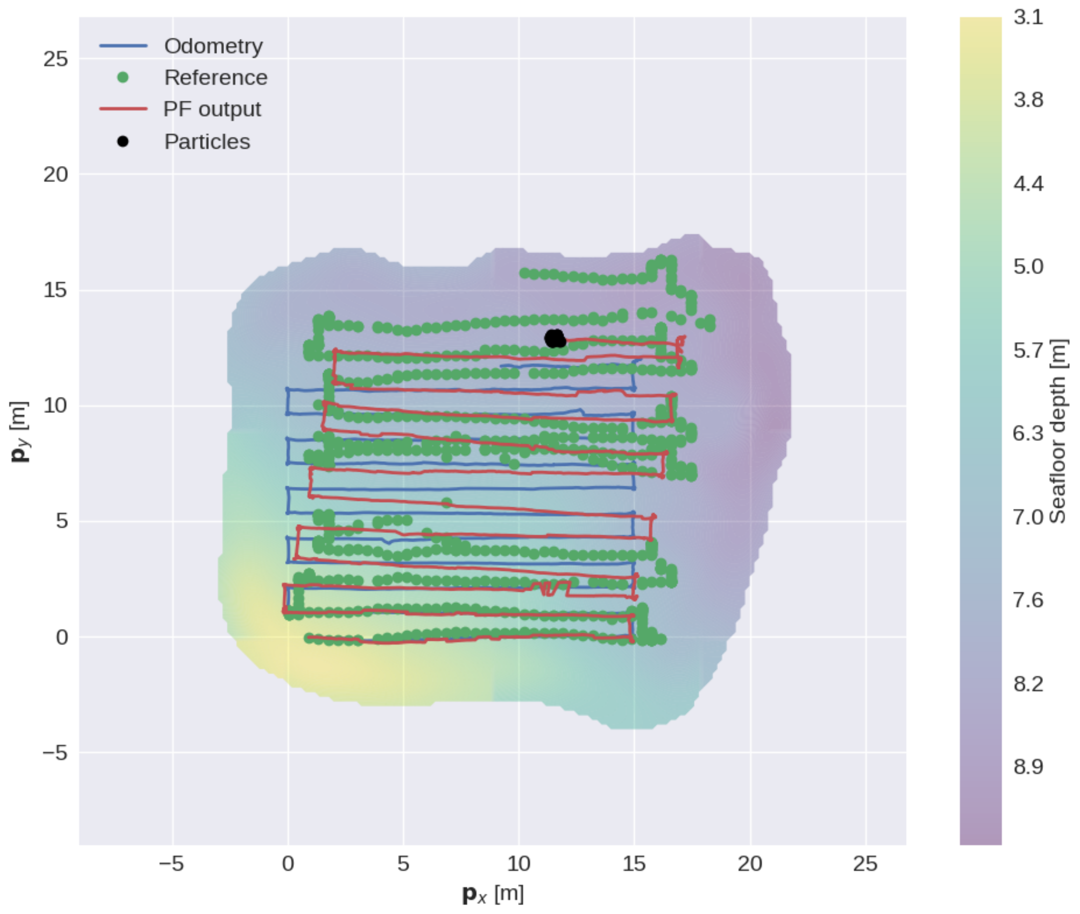
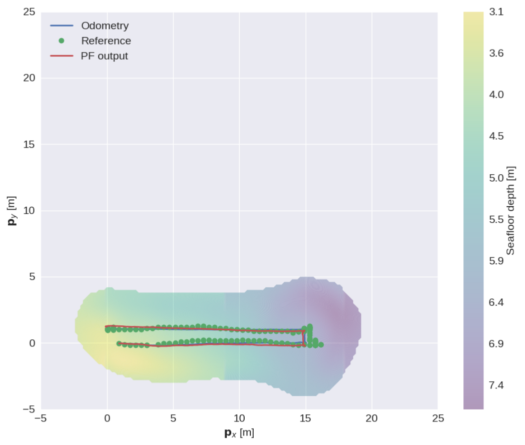


Figure 5-9: Resulting best trajectory and map of SLAM algorithm visualized against the GNSS measurements and odometry trajectory.

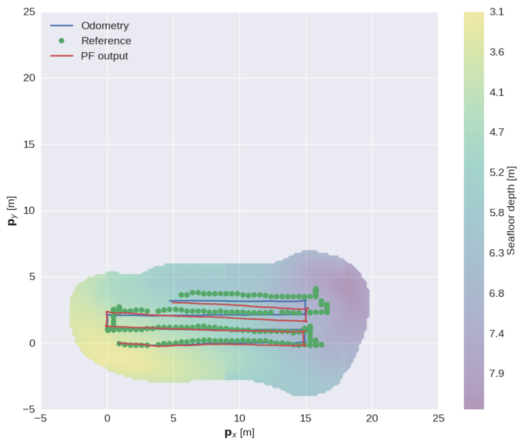
influences these parameters approximately linear. When taking half the number of particles, the run time and the memory required decrease by 50%. In Figure 5-10, intermediate outputs of the particle filter are shown. Both the bathymetry and the trajectory are visualized for six intermediate steps.

Number of basis functions	**Computation time**	**Total memory required**	**Real-time factor**
$m = 128$	31.2 sec	254 Mb	28.8
$m = 256$	171.1 sec	560 Mb	5.2

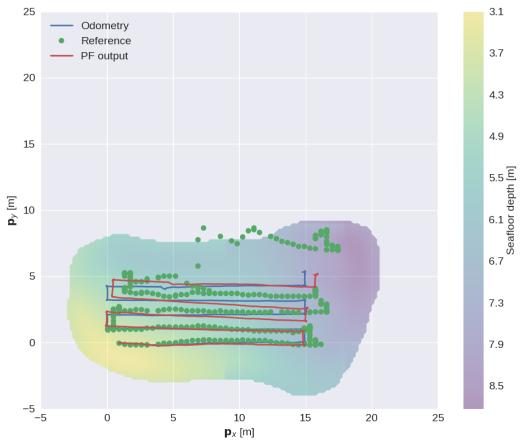
Table 5-11: Computational costs of Algorithm 2, run multi-threaded a notebook with C^{++} implementation with 200 particles.



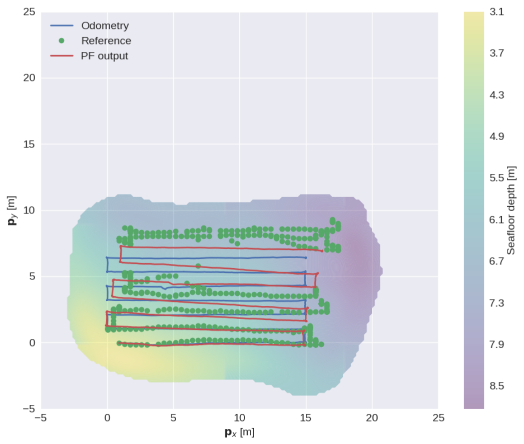
(a) SLAM output at $k = 700$



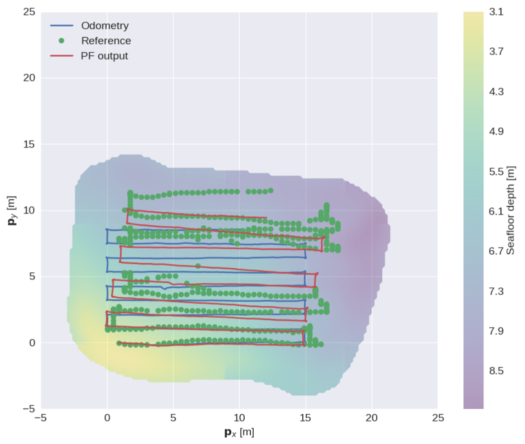
(b) SLAM output at $k = 1400$



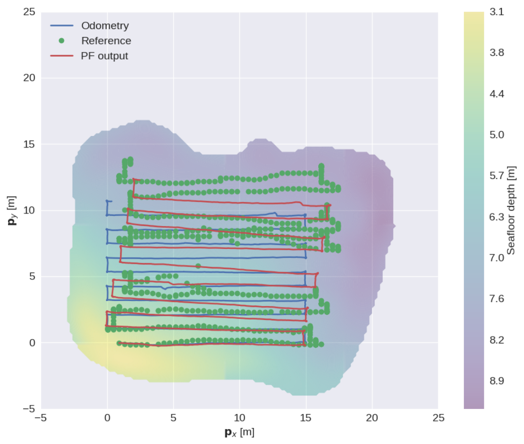
(c) SLAM output at $k = 2100$



(d) SLAM output at $k = 2800$



(e) SLAM output at $k = 3500$



(f) SLAM output at $k = 4200$

Figure 5-10: Intermediate outputs of the SLAM algorithm.

Chapter 6

Discussion & Conclusions

6-1 Discussion

As stated in the introduction, the main identified problem is that the current way of accurate underwater positioning relies on acoustic beacons or existing detailed bathymetry maps. Both methods pose operational requirements for the ocean exploration mission, contributing to the high cost of underwater data gathering. Bathymetry Simultaneous Localization and Mapping (SLAM) could lower the threshold for underwater data gathering, as it has been shown to improve the position estimates without topside communication or existing bathymetry maps [23].

This research focused on a bathymetry map representation that has not been used before in underwater position estimation, to the author's best knowledge. Significant computational performance is gained by approximating Gaussian Processes with the reduced rank approximation of [28]. The reduced rank approximation has been applied in other fields, such as magnetic field mapping and SLAM [16, 27], and precipitation level mapping [28]. The research question posed in this report is how the computational efficient reduced rank bathymetry map representation could be used in a SLAM algorithm for position estimation of the Scout Autonomous Underwater Vehicle (AUV). The SLAM algorithm was tested on experimental data from the Scout. The main sensors used for the position estimation are a Doppler Velocity Log (DVL) and a micro-electro-mechanical systems (MEMS) gyroscope.

The key results presented in this report are that the reduced rank approximation can yield significant computational improvements while only suffering from a slight accuracy loss compared to the dense Gaussian process (GP). Also, there was shown how the reduced rank approximation can be consequently used to evaluate the incoming measurements. Finally, these two elements were combined in Rao Blackwellized particle filter (RBPF) and tested on a dataset of the Scout with Global Navigation Satellite System (GNSS) measurements as the reference position. The SLAM algorithm estimated a trajectory closer to the GNSS measurements.

Due to the implementation of reduced rank GPs, it is possible to store the bathymetry information efficiently. This differs from the SLAM algorithm of [1], where all measurements were stored in a central log that each particle has access to save memory. Instead, a local GP map is created with a selection of measurements based on the distance to the current measurement. The reduced rank implementation might be more straightforward as the evaluation of measurements can be selected based on the uncertainty of the map to prevent extrapolating information. The GP model in [1] of the bathymetry contained a constant mean function which was occasionally updated by averaging the bathymetry measurements. In this report, the mean function is not necessary as the GP model includes a constant covariance (4-6). This allows for non-zero means away from the measurement points. As the non-zero mean is taken care of automatically, the constant covariance model appears to be the favorable approach.

A 3D occupancy gridmap based bathymetry SLAM is presented in [8]. This map representation differs as it does not put uniqueness assumptions on the seafloor terrain. Therefore a 3D occupancy grid map is well suited to navigate an underwater vehicle in a vertical water shaft, as is shown in this research. The primary drawback of the 3D occupancy grid maps is the cubic memory requirement. The required memory was improved by utilizing smart sharing of regions between particles, however it remains a cubic requirement. Al tough the use case is slightly different, using reduced rank Gaussian processes, the memory requirements scale quadratically, as each tile has a required memory of $\mathcal{O}(m^2)$ and the required tiles increase quadratically with the mission area.

The reduced rank GP approximation of the bathymetry model contributes to the existing publications, as it enables memory and time optimization while it is a non-discretized map representation. There was shown that the model can be implemented efficiently in a RBPF algorithm as the position estimates improved and the RBPF runs 30 to 5 times faster than real-time, depending on the parameters. The resulting accuracy improvement is hard to compare against other publications as it depends on the alignment of sensors, sensor quality, and terrain variability. Therefore no meaningful comparisons can be made based on the estimation accuracy results presented in this thesis.

The limitation of this research is that the SLAM algorithm is tested on a single dataset. There exist a chance that the results presented are not an accurate representation of the behavior of the algorithm. Nevertheless, the proposed SLAM algorithm does produce an increased positional accuracy over the dead-reckoned odometry trajectory. Another practical aspect is that the proposed method is sensitive to outliers; too many outliers could cause the algorithm to diverge. Therefore the outlier rejection of range measurements should be done before running the algorithm. For the results in this thesis, information about the entire dataset was used to remove the outliers and was manually checked, which is not possible for online applications. Any bathymetry SLAM depends on the significance of bathymetric variations in the observed terrain. Therefore the algorithm should be tested on various seafloor terrains to assess the robustness of the proposed algorithm. On a completely flat seafloor, the bathymetry will not provide any information at all. For real-world applications, the seafloor variations could be checked by learning the GP hyperparameters from a selection of the data and making decisions based on the length scale, similarly as proposed in [1].

6-2 Research Questions

Before answering the main research question, as stated in the introduction, the sub-questions will be answered first.

Sub-question 1

> How can the bathymetry be estimated using reduced rank GP regression assumed with known position, and which sources of errors are present in the map?

In Chapter 3 the models for processing range measurements to bathymetry measurements and the GP prior model were presented. The reduced rank approximation of the GP model is discussed in Section 4-1, and a mapping algorithm is presented in Algorithm 1. In order to execute the mapping algorithm, several assumptions were made about the inputs. If these assumptions do not hold, the estimated bathymetry will be prone to errors compared to the real world. The known AUV pose assumption is an obvious source of error. The pose is necessary for converting range measurements to bathymetry points Eq. (3-7). Part of the pose is subsequently estimated with the SLAM algorithm that aims to lower this error by learning from the map. Other sources of error are related to the determination of the measured range. These include the incorrect speed of sound, varying speed of sound over the traveled acoustic path, and bending of acoustic paths.

Sub-question 2

> How can the bathymetry map be used to improve the position estimate, and how influential are the GP hyperparameters on the position estimation accuracy?

In Section 4-2 is shown how knowledge about the bathymetry map can be used for evaluating the likelihood of measurements. This is realized by a particle filter where each particle has a local estimate of the current position and heading. At each intersection point of a DVL beam, the map is evaluated. Since all uncertainties are assumed to be Gaussian, a probability that indicates the particle's likelihood can be evaluated. Simulations based on experimental data from a AUV show that the AUV trajectory consistently converged towards the reference trajectory used for map generation. Numerous simulations were run with different sets of hyperparameters. These experiments indicate that the most influential on the performance is the length scale of the squared exponential (SE) covariance function. Another important parameter is the number of basis functions used in the approximation. More basis functions can lead to better position estimation Table 5-7, at the cost of increased computational complexity.

Sub-question 3

> Is a reduced rank GP SLAM algorithm viable to run on an underwater vehicle in real-time?

The algorithm's speed mainly depends on the number of basis functions and particles. By using 200 particles and 128 or 256 basis functions, the computation time is 30 and 5 times faster than real-time respectively, on a modern notebook. As underwater vehicles can have similar computing performance, running this SLAM implementation should be viable onboard the Scout or other underwater vehicles. In order to reach this performance level, an implementation in a compiled programming language like C or C^{++} was used, and the number of copied variables during resampling must be kept to a minimum.

Main research question

The main research question posed is

> How can reduced rank GP bathymetry SLAM with MEMS gyroscopes and DVL as main sensors be used to improve the position estimation of underwater vehicles?

In Chapters 3 and 4 of this report the models and methods of a reduced rank approximation of GP bathymetry SLAM were presented. The underlying algorithm for SLAM is a RBPF which samples the vehicle states. A particle filter is chosen as the bathymetry can be locally similar and thus lead to multi-modal state distributions. In Chapter 5 was shown that the algorithm could reduce position estimation errors relative to using the odometry model referenced against GNSS measurements. The computational requirements are modest, so it can potentially improve position estimation in real-time on AUV.

6-3 Recommendations

In future research, it might be interesting to research some of the following aspects of bathymetry SLAM using reduced rank GPs:

- There is a large influence of the GP hyperparameters on the prediction quality, as seen in Table 5-7. Therefore it would be interesting to research how to handle terrain variations in the GP prior. As the underwater vehicle enters unknown terrain in a SLAM algorithm, setting the hyperparameters fixed prior to the mission might not be the best solution. Also, when operating on a flat seabed, the SLAM algorithm should not get overconfident. It might be wise to make the resample policy dependent on the currently observed length scale.
- Another interesting future research is to evaluate how the proposed SLAM algorithm compares against other bathymetry SLAM algorithms. A dataset with accurate heading estimates and an acoustic beacon trajectory for comparison would be best suited as this provides the accurate reference position. An interesting dataset that might be suitable is recently published in the work of Kristopher Krasnosky [18].
- Multi-beam or side scan sonars provide a larger swath width than the DVL, which is ideal for overlap during a typical lawnmower mission pattern. However, the areas with overlap will be mainly at the end of the swath, where the α_j^b angles have a more significant component in the horizontal plane. Research on how to consider the GP input uncertainty will be more relevant here, as the effects of neglecting the input uncertainty are larger.

- For practical applications, the development of bathymetry smoothing would be interesting as post-processing step. This also allows information from resurfacing GNSS measurements to be taken into account. By taking into account the bathymetry, the required resurfacing steps might be reduced.

For real-world implementations, keep in mind the following recommendations.

- Due to the high overhead costs of a support vessel, it quickly makes sense to spend additional money for a higher grade IMU capable of accurately determining the heading. If this is the case, the algorithm can be easily changed by considering the heading estimate of the Extended Kalman Filter (EKF) as deterministic input and changing the corresponding dynamic model. Due to the reduced sample dimension and the more accurate odometry predictions, the algorithm would deliver improved results in terms of positional accuracy.
- For fast SLAM implementations, shared pointers for tiles are key to copying the map efficiently during resampling.

Appendix A

Rotations

Rotations are used to represent the orientation of the different reference frames. A rotation can be represented in multiple formats, this section introduces the rotation matrix, the Euler angles, and the quaternion.

A-1 Rotation matrices

Rotation matrices $R \in \mathbb{R}^{3\times 3}$ are parametrized with 9 parameters as

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}. \tag{A-1}$$

These parameters are related to each-other based with the following properties of rotation matrices:

$$RR^\top = R^\top R = I_3, \qquad \det(R) = 1. \tag{A-2}$$

Vector v defined in the $\{b\}$ frame can be rotated to the $\{n\}$ frame as

$$v^n = R^{nb} v^b. \tag{A-3}$$

The reverse rotation of R^{nb} is related as: $R^{bn} = (R^{nb})^\top$ based the same rotation matrix

$$v^b = (R^{nb})^\top v^n = R^{bn} v^n. \tag{A-4}$$

The rotation matrix is can represent any rotation and is a unique parametrization of the rotation.

A-2 Unit quaternions

Unit quaternions represents the rotation between two reference frames. An unit quaternion $\boldsymbol{q} \in \mathbb{R}^4$ is parametrized as:

$$\boldsymbol{q} = \begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^\top, \qquad ||\boldsymbol{q}||_2 = 1. \tag{A-5}$$

Unit quaternions are well suited in attitude estimation algorithm as quaternions only have 4 parameters for a singularity free parametrization [20]. An unit quaternion can be converted to a rotation matrix is calculated as

$$R^{nb}(\boldsymbol{q}^{nb}) = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y - q_zq_w) & 2(q_xq_z + q_yq_w) \\ 2(q_xq_y + q_zq_w) & 1-2(q_x^2+q_z^2) & 2(q_yq_z - q_xq_w) \\ 2(q_xq_z - q_yq_w) & 2(q_yq_z + q_xq_w) & 1-2(q_x^2+q_y^2) \end{bmatrix}. \tag{A-6}$$

A-3 Euler angles

Euler angles are a rotation parametrization that are defined as three consecutive rotations around three axes. Euler angles are defined by three angles, the roll angle $\phi \in [-\pi, \pi]$, the pitch angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, and the heading angle $\psi \in [-\pi, \pi]$. There are many different Euler angle conventions, which differ in multiplication order and axes conventions. This report uses right handed coordinate frames and right handed rotations. Each of the roll, pitch and yaw rotations can be described by a rotation matrix

$$\begin{aligned} R(\psi) &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ R(\theta) &= \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \\ R(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \end{aligned} \tag{A-7}$$

Using the multiplication order of (A-8), Euler angles can be converted to a rotation matrix as

$$\begin{aligned} R(\phi,\theta,\psi) &= R(\psi)\,R(\theta)\,R(\phi), \\ &= \begin{bmatrix} \cos\psi\cos\theta & -\cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\phi - \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi - \sin\psi\sin\theta\cos\phi \\ \sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \end{aligned} \tag{A-8}$$

Wrapping of the Euler angels has to be taken into account as they are only defined on a bounded domain. The Euler angles are not unique and suffer from *gimbal lock* at $\theta = \frac{\pi}{2}$, for more detail see [16]. The representation is therefore not numerically stable, however the Euler angles have an intuitive physical interpretation.

Based on the choice of Euler angle multiplication order of Eq. (A-8), a rotation matrix R is converted to Euler angles as

$$\begin{aligned} \psi &= \tan^{-1}\left(\frac{R_{21}}{R_{11}}\right), \\ \theta &= \sin^{-1}\left(R_{31}\right), \\ \phi &= \tan^{-1}\left(\frac{R_{32}}{R_{33}}\right). \end{aligned} \tag{A-9}$$

Based on equations Eq. (A-6) and Eq. (A-9), we can also specify the conversion from unit quaternion to our definition of Euler angles as

$$\begin{aligned} \psi &= \tan^{-1}\left(\frac{2(q_x q_y + q_z q_w)}{1 - 2(q_y^2 + q_z^2)}\right), \\ \theta &= \sin^{-1}\left(2\left(q_x q_z - q_y q_w\right)\right), \\ \phi &= \tan^{-1}\left(\frac{2(q_y q_z + q_x q_w)}{1 - 2(q_x^2 + q_y^2)}\right). \end{aligned} \tag{A-10}$$

Appendix B

Element-wise expansion of geometric conversion

This chapter writes out the geometric conversion of Eq. (3-7) element-wise for the z-component and concludes that the z-component is independent of heading and horizontal position.

Let's take a vector $\boldsymbol{a} \in \mathbb{R}^3$ with nonzero components as

$$\begin{aligned} \boldsymbol{a} &= \boldsymbol{\ell}^b_{\mathrm{DVL}} + \boldsymbol{\alpha}^b_j r_{k,j}, \\ &= \begin{bmatrix} k \\ l \\ m \end{bmatrix}. \end{aligned} \tag{B-1}$$

Then the pose transformation of Eq. (3-7) for rotating and translating the vector $\boldsymbol{a}$ is

$$\boldsymbol{s}^n = \boldsymbol{p}^n + R\left(\phi, \theta, \psi\right) \boldsymbol{a}. \tag{B-2}$$

In Eq. (A-8) the rotation matrix for the considered Euler angle multiplication order is derived. Here the shorthand notation of $\sin\psi = \mathrm{s}\psi$ is be used for brevity, the rotation matrix Eq. (A-8) using this shorthand notation is

$$\begin{aligned} R\left(\phi, \theta, \psi\right) &= R\left(\psi\right) R\left(\theta\right) R\left(\phi\right), \\ &= \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi - \mathrm{s}\psi\mathrm{c}\phi & \mathrm{s}\psi\mathrm{s}\phi - \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi - \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi & -\mathrm{c}\psi\mathrm{s}\phi - \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi \\ \mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix}. \end{aligned} \tag{B-3}$$

Writing out Eq. (B-2) using the short hand notation results in

$$\begin{aligned}
\boldsymbol{s}^n &= \boldsymbol{p}^n + R(\psi)\, R(\theta)\, R(\phi)\, \boldsymbol{a} \\
&= \begin{bmatrix} \boldsymbol{p}_x^n \\ \boldsymbol{p}_y^n \\ \boldsymbol{p}_z^n \end{bmatrix} + \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi - \mathrm{s}\psi\mathrm{c}\phi & \mathrm{s}\psi\mathrm{s}\phi - \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi - \mathrm{s}\psi\mathrm{s}\theta\mathrm{s}\phi & -\mathrm{c}\psi\mathrm{s}\phi - \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi \\ \mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \begin{bmatrix} k \\ l \\ m \end{bmatrix}, \\
&= \begin{bmatrix} \cdot \\ \cdot \\ \boldsymbol{p}_z^n + k\mathrm{s}\theta + l\mathrm{c}\theta\mathrm{s}\phi + m\mathrm{c}\theta\mathrm{c}\phi \end{bmatrix}.
\end{aligned} \tag{B-4}$$

Where in Eq. (B-4) the z-component of $\boldsymbol{s}^n$ is independent of $\boldsymbol{p}_x^n$, $\boldsymbol{p}_y^n$ and ψ for any $\boldsymbol{a}$. Note that this is dependent on Euler angle conventions and is not the case in general. Any parametrizations will work as long as first the vector is rotated to the horizontal plane and then rotated in the horizontal plane.

Appendix C

Code snippets

C-1 Reduced rank approximation

Python code for determining the N and Λ matrix for the reduced rank approximation of the Gaussian process (GP) model and calculate the eigenvectors $\phi(\boldsymbol{s}_i)$.

```
from typing import Tuple
import numpy as np
from math import pi


def sorted_sd_eigenvalues(m: int,
                          L_tile: float,
                          sigma_se: float,
                          sigma_const: float,
                          L_se: float,
                          max_test_values: int = 20) -> Tuple[np.ndarray,
                              np.ndarray]:
    """
    Function that calculates the m largest spectral densities of the
        covariance function
    :param m: number of basis function used in approximation
    :param L_tile: half of the square tile domain.
    :param sigma_se: hyperparameter of SE
    :param sigma_const: hyperparameter of constant covariance
    :param L_se: hyperparameter of SE
    :param max_test_values: number of test values, typically set to 5 * m
        .
    :return: Diagonal matrix Lambda containing spectral densities in
        descending order and the corresponding sorted
               indices for eigenfunction calculation
    """
    test_integers = np.arange(1, max_test_values + 1)

```

```python
    # create all possible combinations between test_integer array in Z_unsorted
    test_integer_mesh = np.meshgrid(test_integers, test_integers, indexing="ij")
    Z_unsorted = np.column_stack([matrix.flatten() for matrix in test_integer_mesh])

    eigenvalues = np.sqrt(np.square(pi / (2 * L_tile) * Z_unsorted[:, 0])
                          + np.square(pi / (2 * L_tile) * Z_unsorted[:, 1]))

    sd_eigenvalues = spectral_density_se(eigenvalues, sigma_se=sigma_se, L_se=L_se)

    # sort in decreasing order and take m largest corresponding indices
    m_largest_idxs = np.argsort(sd_eigenvalues)[::-1][:m]

    Lambda = np.diag([sigma_const ** 2, *sd_eigenvalues[m_largest_idxs]])
    Z = Z_unsorted[m_largest_idxs, :]
    return Lambda, Z


def spectral_density_se(eigv: np.ndarray, sigma_se, L_se) -> np.ndarray:
    """
    Spectral density of Squared Exponential kernel with 2 inputs
    """
    return 2 * pi * sigma_se ** 2 * L_se ** 2 * np.exp(-0.5 * L_se ** 2 * np.square(eigv))


def get_phi(s: np.ndarray, m: int, Z: np.ndarray, L_tile: float) -> np.ndarray:
    """
    Calculate eigenvector phi based on permutation integers matrix Z.
    """
    phi = np.ones(m + 1)
    phi[1:] = 1 / L_tile
    for d in range(2):
        phi[1:] *= np.sin(pi * Z[:, d] * (s[d] + L_tile) / 2 * L_tile)
    return phi
```

C-2 Resampling of particle filtering

Example code for creating a resampled particle set based on a single evaluation of a random number and using standard increments to select all new particles.

```cpp
#include <random>
#include "../slam/slamParticle.h"

std::vector<slamParticle> Resample(const std::vector<slamParticle>& old_particles){
    std::vector<slamParticle> new_particles;

    int Np = old_particles.size();

    // Generate a random number between 0.0 and 1 / (Np + 1)
    std::uniform_real_distribution<> resampleRNG{0.0, (1.0f/(float)(Np + 1))};
    std::random_device rd{};
    std::mt19937 gen{rd()};
    auto random_number_weight = resampleRNG(gen);

    float particle_weight_sum = .0f;
    int number_of_new_particles = 0;

    for (auto &particle: old_particles) {
        particle_weight_sum += particle.weight;

        while (random_number_weight < particle_weight_sum && number_of_new_particles < Np) {
            new_particles.emplace_back(particle);
            number_of_new_particles++;
            random_number_weight += 1.0f / (float) Np;
        }
    }
    return new_particles;
}
```

Appendix D

Full page mapping images

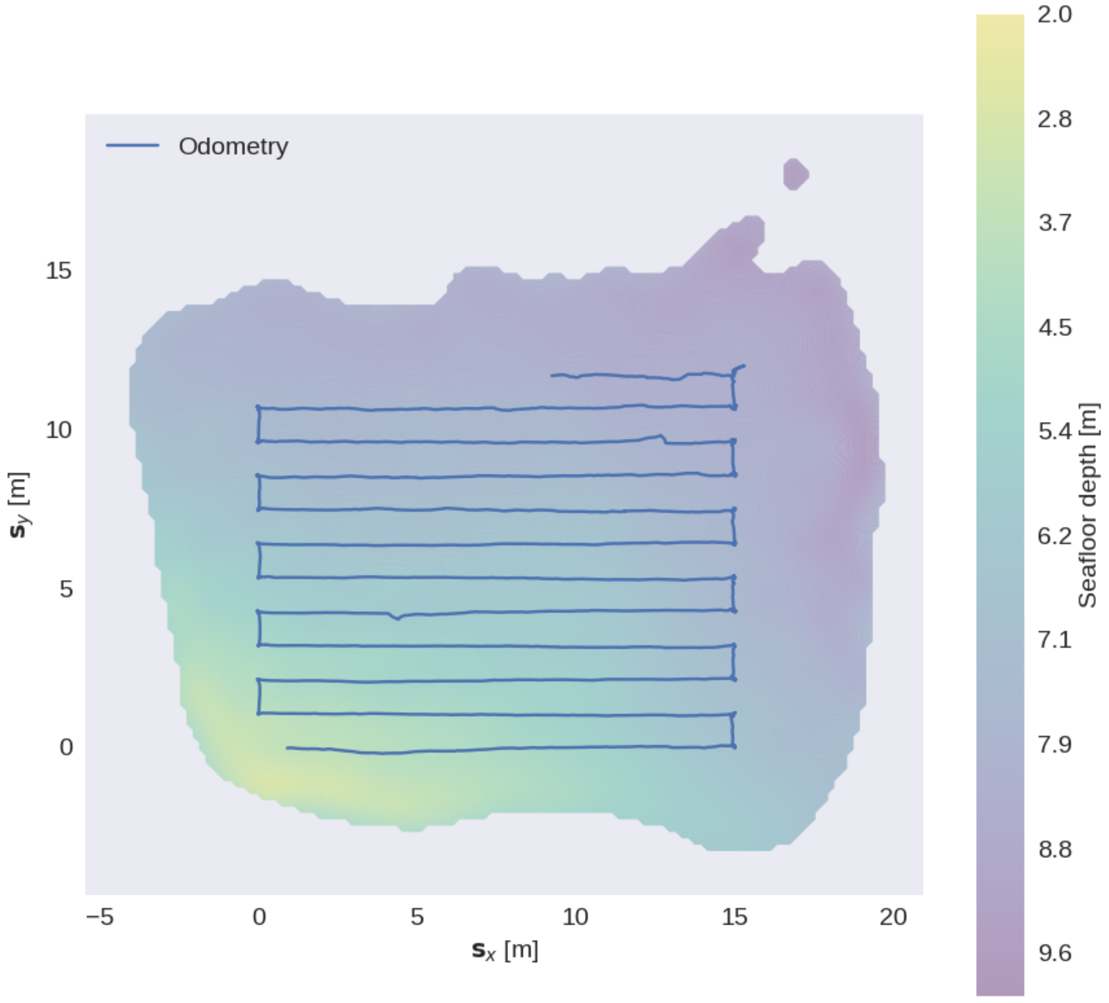


Figure D-1: Dense Gaussian process (GP) bathymetry of dataset 2

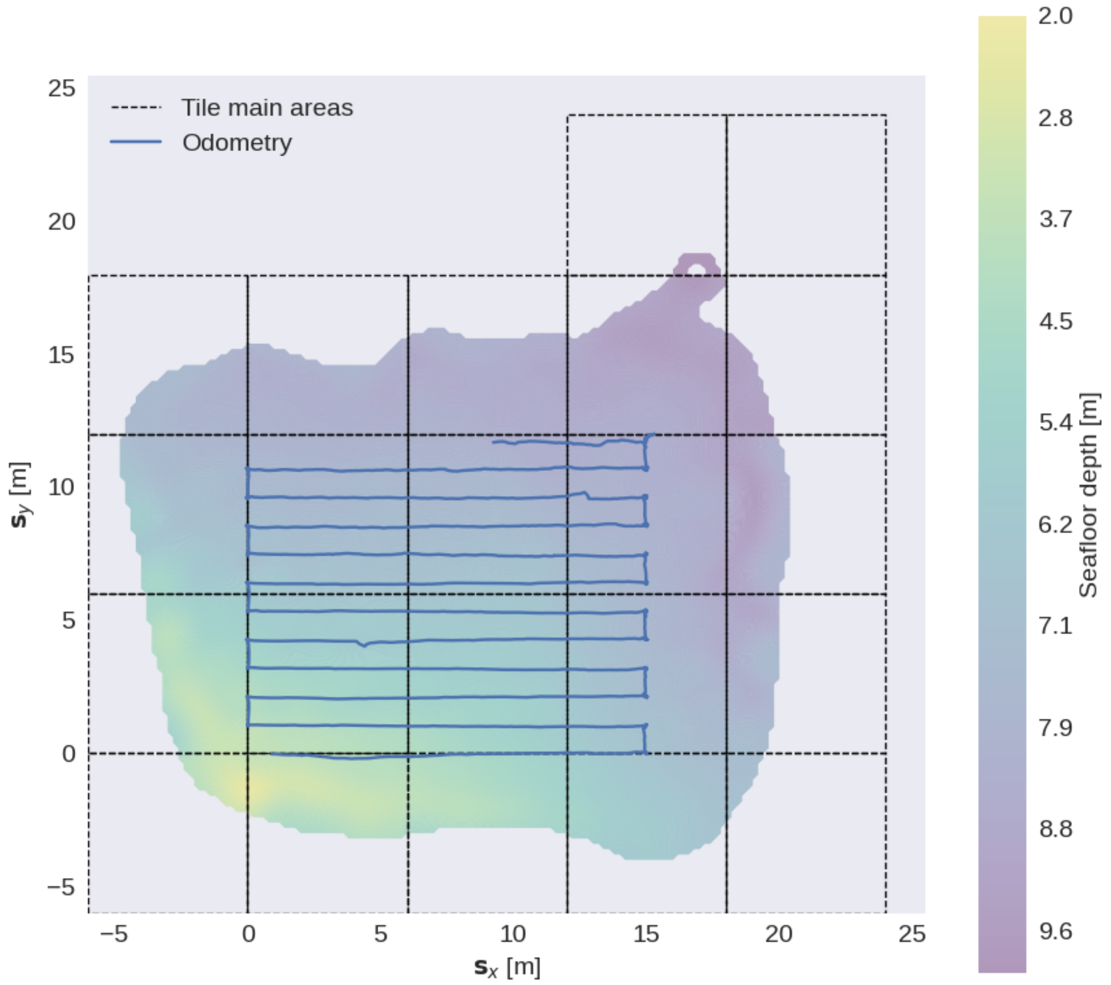


Figure D-2: Reduced rank GP bathymetry of dataset 2

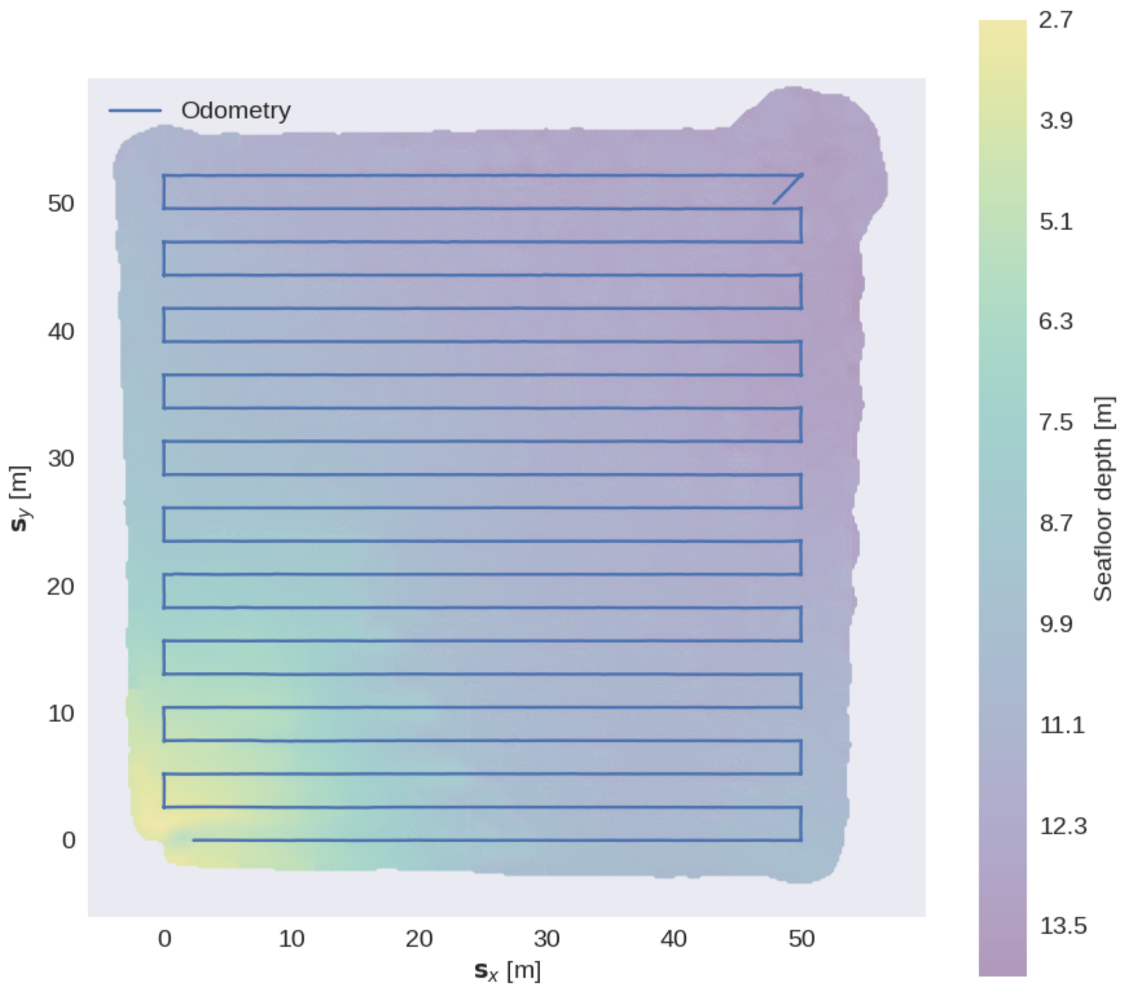


Figure D-3: Bathymetry map of Dataset 1 obtained by Algorithm 1, the edges are determined based on a maximum uncertainty threshold.

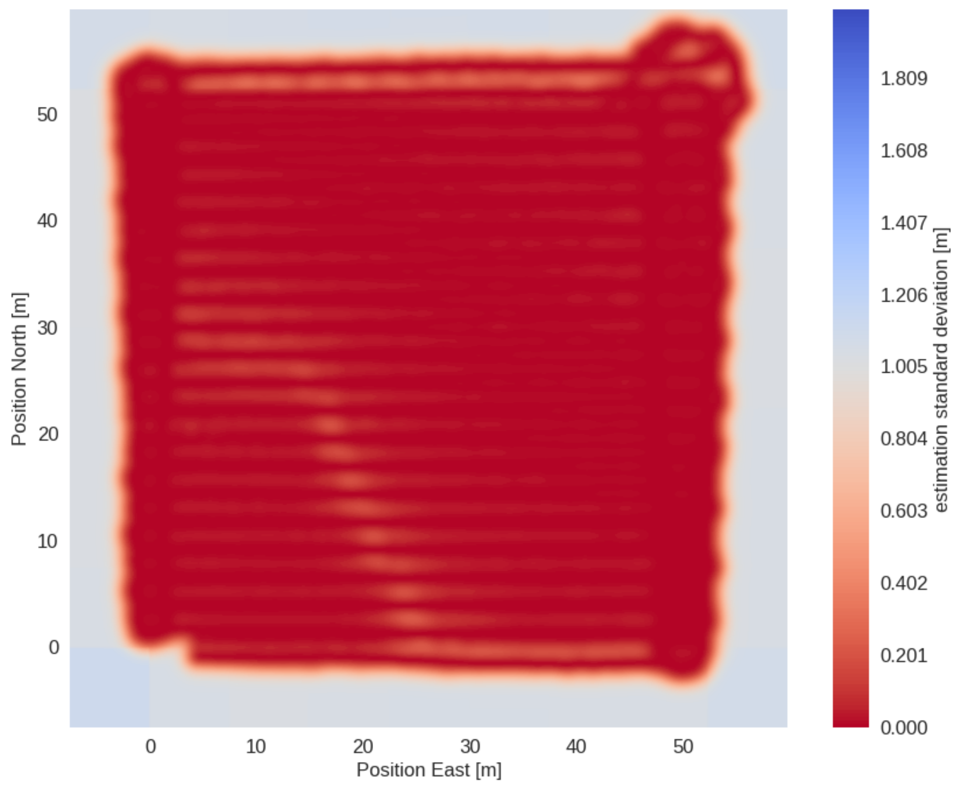
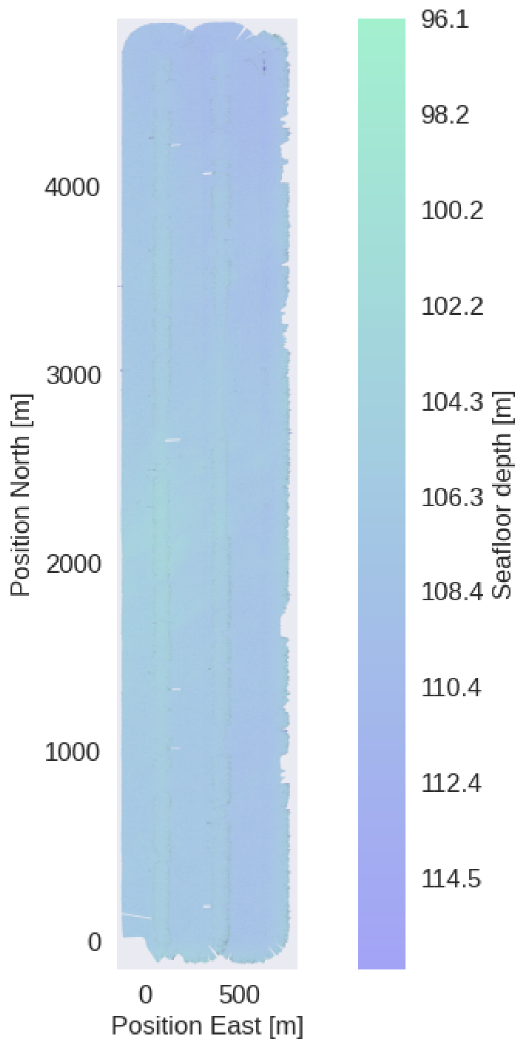
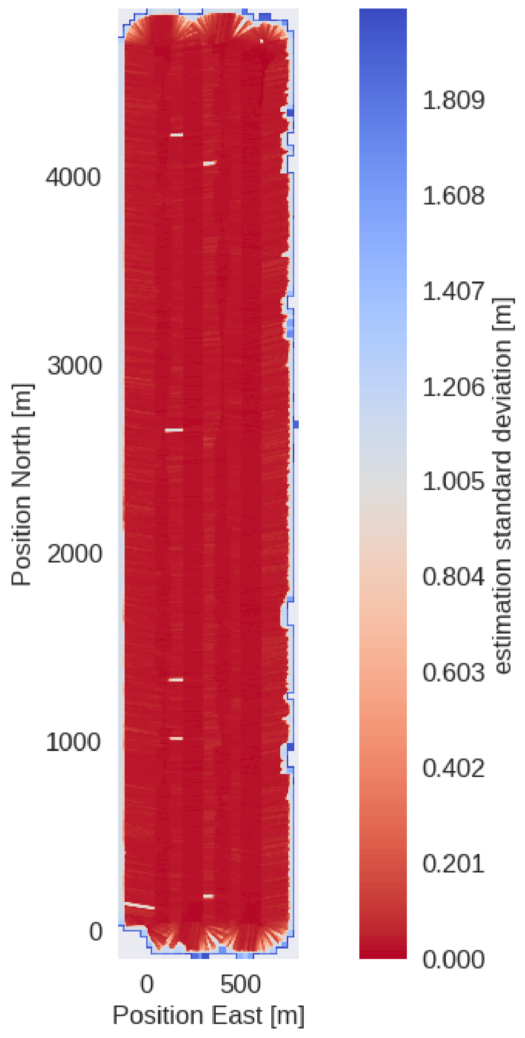


Figure D-4: Standard deviation associated with bathymetry map of Figure D-3.



(a) Bathymetry map



(b) Standard deviation map

Figure D-5: Bathymetry map of Aurora open-source dataset [2]. There are some blind spots due to lack of valid measurements in these areas.

Bibliography

[1] Stephen Barkby, Stefan B Williams, Oscar Pizarro, and Michael V Jakuba. Bathymetric SLAM with no map overlap using Gaussian processes. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, pages 1242–1248, 2011.

[2] Marco Bernardi, Brett Hosking, Chiara Petrioli, Brian J. Bett, Daniel Jones, Veerle A.I. Huvenne, Rachel Marlow, Maaten Furlong, Steve McPhail, and Andrea Munafò. AURORA, a multi-sensor dataset for robotic ocean exploration. *The International Journal of Robotics Research*, 41(5):461–469, 2022.

[3] Steve Bloomer, Peter Kowalczyk, Jeff Williams, Tony Wass, and Keisuke Enmoto. Compensation of magnetic data for autonomous underwater vehicle mapping surveys. In *Proceedings of 2014 IEEE/OES Autonomous Underwater Vehicles*, pages 1–4, 2014.

[4] Ned A Brokloff. Matrix Algorithm for Doppler Sonar navigation. In *Proceedings of Oceans 1994*, pages 378–383, 1994.

[5] Manhar R. Dhanak and Nikolaos I. Xiros. *Springer Handbook of Ocean Engineering.* Springer Cham, 2016.

[6] Heidi M. Dierssen and Albert E. Theberge. Bathymetry: Features and Hypsography. In Yeqiao Wang, editor, *Encyclopedia of Natural Resources: Water*, volume 2, pages 1–7. Taylor & Francis Group, 2014.

[7] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[8] Nathaniel Fairfield, Dominic Jonak, George Kantor, and David Wettergreen. Field Results of the Control, Navigation, and Mapping systems of a Hovering AUV. In *Proceedings of the 15th International Symposium on Unmanned Untethered Submersible Technology*, pages 20–31, 2007.

[9] R.C. Fofonoff, N.P., Millard Jr. Algorithms for computation of fundamental properties of seawater. *UNESCO Technical Papers in Marine Science*, 44(1), 1983.

[10] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons Ltd., Chichester, United Kingdom, 2011.

[11] Agathe Girard and Roderick Murray-Smith. Learning a Gaussian Process Model with Uncertain Inputs. Technical report, Department of Computing Science, University of Glasgow, Glasgow, 2003.

[12] Fredrik Gustafsson. *Statistical Sensor Fusion.* StudentLiteratur, Lund, Sweden, 3 edition, 2018.

[13] Bjørn Jalving, Magne Mandt, Ove Kent Hagen, and Freddy Pøhner. Terrain referenced navigation of AUVs and submarines using multibeam echo sounders. In *Proceedings from UDT Europe 2004*, Nice, France, 2004.

[14] R. Karlsson and F. Gustafsson. Particle filter for underwater terrain navigation. In *Proceedings of IEEE Workshop on Statistical Signal Processing*, pages 526–529, 2003.

[15] James C. Kinsey, Ryan Eustice, and Louis L. Whitcomb. A Survey of Underwater Vehicle Navigation: Recent Advances and New Challenges. *7th Conference on Manoeuvring and Control of Marine Craft (MCMC'2006)*, pages 1–12, 2006.

[16] Manon Kok, Jeroen D. Hol, and Thomas B. Schön. Using inertial sensors for position and orientation estimation. *Foundations and Trends in Signal Processing*, 11(1-2):1–153, 2017.

[17] Manon Kok and Arno Solin. Scalable Magnetic Field SLAM in 3D Using Gaussian Process Maps. In *Proceedings of the 20th International Conference on Information Fusion*, pages 1353–1360, Cambridge, UK, 2018.

[18] Kristopher Krasnosky, Christopher Roman, and David Casagrande. A bathymetric mapping and SLAM dataset with high-precision ground truth for marine robotics. *International Journal of Robotics Research*, 41(1):12–19, 2022.

[19] Mikael Bliksted Larsen. High Performance Doppler-Inertial Navigation-Experimental Results. In *Proceedings of Oceans 2000 MTS/IEEE Conference and Exhibition*, pages 1449–1456 vol.2, 2000.

[20] F. Landis Markley. Attitude error representations for Kalman filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317, 2003.

[21] Deborah K. Meduna, Stephen M. Rock, and Rob McEweny. Low-cost terrain relative navigation for long-range AUVs. In *Proceedings of Oceans 2008*, pages 1–7, 2008.

[22] Paul A. Miller, Jay A. Farrell, Yuanyuan Zhao, and Vladimir Djapic. Autonomous underwater vehicle navigation. *IEEE Journal of Oceanic Engineering*, 35(3):663–678, 2010.

[23] Albert Palomer, Pere Ridao, and David Ribas. Multibeam 3D underwater SLAM with probabilistic registration. *Sensors (Switzerland)*, 16(4), 2016.

[24] Konstantinos Papafotis, Dimitris Nikitas, and Paul P Sotiriadis. Magnetic Field Sensors ' Calibration : Algorithms ' Overview and Comparison. *Sensors*, 21(16), 2021.

[25] C.E. Rasmussen and K.I. Williams. *Gaussian Processes for Machine Learning.* The MIT press, Londen, England, 2006.

[26] Simo Särkkä. *Bayesian filtering and smoothing.* Cambridge university Press, Cambridge, England, 2013.

[27] Arno Solin, Manon Kok, Niklas Wahlstrom, Thomas B. Schön, and Simo Sarkka. Modeling and Interpolation of the Ambient Magnetic Field by Gaussian Processes. *IEEE Transactions on Robotics*, 34(4):1112–1127, 2018.

[28] Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.

[29] Luke Stutters, Honghai Liu, Carl Tiltman, and David J. Brown. Navigation technologies for autonomous underwater vehicles. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 38(4):581–589, 2008.

[30] Ioseba Tena Ruiz, Sébastien de Raucourt, Yvan Petillot, and David M. Lane. Concurrent mapping and localization using sidescan sonar. *IEEE Journal of Oceanic Engineering*, 29(2):442–456, 2004.

[31] Martin Visbeck. Ocean science research is key for a sustainable future. *Nature Communications*, 9(1):1–4, 2018.

[32] Louis L. Whitcomb, Dana R. Yoerger, Hanumant Singh, and Jonathan Howland. Advances in Underwater Robot vehicles for deep ocean exploration: Navigation, Control, and Survey Operations. In *Robotics Research*, volume 9, pages 439–449. Springer, 2000.

Glossary

List of Acronyms

AUV	Autonomous Underwater Vehicle
GP	Gaussian process
RBPF	Rao Blackwellized particle filter
MEMS	micro-electro-mechanical systems
DVL	Doppler Velocity Log
LBL	Long Base Line
SLAM	Simultaneous Localization and Mapping
RMSE	root mean squared error
GNSS	Global Navigation Satellite System
EKF	Extended Kalman Filter
SE	squared exponential