Master's thesis

Fredrik Elgsaas Alnæs

AUV mission planning for Multiple Vehicles using Adaptive sampling

Master's thesis in Marine Technology
Supervisor: Martin Ludvigsen
June 2020

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Fredrik Elgsaas Alnæs

AUV mission planning for Multiple Vehicles using Adaptive sampling

Master's thesis in Marine Technology
Supervisor: Martin Ludvigsen
June 2020

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER THESIS IN MARINE UNDERWATER ENGINEERING

SPRING 2020

FOR

STUD. TECHN. Fredrik Elgsaas Alnæs

AUV mission planning

Work description (short description)
Underwater missions are becoming more complex. This increase the importance of cooperative and coordinated behaviour between autonomous underwater vehicles. By using adaptive mission planning on multiple underwater vehicles simultaneously, it can improve accuracy, time, functional distributions, flexibility, and adaptability. In this project the goal is to find a method for two or more autonomous underwater vehicles to execute a mission in an area and simultaneously updating their individual ocean models by using adaptive mission planning to optimize ocean data sampling.

Scope of work
1. Review necessary literature within the field of mission planning, cooperative underwater systems and adaptive communication and mission planning.
2. Propose a control method/model for the AUVs communication system.
3. Propose an algorithm for communication and cooperative AUV system.
4. Propose a structure for real time information processing and modelling for adaptive and collaborative mission planning for multiple vehicles.
5. Test and verify the software and system by simulation.
6. Test and verify the software and system through field tests with AUVs.
7. Analyse and compare the results from simulations and field test.

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulation results, model test results, discussion and a conclusion including a proposal for further work. Source code should be provided. It is supposed that Department of Marine Technology, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student’s work.

The thesis should be submitted within 10. June 2020.

Co-supervisor: Trygve Olav Fossum

Professor Martin Ludvigsen
Supervisor

Abstract

Multiple AUVs are today used in ocean sampling to study how marine life is influenced by changes due to ocean phenomena's like ocean fronts or eddies. Ocean sampling often relies on pre-programmed missions. This is problematic since prior knowledge of the ocean is usually poor and constantly changing. The purpose of this thesis is to develop an algorithm and test decision methods to use adaptive sampling onboard multiple AUVs to improve ocean sampling. Adaptive sampling can result in improved efficiency and sampling results. By using multiple AUVs simultaneously and perform a cooperative mission with communication, the goal is to increase the efficiency and improve the sampling results compared to multiple AUVs running a pre-planned mission.

In this thesis, 12 simulations of adaptive sampling are done by using a Python program aided by PyIMC to control the multiple AUVs. The AUVs are simulated using the LSTS Toolchain. An ocean server based on data from a SINMOD file is used to simulate the ocean sampling. Gaussian Process is used to predict the temperature in the mission area based on the ocean sampling done by the AUVs. As an input to the Gaussian Process, the mean of all temperature grids available in the SINMOD file, are used. The prediction from the Gaussian Process is used as the input for testing several myopic decision methods.

The simulation results show an improvement in both efficiency and quality when gathering ocean data. The mean absolute error between the actual temperature and mean temperature used in the Gaussian Process was $0.03046°C$. Results from adaptive sampling with two AUVs communicating show a mean absolute error as low as $0.01410°C$. This simulation ran for 13 hours. The mean absolute error for adaptive sampling with one AUV or two AUVs not communicating was approximately $0.020°C$. The simulation ran for 25 hours for one AUV and 12 hours for two.

The algorithm presented in this thesis makes it possible to survey an area both time-efficient and with a finer resolution. This algorithm should be able to work with any ocean data. By using Gaussian Process to predict the ocean data, adaptive sampling can broaden the possibilities to discover data sets never gathered before. This is possible even with little or no previous ocean data. The results show the successful implementation of a decision model for multi-AUV systems doing the adaptive sampling. Real-life performance has to be tested before this can be said for certain. Before using the algorithm provided here in a field test, some issues need to be looked further into. Further work needs to be done in regards to making a better communication model. As technology develops the possibilities of using non-myopic decision methods together with more complex Gaussian Process models can contribute to better ocean sampling.

Keywords: Marine Robotics, Ocean Sampling, Multi-Vehicle Operations, Cooperative Approach, Adaptive Sampling

Sammendrag

Flere AUV-er brukes i dag i prøvetaking av havet for å studere hvordan marint liv i havet påvirkes av endringer på grunn av havfenomener som havfronter eller eddier. Havprøvetaking er ofte avhengig av forhåndsprogrammerte oppdrag. Dette er problematisk siden forkunnskap om havet vanligvis er dårlig og stadig endrer seg. Hensikten med denne oppgaven er å utvikle en algoritme og teste beslutningsmetoder for å bruke adaptiv prøvetaking ombord i flere AUV-er for å forbedre prøvetaking av havet. Adaptiv prøvetaking kan resultere i forbedret effektivitet og prøvetakingsresultater. Ved å bruke flere AUV-er samtidig og utføre et samarbeidsoppdrag med kommunikasjon, er målet å øke effektiviteten og forbedre prøvetakningsresultatene sammenlignet med flere AUV-er som kjører et forhåndsplanlagt oppdrag.

I denne masteroppgaven gjøres det 12 simuleringer av adaptiv prøvetaking ved å bruke et Python-program hjulpet av PyIMC for å kontrollere flere AUVer. AUVene simuleres ved hjelp av LSTS Toolchain. En havserver basert på data fra en SINMOD-fil brukes til å simulere havprøvetaking. Gaussiske prosesser brukes til å forutsi temperaturen i oppdragsområdet basert på havprøvetaking utført av AUVer. Som input til den gaussiske prosessen, brukes gjennomsnittet av alle tilgjengelige temperaturer fra kartreferansene i SINMOD-filen. Prediksjonen fra den gaussiske prosessen brukes som input for å teste flere myopiske beslutningsmetoder.

Simuleringsresultatene viser en forbedring i både effektivitet og kvalitet når man samler havdata. Den gjennomsnittlige absolutte feilen mellom den faktiske temperaturen og den gjennomsnittlige temperaturen som ble brukt i den gaussiske prosessen var $0,03046°C$. Resultat fra adaptiv prøvetaking med to AUVer som kommuniserer, viser en gjennomsnittlig absolutt feil så lavt som $0,01410°C$. Denne simuleringen kjørte i 13 timer. Den gjennomsnittlige absolutte feilen for adaptiv prøvetaking med en AUV eller to AUVer som ikke kommuniserer, var omtrent $0,020°C$. Simuleringen gikk i 25 timer for en AUV og 12 timer for to.

Algoritmen presentert i denne oppgaven gjør det mulig å kartlegge et område både tidseffektivt og med en finere oppløsning. Denne algoritmen skal kunne arbeide med alle havdata. Ved å bruke gaussiske prosesser for å forutsi havdata, kan adaptiv prøvetaking utvide mulighetene for å oppdage datasett som aldri har blitt samlet før. Dette er mulig selv med lite eller ingen havdata på forhånd. Resultatene viser vellykket implementering av en beslutningsmodell for fler-AUV-systemer som gjør adaptiv prøvetaking. Utførelser i det virkelige liv må testes før dette kan sies med sikkerhet. Før bruk av algoritmen beskrevet her i en feltprøve, må noen problemer undersøkes nærmere. Det må jobbes videre med å lage en bedre kommunikasjonsmodell. Når teknologien utvikles, vil mulighetene for å bruke ikke-myopiske beslutningsmetoder sammen med mer komplekse gaussiske prosessmodeller, kunne bidra til bedre prøvetaking av havet.

Nøkkelord: Marin Robotikk, Havprøvetaking, Operasjoner Med Flere Fartøy, Samarbeidsmetode, Adaptiv Prøvetaking

Preface

This master thesis is written as the final part of the five years integrated master program Marine Technology, with specialisation in underwater engineering, at the Norwegian University of Science and Technology. Some of the underlying research was carried out in the author's project thesis, the fall of 2019. The remaining research and development of the product were carried out during spring 2020. It is assumed that the reader of this thesis possesses a basic knowledge of engineering science.

Acknowledgement

First of all, I would like to thank Professor Martin Ludvigsen as my main supervisor. Together we have defined an interesting scope and objective for this master thesis. Thank you for your guidance regarding interesting articles and perspectives on the topic. Secondly, I would like to thank PhD candidate Trygve Olav Fossum as my co-supervisor. Thank you for providing me with a Gaussian Process program and helping me both with programming, simulation environment and discussing the problem. Thanks for providing me with good "tips and tricks" and constant support, both morally and academically. I would also like to thank PhD candidate Øystein Sture for giving me an introduction to and helping me set up his program PyIMC.

Finally, I would like to thank my parents, sister and my companions at my office, A2.019, for always supporting me and make me concentrate on the most stressful days. Last but not least, I would like to thank my wife Maren; this master thesis would not be possible without your support.

Trondheim, June 9, 2020

Fredrik Elgsaas Alnæs

Contents

List of Figures

Nomenclature

Acronyms

ℓGP	log-Gaussian Process
AI	Artificial Intelligence
AUV	Autonomous Underwater Vehicles
CCU	Central Control Unit
Chl	Chlorophyll
$Chla$	Chlorophyll-a
CTD	Conductivity Temperature and Depth
GP	Gaussian Process
GPs	Gaussian Processes
IMC	Inter-Module Communications protocol
$LAUV$	Light Autonomous Underwater Vehicle
$LSTS$	Laboratório de Sistemas e Tecnologia Subaquática
MAE	Mean Absolute Error
MCL	Mission Control Language
ML	Mission Language
R/V	Research Vessel
RF	Radiofrequency
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
UAV	Unmanned Aerial Vehicles
UUV	Unmanned Underwater Vehicle
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Background

Only a small part of the large ocean is surveyed. The ocean is the source of all life and is constantly changing. Gathering more and better data from the ocean is something biologists and oceanographers are working on all over the world. Some important applications of ocean data are water quality monitoring, ecosystem modelling, ocean modelling, and studying harmful algal blooms. Usually, ocean data are sampled by either taking a physical water sample from a boat and analysing it or deploying sensors attached to buoys or docks. Important samples these sensors record is Conductivity, Temperature and Depth (CTD), chlorophyll (Chl), pH, turbidity, nitrate, and dissolved oxygen. Since the sensors only take spot measurements in their position, this data gathering method is time and cost-intensive and gives only limited data. To use aquatic robots instead could improve cost, time, and give a finer resolution of the sampled data.

Underwater missions using Autonomous Underwater Vehicles (AUV) is often pre-planned. This means that the AUV is told where to go and what to do before the mission starts. With an ocean in motion and large areas with low or no data available the question arises: *"How can we use multiple AUVs to ensure that we gather the best data from an area in the most efficient way with minimal prior information?"*

Adaptive mission planning is a method used to improve missions, both in efficiency and quality. When doing adaptive mission planning, the mission is planned onboard the vehicle instead of before the mission. This makes it possible for the vehicle to make decisions based on the sampled data, resulting in finding the most informative locations and sampling it more thoroughly and more efficient than with pre-mission planning. One way to do this is by using Gaussian Process (GP) to predict the data changes in the ocean while sampling. Since GP is computational heavy, because of the need to transpose large matrixes, the programming language Python can be used to predict the resulting ocean data from measurements. The AUVs at NTNU uses the software from LSTS and therefore the Python package PyIMC by Sture (2019) is a good way to use Python to control the AUVs through IMC-messages.

1.2 Objectives

Earlier, the sampling of the ocean was done by ships and based on experience. Today more and more robotic solutions are used to take samples of the ocean. Still, a lot of the experiments done with the equipment available today is done based on earlier experience and knowledge. By using adaptive sampling, it is possible to do more efficient experiments with better results.

When doing experiences over larger areas, the efficiency can be increased even more by using several vehicles, to cover a larger area in less time. By introducing several vehicles, the complexity of the mission increases. When using AUVs, communication between them is often limited to acoustic communication with low bandwidth. This makes it hard to optimize operations.

The primary goal of this thesis is to develop an application for adaptive mission planning for multiple AUVs doing ocean sampling within a given area and updating and sharing their ocean model to optimize ocean data sampling. The motivation behind this thesis is to contribute to the research and testing of adaptive mission planning for multiple AUVs to improve ocean sampling. To reach the goal of this thesis, the following objectives were planned to be performed:

- Review necessary literature within the field of mission planning, cooperative underwater systems and adaptive communication and mission planning.
- Propose a control method/model for the AUVs communication system.
- Propose an algorithm for communication and cooperative AUV system.
- Propose a structure for real-time information processing and modelling for adaptive and collaborative mission planning for multiple vehicles.
- Test and verify the software and system by simulation.
- Test and verify the software and system through field tests with AUVs.
- Analyse and compare the results from simulations and field test.

1.3 Limitations

The application for adaptive sampling by using PyIMC is made with NTNUs vehicles running with software from LSTS (explained further in section 3.2) in mind. Therefore, the application will not work for AUVs running another operating system, i.e. robotic operating system (ROS).

There was supposed to be a field test included in this master thesis, but unfortunately, this was not possible to do due to the Coronavirus. Therefore, all results of the testing of the application are simulated. The data the AUVs are gathering is also stationary, and the result will, therefore, not resemble reality. The simulations are also done in the surface to simulate a more realistic way of accessible communication since it can use several communication methods, i.e. Wi-Fi, satellite, and cellular network, when in the surface. If the application is to be run sub-surface, a more advanced communication method is needed.

1.4 Approach

This thesis is a continuation of the project thesis conducted during the fall of 2019 by the author. In the project thesis, a simple adaptive sampling application was carried out and simulated for one AUV. In this thesis, this is taken a step further by expanding the application to work for two AUVs and developing a better decision method. Before starting the implementation, the author gained new knowledge within the field by studying literature and theory. After the implementation was done, several simulations were run to verify that the application worked. The results were then analysed, and the conclusion and recommendations for further work can be found at the end of this thesis.

1.5 Contributions

The main contribution of this thesis is the application for simulating two or more AUVs in a cooperative mission using GP to do adaptive sampling together. This provides a framework for testing different decision methods when sampling ocean data. By doing minor changes to the code, it is possible to test pre-planned missions, brute force adaptive sampling and adaptive sampling done by using GP for two or more AUVs. The focus of this thesis has been to test different myopic decision models for adaptive sampling for two AUVs using GP. The results show successful implementation of a decision model for multi AUV systems doing the adaptive sampling. The thesis contributes with visualisation of decision methods, showing which decision models that work. The results also show the benefits of communications between the AUVs, making it an important part when using several vehicles.

1.6 Structure of thesis

Chapter 2: presents the literature study done on adaptive sampling and cooperative underwater systems.

Chapter 3: gives an overview of the programing tool PyIMC, the ocean model SINMOD, the proposed algorithm, GP and the decision model. At the end of the chapter, the setup of each simulation is described.

Chapter 4: presents the results from the simulations done in this thesis.

Chapter 5: analyses and discuss the results of the adaptive sampling.

Chapter 6: concludes on the thesis and suggests further work that can be done with this algorithm.

Appendix A: presents a short description and overview of the code used in this project thesis.

Appendix B: presents the temperature grid for the different timestamps considered in the mean temperature used in this thesis.

Chapter 2

Literature review

2.1 Underwater Vehicles

Unmanned underwater vehicles (UUVs), often divided into two categories. Remotely operated vehicles (ROVs) are linked to a surface vessel or another control unit by an umbilical that delivers power and control commands. The AUV, on the other hand, has no umbilical, making it flexible and able to reach high speed and operate in complex environments.

Since the first AUVs in the 1970s, there has been a significant advancement in size and efficiency. The memory capacity of computers has made it possible to enhance the potential and extend the usage (Paull et al. 2014). Today AUVs are often equipped with a variety of sensors used in different operations. The AUVs are shaped like a torpedo which is a good hydrodynamical shape. This makes the AUVs able to reach high speeds and operate in complex environments. One of the biggest challenges with the AUVs are their navigation and localisation systems. GPS systems do not work underwater, so the AUVs are often limited to acoustic navigation together with inertial navigation by sensors to do dead reckoning. The drawback of inertial navigation is that the error propagates by the iterations. Therefore, the AUVs often go to the surface with given intervals to recalibrate based on the GPS signal.

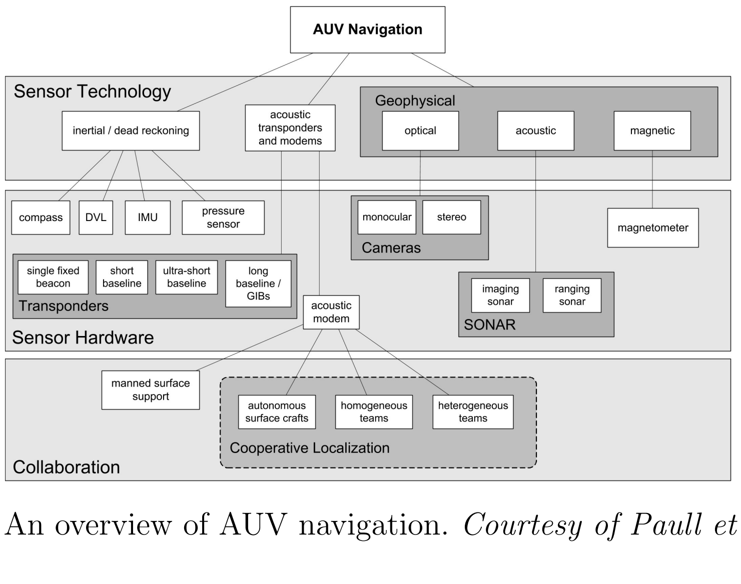


Figure 2.1: An overview of AUV navigation. *Courtesy of Paull et al. (2014).*

NTNU has several underwater vehicles, but the majority of them is Light Autonomous Underwater Vehicles (LAUVs). They are developed by the Underwater Systems and Technology Laboratory at University of Porto. The concept of the LAUV is that they are one-man-portable vehicles that can be easily launched and recovered with minimal operational setup. The LAUVs are relatively easy to use so that there is no need for extensive operator training. While being affordable, the LAUVs is based around a basic functional system that includes communication, computational system and basic navigation. The system is built so that it is possible to add optional payload modules (*Light Autonomous Vehicle LAUV - AUVAC* 2019).

2.1.1 Sensor Systems and Applications

When AUVs are deployed, it is often to gather data underwater. Both for navigating and sensing the environment, AUVs are equipped with several sensors. The sensors is often divided into two main groups: the *payload sensors* and the *navigation sensors* (Sørensen and Ludvigsen 2015). Navigation sensors measure the state of the vehicle in 6 degrees of freedom and use the internal control system to position the AUV. The sensors that collect data is known as the payload sensors; what sensors an AUV has is dependent on the goal of the mission.

Some common payload sensors in AUVs are:

- **Acoustic Doppler Current Profiler:** uses acoustic pulses, often from four transducers, to measure backscatter intensity and Doppler Shift of the reflected signal. This is used to calculate the current velocity vector in three dimensions. It can also work as an acoustic Doppler Velocity Log or track the seabed, measuring velocity relative to the seabed.
- **Conductivity Temperature Depth Sensor (CTD):** measures the water conductivity to find the salinity, the temperature and the pressure to find depth. From this, the CTD can find the speed of sound in water.
- **Side Scan Sonar (SSS):** uses fan-shaped acoustic impulses to map the seabed by measuring the intensity and travel time of the reflected signals. The advantage of the AUV compared to surface vehicles is that it can get close to the seabed, resulting in higher spatial resolution. By looking for landmarks, the SSS can improve AUV navigation.
- **Environmental Characterisation Optics:** often uses a scattering sensor together with a fluorimeter to gather data on chlorophyll and turbidity.
- **Multiparameter Sonde:** is used to measure dissolved oxygen, oxidation-reduction potential and pH levels in the water.

Some common navigation sensors in AUVs are:

- **Acoustic Baseline sensors:** is often used in three different classes, long-baseline, short-baseline, and ultra-short baseline. The long-baseline uses seabed mounted transponders placed at known positions to increase accuracy in navigation. The short baseline uses transponders deployed to the surface vessel, and AUV finds its position relative to the surface vessel. The ultra-short baseline uses only one transponder to find the position. The AUV has an array of hydrophone that can calculate horizontal and vertical angles to the transponder. Together with range measurement, this gives the position relative to the transponder.

- **Doppler Velocity Log:** uses the acoustic doppler current profiler as a doppler velocity log by measuring the doppler shift of the signal reflected off the seabed. This makes it possible to calculate the velocity relative to the seabed.
- **Heading and Inertial Sensors:** are used to find heading, accelerations and rate of change in orientation angles. The heading can be found using a magnetic compass or a gyrocompass. Angular rates can be found using a ring laser gyroscope.

2.1.2 Autonomy

When describing autonomy, a lot of literature, research and terms come up. Insaurralde and Petillot (2013) describes the conceptual definitions within the robotics domain as this:

- **Capability:** The ability a system has to fulfil its task or activity in an environment effectively.
- **Autonomy:** The ability a system has to make choices autonomously and enforce the decision made. This is accomplished when the system is given self-government and self-determination conditions.
- **Intelligence:** The ability the system has to accurately determine what activity will result in a maximum likelihood of achieving the goal successfully.
- **Knowledge:** The ability to use intellectual machinery to achieve goals and create new information from data with meaning.

An AUV can be seen as autonomous and flexible if it interacts with the environment by reacting to certain situations (reactive behaviour). The AUV needs to be able to guide itself based on the goal to carry out different activities (proactive behaviour). It also needs to be able to communicate with other AUVs by having the same universal language (social behaviour) (Insaurralde and Petillot 2013). To become intelligent, the AUV needs to have the capacity to perform intellectual functions like planning, reasoning and communicating, when solving problems and making decisions.

2.2 Ocean sampling

More than 70 per cent of Earths surface is water. The oceans are essential for driving the weather, regulating the temperature and supporting all living organisms. Even though the ocean is a vital source for transport, sustenance, commerce, growth, and inspiration, still more than 80 per cent of the ocean is not mapped, observed or explored (NOAA 2018). Known ocean observations reach back to Charles Darwin when he set sail on the HMS Beagle where he made several observations about the ocean life among other things in 1831 (Darwin 1989). The more ocean specific expeditions started in 1872 when the HMS Challenger travelled more than 100.000 km to sample ocean data until 1876 (Bailey 1953). Since then, the way oceanographers gather and study ocean data has improved. They are still looking for ways to improve the systems and technologies further. This is mainly driven by the need for new and different data measurements and a goal of reducing the cost of today's technology and methods.

Today we know about several ocean phenomena in the ocean, like upwelling, eddies, and ocean fronts. This makes for interesting observation points for scientists as they often result in changes in the ecosystem. To know when and where this phenomenon is going to occur is challenging. It is challenging to create models describing the ocean since it is of a large scale with nonlinear processes and high spatiotemporal variability. Models are mainly based on gathered ocean data and hydrodynamic models. When planning a mission, it is essential to have as much information as possible beforehand of the area the mission takes place. The simulated data and the real-world data is often very different, and therefore an adaptive method making real-time adjustments based on current observations can improve the results.

Fossum, Fragoso, et al. (2019) describes the goal of developing an adaptive behaviour of an AUV so that it can give an increased sampling resolution of the water-column process in three dimensions. This is done by using GP to model the subsurface chlorophyll-a (Chla) maxima. Chla is an important measurement since it says something about the amount of phytoplankton.

In Fossum, Eidsvik, et al. (2018) a SINMOD (further described in subsection 3.1.1) ocean model is used to describe the state of the ocean. This is used together with a greedy adaptive sampling algorithm running onboard the AUV to utilize the variance and mean estimates from the GP model to evaluate objects and find the best survey line. In this article the T-REX autonomous agent architecture is used, it synthesizes plans and uses an artificial intelligence AI-based automated planning execution temporal framework to execute tasks continuously based on the sensing and control data fed to it. Communication between T-REX and DUNE is handled by the LSTS toolchain described in section 3.2.

The PhD thesis of Kemna (2018) focuses on obtaining data from natural phenomena resulting in an abundance of algae. The studying of the abundance of algae is essential to understand and explain potentially harmful algal blooms. To improve cost- and time-efficiency of lake and ocean sampling single- and multi-robot deployments are tested both with formation control and adaptive sampling methods.

2.3 Adaptive Sampling

2.3.1 Intelligent sampling

Buadu (2018) and second chapter of Kemna (2018) considers multiple robots with formation control to sample ocean data. This method results in improved efficiency with more persistent and efficient mapping with finer resolution without much prior information. Kemna (2018) states that environmental sampling with multiple robots in a formation results in an effective and robust approach. Still, the most important aspect when doing environmental sampling is the quality of the data gathered. In environmental sampling, the biologist, oceanographer, or other environmental scientists as end-users of the data gathered is more interested in the measurements made than how it is done. To them, the most informative data is the most interesting data. The formation control method for ocean sampling is often pre-planned. Adaptive sampling use information-theoretic matrices to determine where the most informative sampling locations are.

When doing adaptive sampling, an AUV is programmed to adjust the sampling plan online onboard during the mission. The AUV samples its surroundings, then compares it to pre-set commands or what it knows before. After that, the AUV updates the sampling plan and generates a new sensing strategy before it acts on it. A perspective of this is shown in Figure 2.2.

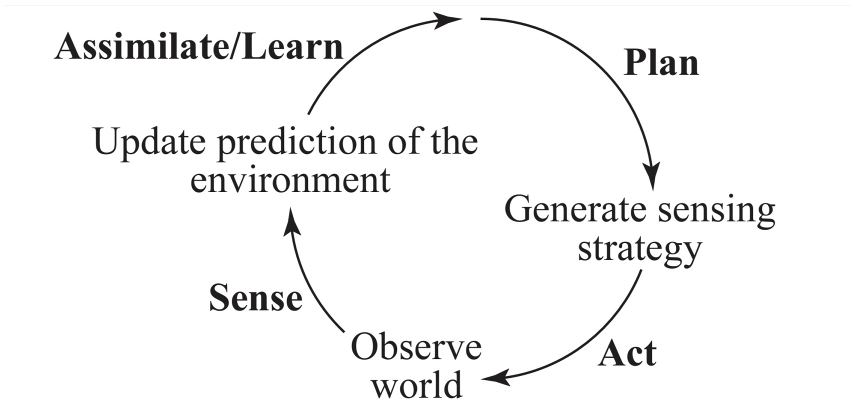

Assimilate/Learn
Plan
Update prediction of the environment
Generate sensing strategy
Sense
Act
Observe world

Figure 2.2: A perspective of the adaptive sampling for ocean observation. *Courtesy of Fossum (2019).*

Guestrin, Andreas Krause, and Ajit Singh (2005) (A. Krause et al. 2006), (Amarjeet Singh et al. 2006) can be seen as some sort of pioneers within informative sampling for using GP regression to model the environment. The goal was to determine where the most informative sampling locations for a GP model was. In A. Krause et al. (2006) this was used to determine where to deploy sensor systems. Amarjeet Singh et al. (2006) used this together with path planning for multiple robots. The method they used was off-line. They started with a GP model, and from that, they decided where to deploy the sensors and the robots.

K. H. Low, Dolan, and Khosla (2008) extended the works of Guestrin, Andreas Krause, and Ajit Singh to do adaptive sampling by doing the path planning on-line, during execution based on the new data incorporated. By using both Gaussian Processes (GPs) and log-Gaussian Processes (ℓGPs) to model the environment. Kian Hsiang Low (2009) could then use a dynamic programming approach in both a single- and multi-robot path planning on-line.

Typical steps in adaptive sampling are:

- By using GP regression, Bayesian regression, multivariate normal distribution, or a linear combination of a set of basic functions, the system should construct a model of the environment.
- The model needs an information-theoretic metric, choose, e.g. entropy, mutual information, or sum of posterior variances.
- A path planning approach needs to be chosen. Some examples of path planning approaches are local greedy, global greedy, recursive-greedy, and dynamic programming.

2.3.2 Gaussian Process regression

GP regression, also known in geostatistics as Kriging, is a common method for creating environmental models of sampled spatial data (Kemna 2018). Since running a high-fidelity numerical ocean model on-board a robotic platform currently is infeasible, due to the computational demands being too high for the platform to manage, non-parametric modelling techniques like GP regression is used. By using a mean function (the prior) and its covariance matrix (the kernel), a GP model can be completely specified. Based on data measurements, x at locations $\boldsymbol{F}$ the GP regression estimates a signal based on the estimating mean and variance. The posterior mean and variance are calculated based on a joint Gaussian model (Fossum 2019):

$$p(\textbf{prior}, \textbf{data}) = \mathcal{N}\left(\begin{bmatrix}\textbf{prior}\\ \textbf{data}\end{bmatrix}; \begin{bmatrix}\boldsymbol{\mu}\\ \boldsymbol{F\mu}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{\Sigma F}^T\\ \boldsymbol{F\Sigma} & \boldsymbol{F\Sigma F}^T + \boldsymbol{T}\end{bmatrix}\right) \tag{2.1}$$

resulting in the two equations (Fossum 2019):

$$\boldsymbol{\mu}_{posterior} = \boldsymbol{\mu} + \boldsymbol{\Sigma F}^T \left(\boldsymbol{F\Sigma F}^T + \boldsymbol{T}\right)^{-1} (\boldsymbol{y} - \boldsymbol{F\mu}) \tag{2.2}$$

$$\boldsymbol{\Sigma}_{posterior} = \boldsymbol{\Sigma} - \boldsymbol{\Sigma F}^T \left(\boldsymbol{F\Sigma F}^T + \boldsymbol{T}\right)^{-1} \boldsymbol{F\Sigma} \tag{2.3}$$

The four matrixes are defined as follow (Fossum 2019):

- **Prior:** $\boldsymbol{\mu} = \mu(\boldsymbol{s}_i)$ is the prior estimate for all locations $i = 1, ..., n$
- **Observation matrix:** $\boldsymbol{F} = m \times n$ is a matrix with zeros and ones describing where the data measured is located. m being the number of observations/measurements.
- **Data:** $\boldsymbol{y} = \boldsymbol{Fx} + \varepsilon$, x being the process (ocean model) with Gaussian measurement noise $\varepsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{T})$; and $\boldsymbol{T} = \boldsymbol{\tau}^2\boldsymbol{I}$, setting $\boldsymbol{\tau}$ manually.
- **Covariance:** $\boldsymbol{\Sigma} = \texttt{cov}\,(\boldsymbol{s}_i, \boldsymbol{s}_j)$, where i and j being all location pairs $= 1, ..., n$.

The main drawback of the GP model is the required inversion of the covariance matrix: $\left(\boldsymbol{F\Sigma F}^T + \boldsymbol{T}\right)^{-1}$. This can be computationally expensive for large dimensional problems with many observations or points. However, as long as the problem is within the dimensions of available time and computational power, the GP serves these useful properties (Fossum 2019):

- Modelling and computation properties
- Model fitting
- Conditioning
- Measure of uncertainty

Gathering more data reduces the uncertainty in the model. The results from the GPs and the path planning method is important to find the best locations to explore. ℓGPs are a variation of GPs. The ℓGPs assume that the data follows a log-normal distribution, where the GPs assume a normal distribution. Environmental sampling for biological phenomena like hotspots typically follows a log-normal distribution (Crow and Shimizu 1987). In Kian Hsiang Low (2009) a ℓGP model is used together with expected sum of posterior variances and map entropy for sampling criteria. The dynamic approach is used both for single- and multi-robot path planning.

2.3.3 Kernel

The covariance or the GPs kernel is the core of the GP model and is affected by hyperparameters, or the kernels parameters. By using gradient-based optimization these can be estimated (Rasmussen and Williams 2006). The standard approach is taking a zero-mean prior and a squared exponential covariance function. This function is given in Rasmussen and Williams (2006) as:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{-\frac{1}{2l^2}|\mathbf{x} - \mathbf{x}'|\right\} \tag{2.4}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two training sample locations. The signal variance σ_f^2 and the kernel's length scale l is the hyperparameters and can be optimized by using backpropagation or conjugate gradient method (Kemna 2018).

2.3.4 Myopic vs. Non-myopic

The dynamic system, when doing spatial sampling, results in added complexity. Therefore, the need to do assumptions and simplifications results in various approaches when doing the adaptive sampling. The two main evaluations are myopic (greedy) or non-myopic (synoptic). The myopic evaluation uses a fixed and short planning horizon. When using a non-myopic method, the planning is done over several sequential steps. Guestrin, Andreas Krause, and Ajit Singh (2005) and K. H. Low, Dolan, and Khosla (2008) shows how a non-myopic approach for multiple robots can be used to assimilate newly gathered data with dynamic programming with both GPs and ℓGPs combined. By using the sum of posterior variances and entropy as the information for the sampling decisions.

Moving away from the myopic strategies often results in issues with running time due to scalability and computational load. This is because of the increased dimensionality of the problem space due to larger graph size or finer resolution. When finding feasible solutions to use non-myopic strategies, different types of heuristics, Markov properties, and Monte Carlo approaches are used to reduce the computational burden (Fossum 2019). The limited look-ahead in greedy approaches avoids this problem entirely, but at the cost of sacrificing optimality and completeness.

2.3.5 Adaptive sampling studies using Gaussian Processing

There has been done several recent works on informative and adaptive sampling based upon the work of Guestrin, Andreas Krause, and Ajit Singh (2005), A. Krause et al. (2006), Amarjeet Singh et al. (2006), and K. H. Low, Dolan, and Khosla (2008). Kemna (2018) has made a comparison of several articles on this topic from the last years. The main differences are:

- Are the model off-line or on-line
- Is there used a single robot or are there used multiple robots

- Is the sensors used myopic or non-myopic
- What type of mission; field estimation, target search or mapping.

2.4 Multiple vehicle and collaborative robotic operations

When trying to understand the ocean, it is necessary to use different forms of equipment and technologies. This can be organised in several ways and controlled with different methods. When researching the ocean, it is common to use a ship as a base and control centre for all the different vehicles and sensors. The multi-robot mission done by #OceanRobotsTeam (Costa et al. 2018) does this by using research vessel (R/V) Falkor. Onboard R/V Falkor was scientists and engineers working together to analyse data from deployed unmanned aerial vehicles (UAVs) and AUVs in near real-time and then deciding their next move.

2.4.1 Mission planning

AUVs are used for different missions, and the most common mission objectives in marine applications are:

- Data transportation and communication
- Mapping
- Tracking
- Monitoring
- Search

To achieve these objectives, the AUVs are instructed with a mission plan which is based on several principles. The principles for AUVs define different vehicle commands or behaviour. For motion principles or manoeuvres, this includes dynamic controllers and control strategies. This can, i.e. be trajectory tracking, path following or "go to a waypoint". For teams of vehicles, it is possible to have team manoeuvres where the whole team do the same manoeuvre autonomously. Communication principles will make the vehicles able to broadcast, accept and decode messages.

For a cooperative mission, two main scenarios must be handled by the cooperative behaviour. The vehicles must not collide, while the vehicles simultaneous presence is exploited. Glotzbach, Eckstein, and Ament (2015) has done a study to prevent collision and in that way, execute a safe mission.

By giving the vehicles in a team, constraints based on each other, it is possible to get them to cooperate. Buadu (2018) explains how vehicle A and B can make choices based on a constraint, where they are to be within a radius from each other at all time. As an example, if one of the vehicles get a thruster malfunction, the other vehicle will have to choose between slowing down to stay within the constraint or keep assigned velocity. This is something that needs to be specified in the cooperative strategy and will vary with the goal of different missions.

2.4.2 Mission Language

When choosing how the vehicles are to operate it is important to use a suitable mission language (ML) the use of ML can vary based on the goal of the mission. Human-readable and writable ML makes it easier for an operator to understand and make a mission plan. Eckstein, Glotzbach, and Ament (2013), evaluated three different mission languages. First, the Petri Net which is a Mission Control Language (MCL). Second, Python, which is a scripting programming language often executed within the ROS onboard the vehicle. Third is the use of Extensible Markup Language (XML) which is a description language.

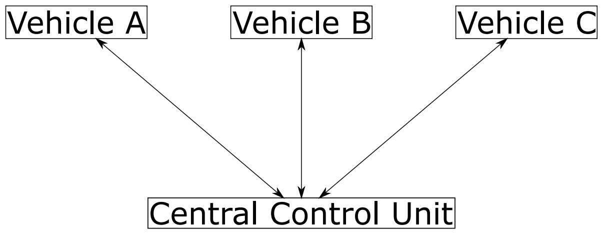
The advantages with Python are that it is human-readable and writable while offering flexibility since every ROS and Python option can be used. The disadvantages of Python are that it has a big overhead which makes the transmission of new mission plans challenging. Another disadvantage of Python is that it does a check after every new receiving of a mission plan or mission update. Since the Python script often is complex every “if” and “switch” statement needs to be checked every time (Eckstein, Glotzbach, and Ament 2013). Buadu (2018) lists XML as a good candidate for ML, based on (Eckstein, Glotzbach, and Ament 2013) evaluation of the three.

2.4.3 Communication

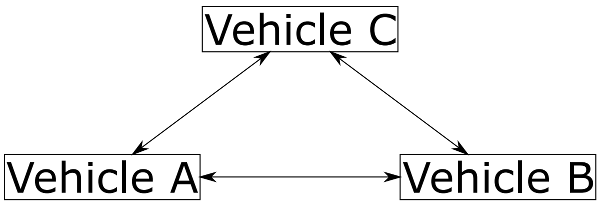
Challenges with AUV missions grows when several vehicles are added to the mix. One of the most dominant challenges is unreliable communication. At the surface, it is possible to have cellular and satellite communication which has high latency and limited bandwidth for data transfer. If they are close enough to each other or a communication-centre, radiofrequency (RF) like Wi-Fi can be used. Subsea communication abilities are limited further. Acoustic communication is an option subsea, but it is prone to interference, disruption and unpredictable delays. The acoustic bandwidth is usually limited, and latency is large (Madureira et al. 2013). The mission plan and cooperative algorithms need to be robust and take the communication faults into account.

When doing multi-vehicle missions formation control is usually used like in Buadu 2018; Costa et al. 2018; Belkin et al. 2018. The formation control can be divided into Central Control Unit (CCU) and decentralized system see Figure 2.3. The decentralized architecture is often used for behaviour based methods Buadu (2018). There are several formation concepts like:

- Behaviour-based: Basic behaviours are given priority, and the controller actions are based on this weighting in each iteration.
- Virtual structures: The usage of nodes in a rigid virtual structure can make a virtual structure where the AUVs collect elements. This works best in two dimensions. Makes it easy to prescribe the coordinated behaviour for a group.
- Leader-follower structure: One of the vehicles is designated as the leader, and the other vehicles follow the leader.



(a) Central control architecture



(b) Decentralized architecture

Figure 2.3: An illustration of how a CCU and decentralized architecture could look like for three vehicles, where the arrows represent the communication links. *Inspired by Buadu (2018).*

2.4.4 Adaptive communication

As mentioned above, there are limitations when doing subsea missions. Since the bandwidth is limited, there are ways to minimise the risk of losing data. A logic-based communication system is one method to reduce the data traffic and robustness between subsea vehicles doing cooperative operations. In Rego et al. (2019), the communication messages are only exchanged when communication triggering conditions like the error being too high occur. This prevents the AUVs to send continuously and reduces the data traffic. To make the system more robust, in regards to packet losses, Rego et al. (2019) implemented a system requiring the AUVs to send a reply whenever they received a message. This allowed the sender to know that the systems had received their message. If after some time the leader AUV has not got any replies it updates the data, gives it a new timestamp and tries to send it again until it has a reply from all AUVs.

Chapter 3

Method

3.1 Simulation environment

For scientists like biologists and oceanographers, different ocean data are of interest. Ocean data are connected, and changes in one type of ocean data can result in changes in another. Ocean temperature affects the density in the ocean and has the highest density at $4°C$. A high density also affects the salinity of the ocean, meaning there is also a correlation between temperature and salinity. Phytoplankton is also affected by temperature, with higher temperature giving a larger growth. The optimal temperature for phytoplankton is generally between $20-24°C$ but generally tolerates temperatures between $16-27°C$. Below $16°C$ the growth decreases and $35°C$ is lethal. The ocean data stored in the SINMOD file (Appendix A) is temperature, salinity and current velocity. Because of the importance of ocean temperature, the temperature is chosen as the most interesting ocean data in this thesis.

The simulation of "LAUV-SIMULATOR-1" and "LAUV-SIMULATOR-2", hereby known as AUV 1 and AUV 2, is run on a computer with Linux Ubuntu 16.4 operating system. The Python program using PyIMC connects to DUNE running in two separate terminal windows on the simulation computer. The ocean data used to simulate the environment and to make a prediction of the area are stored in the SINMOD file (Appendix A). Receiving the ocean data is done by the oceanserver code (Appendix A) using the socket package in Python to send position from the main programs to the server and receiving the ocean data. Neptus is used to move the AUVs to their starting position before running the Python script and to visualize the mission while it is running. In the next section, the ocean model will be described further, while the Python code will be further described in the following sections.

3.1.1 Ocean model

The ocean model used in the simulation is data from the SINMOD ocean model. SINMOD is a model system developed since the 1980s by SINTEF. The engine of the model is a 3D hydrodynamic model, based on Navier-Stokes equations solved by finite differences method on a regular Arkawa C-grid (*SINMOD* 2020). The hydrodynamic model is based on the model described in Slagstad and McClimans (2005).

By using finite difference techniques with a z-coordinated regular grid with square cells, the hydrodynamic components, which are based on the primitive equations, are solved (Fossum 2019). Fossum (2019) writes in his PhD that SINMOD has been used for ocean circulation and ecosystem studies along the Norwegian coast and in the Barents Sea, in ecosystem risk assessment studies, in kelp cultivation potential and climate change effects studies. An example of the model output used in this thesis can be seen in Figure 3.1.

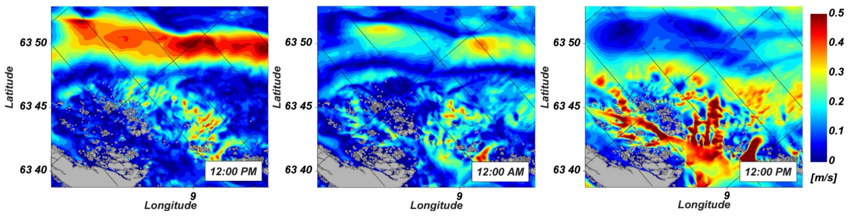


Figure 3.1: Surface current speeds from the SINMOD file outside the Frøya region. Predictions are from the 5th and 6th of May, 2017. *Courtesy of Fossum (2019).*

3.1.2 Mission area

The ocean model data from SINMOD used in this thesis is from an area outside the Frøya area in the middle of Norway, see Figure 3.1. The data is from 05.05.2017-06.05.2017, containing information about salinity, temperature and current. It covers a large area of approximately 20×20 km. The data is stored in the file SINMOD file (Appendix A) and can be retrieved using the Python package "netCDF4", used in the ocean server.

For simplicity, both in regards to simulation time and computation time, the area used in this thesis is set to be 10×10 km, starting in the centre of the total ocean model area. The area is so divided into 50×50 grid points resulting in a distance between each grid point of 200 meters. The focus of this thesis is to look at the temperature. All temperatures considered in this thesis can be seen in Appendix B. To be able to compare the results from a mission, the temperature is gathered in a 50×50 grid. This means that the data in the grid is only valid for the latitude and longitude for each grid point and do not take into consideration the values between each grid point.

When estimating the temperature in the grid area pre-mission, the longitude and latitude of all grid point are sent to the oceanserver. The temperature, salinity, velocity and acceleration of the current is received for each grid point. The AUVs gather their data in the same way and send their position to the server every second through the simulation time, gathering data. The temperature measured between two grid points is then divided into two, and the mean of the measured temperature is stored to the previous waypoint and the current waypoint.

3.1.3 Pre/post processing

Before starting the main program a pre-mission program (Appendix A) is run. The program is used to find the start coordinates for AUV2 based on start position for AUV 1 to be used in Neptus. Predicted mean temperature, as seen in Figure 3.8a, used in the GP onboard the AUVs are stored to a text file. It also stores the wanted temperature grid for the given mission time and day to a text file to be used as a comparison to the predicted field at the end of the mission.

After the main program is stopped the plot program (Appendix A) is run. The main task of the program is to plot the different plot shown in chapter 4 of this thesis. The program also calculates an evaluation of the results. By taking the actual temperature grid at the given time of the mission and subtracting the predicted grid from the GP and summing the absolute value of the matrix, we get a total error of the predicted field. The total error is then divided by the number of grid points, resulting in the mean absolute error (MAE). This method is also used for the actual temperature grid and subtracting the mean temperature grid shown in Figure 3.8a. For actual temperature, at day one at 13:00, the MAE is $0.03046°C$.

3.2 LSTS Toolchain

When controlling multiple vehicles in a mission, there are several methods to do so. In this project, the focus has been on the LSTS Toolchain. The toolchain from Laboratório de Sistemas e Tecnologia Subaquática (LSTS) is an open-source software toolchain. The software can be used for mixed-initiative control of networked heterogeneous unmanned scenarios for the ocean and air systems. It is capable of handling communication challenged by the environments (Costa et al. 2018). The system is capable of doing multi-vehicle surveys as tested in Pinto et al. (2013) where two LAUVs were used together to do a side-scan and bathymetry mapping in half the time it would be possible doing it using only one AUV. The report also shows that the LSTS system can run onboard deliberative planning like having a TREX program running side-by-side with DUNE to be able to create a plan based on operational constraints.

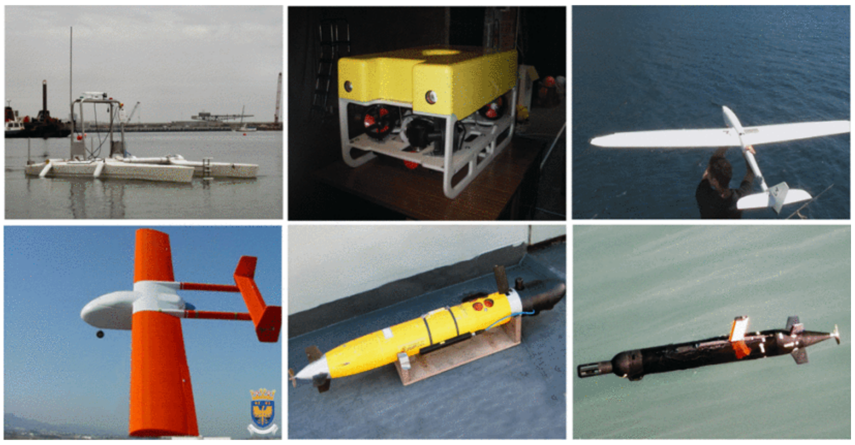


Figure 3.2: Picture of supported vehicles supported by the LSTS toolchain. *Courtesy of Pinto et al. (2013).*

3.2.1 Inter-Module Communications protocol

All components of the LSTS Toolchain uses the same communication protocol; The Inter-Module Communications protocol (IMC). It is an XML-based message-oriented communication protocol, and all processes and devices use it to communicate by exchanging IMC messages. This makes it possible for all types of vehicles and computer nodes in the networked environment to understand each other. The messages can include chunks of data streams or represent an asynchronous event to be handled by other components (Pinto et al. 2013). The shared set of messages can be serialized and transferred over different means like UDP, TCP, HTTP, acoustic modems or Iridium. All messages in IMC are time-tagged and include information about its origin and destination sub-system Ferreira et al. 2019.

3.2.2 DUNE

The embedded system uses DUNE Unified Navigation Environment (DUNE) as the on-board software. The software is written in C++ and runs on Linux, QNX, Solaris, Mac OS, eCos, RTEMS, OpenBSD, FreeBSD, NetBSD and Microsoft Windows (Pinto et al. 2013). This is the heart of the vehicles as embedded software. It includes modules for control, navigation, simulation, networking, sensing and actuation. The different tasks are divided into different categories (Pinto et al. 2013):

- Sensors: These tasks are handling the hardware for measuring the environment.
- Actuators: drivers handling hardware interacting with the environment, like making the vehicle move.
- Estimators: These tasks take the information from sensors and changes it into state estimations, like for the navigation task.
- Controllers: These tasks transform high-level commands into lower-level commands according to the estimated state, i.e. all manoeuvres have a manoeuvre controller.
- Monitors: These tasks receive information from other tasks and can change the vehicles state. For instance, if the operational limits are breached, the vehicle mode will be changed to “blocked”.
- Supervisors: These tasks use the vehicle stat to enable and disable other tasks. If the vehicle is in “blocked” mode, the manoeuvre controller will stop sending commands.
- Transports: These tasks transports messages in and out of the message bus. Some of the transport tasks are logging, UDP, TCP, and HTTP.

3.2.3 Neptus

Neptus is a command and control infrastructure for the operation of all unmanned vehicles supported by the LSTS Toolchain. It supports, planning, simulation, execution and post-mission analysis. It has a comprehensive plug-in infrastructure, which allows for quick adaptation to fit mission-specific requirements by operators and developers (Costa et al. 2018). The software is written in Java and runs on Linux and Microsoft Windows where it uses the IMC to command the different systems. It is possible to add plug-ins for Neptus, i.e. for deliberative planning capabilities, there is a TREX plug-in.

Figure 3.3 show the console in Neptus. By dividing it into left and right, we see the map on the left side with the virtual representation of the mission cite. There are two LAUVs within the boundary set. The map tile can be chosen from different sources like Open street map, Google Maps, or Virtual Earth. On the right side is the interface where the console provides several planning components and commands (Pinto et al. 2013).

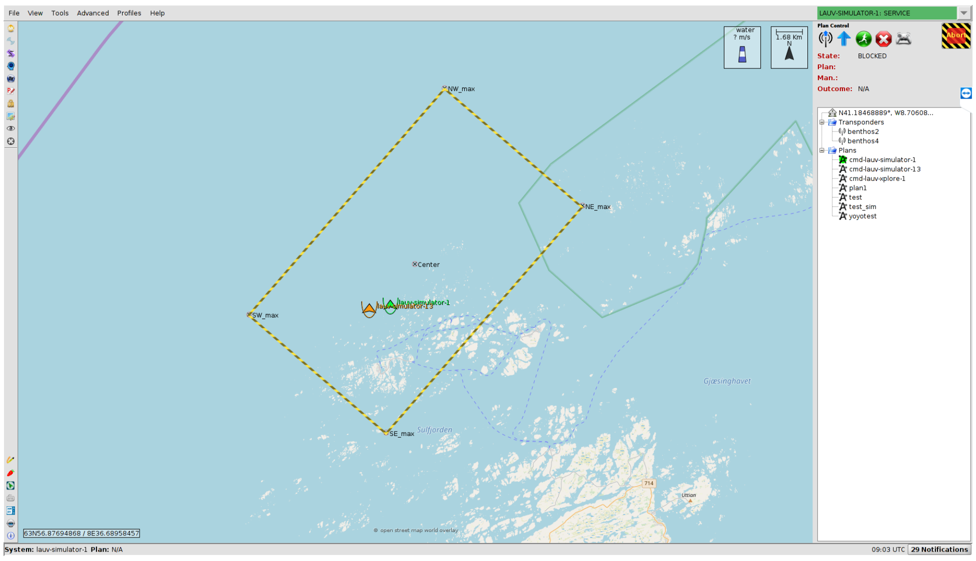


Figure 3.3: A screenshot of Neptus in the console view.

In this thesis Neptus is mainly used to visualise the mission while the mission is running in PyCharm (*PyCharm* 2019), see Figure 3.4. The program is also used to send a GPS Fix to AUV 2 to get it in position before the mission can start. In this thesis, all missions are run with AUV 1 starting in grid position [0,0] and AUV 2 starting in position [0,49]. In Figure 3.4 this means that AUV 1 starts in the point called Centre and AUV 2 starts in the point marked AUV2_start.

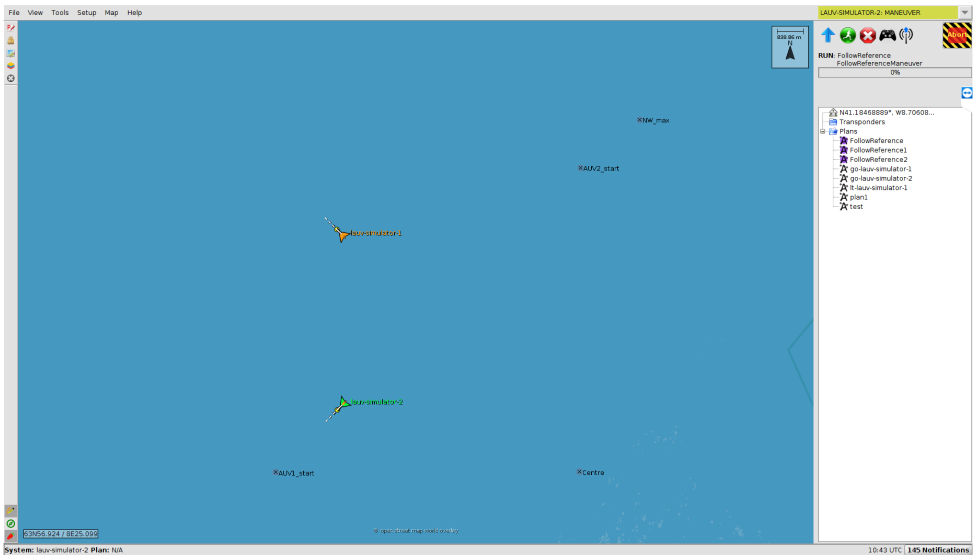


Figure 3.4: Screenshot from the LSTS software Neptus, showing how the mission is visualised.

Neptus can also be used post-mission to analyse the mission. The program can be used to analyse several things. Some of the things that can be seen in neptus are; messages sent to and from the AUVs, bathymetry, CTD and battery usage. In this thesis, the analysing tool in Neptus is only used for missions where the path of the AUVs has not been possible to plot, or other errors have occurred during a mission.

3.2.4 PyIMC

"PyIMC - Python binding for Inter-Module Communication Protocol (IMC) used to communicate between modules in the LSTS toolchain", Sture 2019.

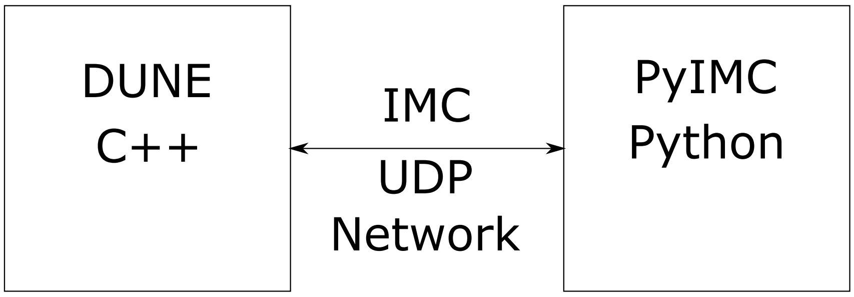


Figure 3.5: Illustration of how PyIMC works together with DUNE.

IMC is the message protocol for the LSTS toolchain. The PyIMC makes it possible to create mission plans in Python. By using the PyIMC package, it is possible to run the Python codes onboard the AUVs to control them. This makes it possible to make more adaptive missions by using common Python commands together with the IMCs, to both get data from the AUVs and give new instructions to the AUVs based on that. The code written for the simulations in this master thesis is based upon Øystein Stures example codes, especially the "follow_reference_example.py" (Sture 2019).

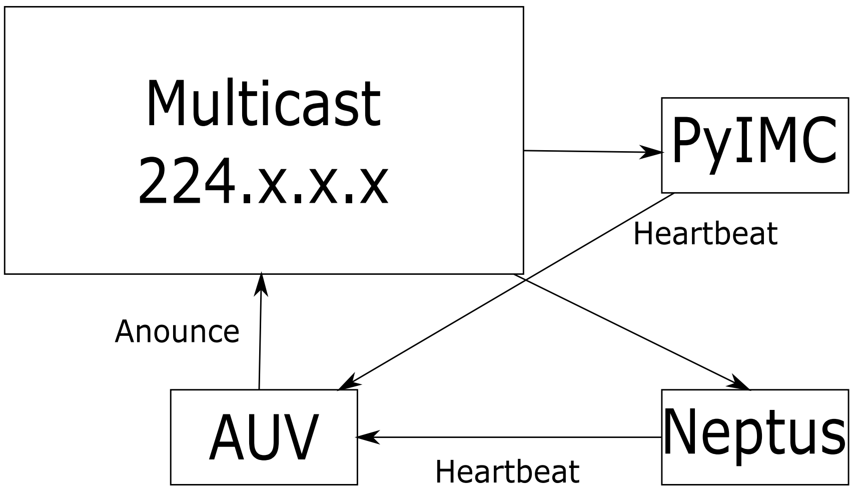


Figure 3.6: Illustration of how Multicast is in centre of PyIMC, Neptus and the AUV.

3.2.5 IMC Messages

The IMC protocol comprises seven logical message groups (Martins et al. 2009):

1. Mission control - defines the mission.
2. Vehicle control - monitors vehicle state and issues manoeuvre commands or external requests
3. Manoeuvre - defines manoeuvres, manoeuvres commands, and execution state
4. Guidance - guidance in autonomous manoeuvring

5. Navigation - defines an interface for reporting on the vehicle's navigation state
6. Sensors - reports sensor readings
7. Actuators - specify hardware actuator controllers

A description of all IMC messages can be found at the LSTS GitHub https://goo.gl/ANUUBw. The essential IMC messages used by PyIMC in this thesis are as follows:

Estimated State is a navigation message and presents the estimated state of the vehicle. The message contains a complete description of the system with parameters such as position, orientation and velocity at a particular moment. Where the data fields of interest are; latitude, longitude and z-position.

Follow Reference State is a manoeuvring message that identifies the source system allowed to control the vehicle.

Plan Manoeuvre is a plan supervision message used to set info about the mission plan, like the missions manoeuvre_id.

Plan Specification is a plan supervision message used to identify the plan, giving it a plan_id and a description.

Plan Control is a plan supervision message used to request and start plans.

Heartbeat is a networking message used as information about the entity's system running normally and are alive.

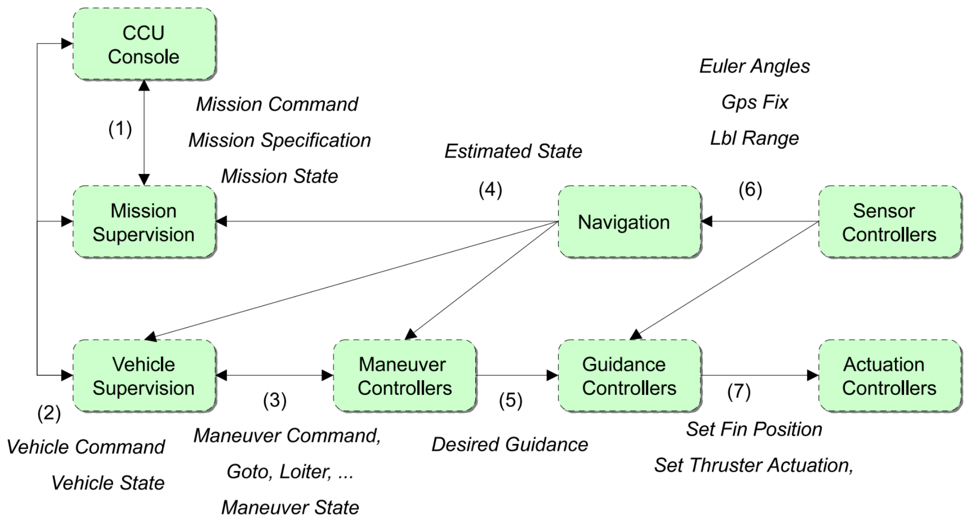


Figure 3.7: Shows how the IMC messages connects. Adapted from Martins et al. (2009).

3.3 Proposed Algorithm

The pseudocode of the algorithm used onboard the AUVs can be seen in algorithm 1. In addition to the stages in the pseudocode, there are used some additional features included in the PyIMC. The function @*Subscribe()* is used to receive continuous information from the AUVs, like *Estimated state* and *Follow Reference State*. @*Periodic()* is another function used to call a definition to run at an interval, given in seconds. This is used to get ocean data every second, to update the info in the mission plan every ten seconds, and to send periodic references to the AUV.

Algorithm 1: Pseudocode AUV

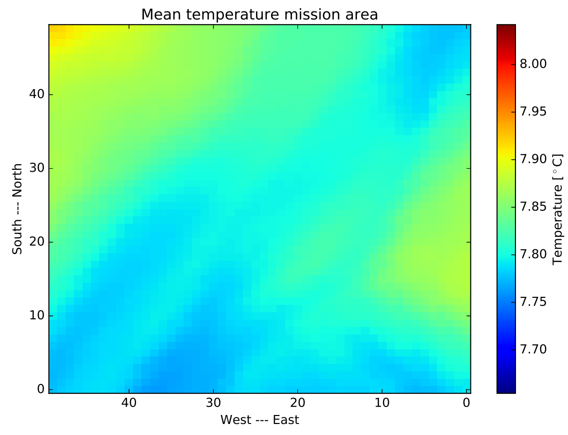
```
import packages;
get hartbeat;
get EstimatedState;
send first reference point;
go to first reference point;
while able to find a path to wanted destination do
    store temperature measured from ocean data to grid;
    add information gained from the other AUV;
    calculate prediction and uncertainty using GP;
    find next waypoint;
    if unable to find waypoint then
        abort mission;
    end
    send next waypoint to AUV;
    send information to the other AUV;
    go to next waypoint;
end
```

3.3.1 Gaussian Processing

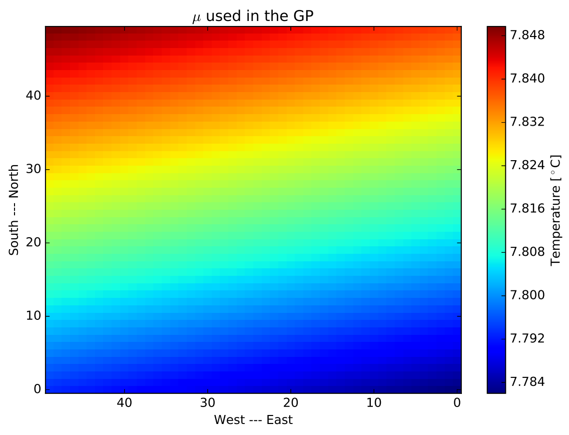
The GP used in the algorithm onboard the AUVs is developed by Trygve Olav Fossum, researcher at NTNU (Fossum 2020). A licence for the program is attached together with the Python-files of this thesis (Appendix A). The GP uses Equation 2.2 and Equation 2.3 to calculate an estimate of the temperature $\mu_{posterior}$ and the uncertainty $\Sigma_{posterior}$. The kernel is calculated as follows:

$$cov\left(\boldsymbol{s}_i, \boldsymbol{s}_j\right) = \sigma^2 e^{-3 \cdot \frac{1}{\ell} \cdot \|\boldsymbol{s}_i - \boldsymbol{s}_j\|} \tag{3.1}$$

where the variance σ^2 is set to 0.04, and the correlation distance ℓ is set to 70. This is default values from Trygve Olav Fossums program.



(a) Predicted temperature used in GP



(b) The μ used in the GP.

Figure 3.8: The gradient temperature used as input for the GP and the resulting mu for the mission area.

As prediction input to the GP, the mean of all temperatures shown in Appendix B are used, as seen in Figure 3.8a. Based on the mean temperature, the GP calculates a prediction μ, shown in Figure 3.8b. This is used in the further calculation of the predicted temperature. The program then finds the distance between and weighting of each point in the grid. Based on the measured data, the prior estimate μ and the covariance, the prediction and uncertainty of the grid is calculated. This is done every time the AUVs reaches their destination and used in the decision model when deciding where to go next.

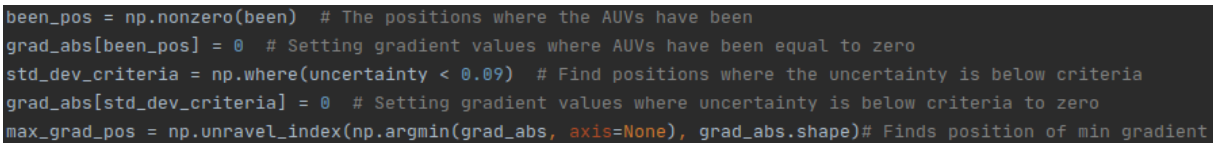
3.3.2 Decision model

One of the main objectives of this thesis is to test different decision methods for adaptive sampling. The goal is for two AUVs working together to map the mission area to find the most informative areas in an efficient way. To find the mission route, both AUVs have a decision model. The general algorithm for the decision model is myopic and can be seen in algorithm 2. The myopic (greedy) approach was chosen because of the limitation in computation power onboard AUVs. It is also easier to implement and understand, and still get a good result. The primary input data is the AUVs current position, a grid showing where both AUVs have been and the temperature prediction and the uncertainty grid from the GP. In addition, the destination of the other AUV is also added as an input for some of the later simulations. Based on the input, the simulations search for high or low temperature or they search for high or low temperature gradient when deciding where to go next.

Algorithm 2: Decision model

```
Input: AUV positions, Temperature prediction grid, AUV been grid, uncertainty
       grid
Data: Decision criteria
if AUV been at position then
    set position value above max or below min;
end
if Other AUVs destination then
    set area values above max or below min;
end
if uncertainty < criteria then
    set position value above max or below min;
end
Find position of max or min in considered grid;
Use A* to find path to max or min position;
if length(path) = 0 then
    Send abort signal to AUV;
else
    Send next step in path to AUV;
end
```

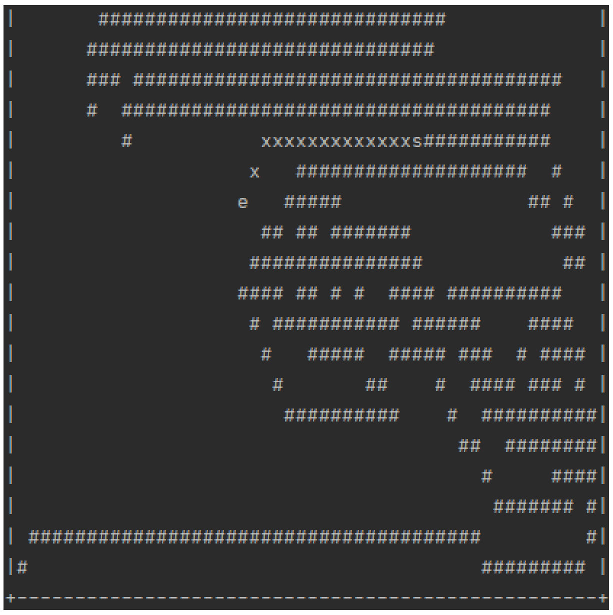
When deciding where the highest temperature or the highest gradient value is the program uses the built-in functions in the Pythons "NumPy" package (Walt, Colbert, and Varoquaux 2011). Before the max/min function is carried out, the decision criteria is taken into account. To prevent the AUVs checking positions already checked the grid values where they have been are set to 0 if looking for max values and to 1000 if looking for min values. To prevent the AUVs to check the whole area, and not gaining the advantages of the GP and the adaptive sampling, the grid values where the uncertainty is below a given uncertainty criteria is set to 0 or 1000. An example of this can be seen in Figure 3.9. In some cases, the area is also split into two or four regions. For the two regions setup, the grid is just split in half by setting all values for the opposite area equal to zero. By letting the AUVs communicate their destination, and by splitting the grid into four regions, it is possible to prevent them from tracking the same area. This is done by setting the grid value for the area the other AUV is going to, to 0 or 1000.



```
been_pos = np.nonzero(been)  # The positions where the AUVs have been
grad_abs[been_pos] = 0  # Setting gradient values where AUVs have been equal to zero
std_dev_criteria = np.where(uncertainty < 0.09)  # Find positions where the uncertainty is below criteria
grad_abs[std_dev_criteria] = 0  # Setting gradient values where uncertainty is below criteria to zero
max_grad_pos = np.unravel_index(np.argmin(grad_abs, axis=None), grad_abs.shape)# Finds position of min gradient
```

Figure 3.9: Example of how the decision of max/min is carried out. Screenshot of "Sim_1.py" in PyCharm (Appendix A).

After the max or min position is found the AUV uses the A* method to get to the position. The A* pathfinding method is used by including the "PyPI" "Pathfinding" package (Bresser 2020). The A* is a path search algorithm based on the Dijkstra algorithm to find the shortest or fastest path from start to end. A* achieves better performance by using heuristics when planning the path but also has the drawback of being computational heavy since it stores all generated nodes in memory. To prevent the AUVs tracking the same position twice the positions where the AUVs have been are treated as obstacles. The AUV then finds the path to the wanted position. The next step in the path is then sent to the AUV as the next waypoint. If the AUV is not able to find a path to the wanted position, due to not finding a way through the obstacles, the AUV aborts the mission.



```
|       ##############################             |
|      ##############################              |
|      ### #####################################   |
|      #  #####################################    |
|         #           xxxxxxxxxxxxxs###########    |
|                    x   ####################  #   |
|                   e   #####                ## #  |
|                     ## ## #######            ### |
|                    ###############            ## |
|                   #### ## # #  #### ##########   |
|                    # ########### ######    ####  |
|                     #   #####  ##### ###  # #### |
|                      #       ##    #  #### ### # |
|                       ##########    #  ##########|
|                                      ##  ########|
|                                        #     ####|
|                                         ####### #|
| #######################################         #|
|#                                       ######### |
+--------------------------------------------------+
```

Figure 3.10: Example from the terminal window of PyCharm, showing the A* method. The # is the obstacles where the AUVs have been, x is the path, and s and e stand for start and end.

3.4 Simulation setup

The missions are done within the mission area, as described in subsection 3.1.2. The x-axis of the grid is negative meaning $x = 0$ position is in the east corner. The y-axis is positive meaning the $y = 0$ position is in the south. AUV 1 starts in southeast corner at grid position $[0, 0]$. AUV 2 starts in northeast corner at grid position $[0, 49]$. Both vehicles start by going to a set waypoint before they start doing the adaptive sampling. AUV 1s first grid point is $[1, 1]$ and for AUV 2 it is $[1, 48]$.

Communication between the AUVs only happens in the Python interface in PyCharm (*PyCharm* 2019) and there is not made an algorithm to simulate a realistic communication system for the AUVs. This means that both AUVs have access to all the text files the other AUV produces at all times. The text files are only updated by an AUV when it is near the next waypoint and goes through the storing of data, GP modelling and finding next waypoint. Data is stored to text files and retrieved from text files by using "NumPys" inbuilt functions "numpy.savetext" and "numpy.loadtext".

Procedure when starting simulation (more info about each Python program can be seen in Appendix A):

- Start DUNE for both vehicle in the terminal windows.

- Run the oceanserver.py code to set up the socket for the ocean server.
- Run the Pre-Mission.py code.
- Position both vehicles in Neptus by using the GPS Fix command.
- Run the main.py code.
- Wait for the simulation to run.
- Run the plot.py to plot the results.

3.4.1 Sim 1

In this simulation, AUV 1 has an uncertainty constrain of 0.12, and AUV 2 has a constrain of 0.06. Both AUVs take into account where both AUVs have been by setting an obstacle in every visited position. This is to prevent the A* method finding a path over the grid points already checked. The results can be seen in section 4.1.

3.4.2 Sim 2

In this simulation, the AUVs are looking for different temperatures. AUV 2 is looking for the lowest temperature while AUV 1 looks for the highest temperature in the grid. When an AUV is looking for the lowest temperature the positions that are not to be checked can not be set equal to 0 but needs to be set equal to a negative number, -1000 is chosen. The uncertainty cap is set to be 0.09 for both vehicles. This to try to cover areas closely without trying to reach every point around a hot spot. The results can be seen in section 4.2.

3.4.3 Sim 3

Also, in this simulation, the two AUVs looks for different temperatures. AUV 1 looks for low temperatures while AUV 2 looks for high temperatures in the grid. The uncertainty cap for this mission is set to 0.09 for both vehicles. The results can be seen in section 4.3.

3.4.4 Sim 4

For this simulation, the grid is separated into an upper and lower part. This is done by setting all gridpoints below 24 for AUV 1 equal to zero and every gridpoint above 24 equal to zero for AUV 2. The uncertainty cap for this simulation is also set to 0.09 for both vehicles. The results can be seen in section 4.4.

3.4.5 Sim 5

In this simulation, the AUVs takes the grid point the other AUV is trying to reach into account when planning where to go. This is done by dividing the grid into four equally large squares 25×25. When the AUV plans where to go next, it checks where the other AUV is going, by setting every gridpoint of the square the other AUV is going to equal to zero the AUV doing the planning chooses a max temperature position in another region. In this simulation, AUV 2 takes the destination of AUV 1 into account at the second iteration while AUV 1 takes the destination of AUV 2 into account at the third iteration. Uncertainty cap is set to 0.09 for both AUVs. The results can be seen in section 4.5.

3.4.6 Sim 6

This simulation is almost the same as simulation five described above in subsection 3.4.5. Both AUVs takes the other AUVs destination into account, and the uncertainty cap is at 0.09 for both AUVs. The difference is that in this simulation, AUV 1 takes AUV 2s destination into account at the second iteration, while AUV 2 starts to take the destination of AUV 1 into account at the third iteration. The results can be seen in section 4.6.

3.4.7 Sim 7

In this simulation, the gradient of the temperature grid is calculated and used as input for the decision method. The gradient is calculated by using the gradient function in the "NumPy" package (Walt, Colbert, and Varoquaux 2011). Both AUVs looks for the highest absolute value of the gradient in the grid. The uncertainty cap for the AUVs is set to 0.08 for this simulation. The result can be seen in section 4.7.

3.4.8 Sim 8

In this simulation, the same method as used in simulation 5 and 6 is used to consider the other AUVs destination. This is still done by dividing the grid into four equally large squares and setting the square where the other AUV is going equal to zero. The uncertainty cap in this simulation is set to 0.09 for both AUVs. Results can be seen in section 4.8.

3.4.9 Sim 9

For this simulation, the AUVs do not look at the absolute value of the gradient. AUV 1 is looking for the lowest gradient, and AUV 2 is looking for the highest gradient. The AUVs was sent in a cross before starting looking for high and low gradients, to solve a problem where the AUVs could only find its position. This was done by setting the value at grid position $[49, 49]$ equal to -1000 for AUV 1 and grid position $[49, 0]$ equal to 1000 for AUV 2 the first 50 iterations. The uncertainty cap for this simulation was set to be 0.09 for both vehicles. The results can be seen in section 4.9.

3.4.10 Sim 10

Since the results from simulation 9 seemed interesting, the cross-action was tried again. For this simulation, the AUVs looks for the absolute value of the gradient again after the 50 first iterations. Both AUVs are looking for the highest absolute value of the gradient, and the uncertainty cap is set to 0.09. The method to prevent the AUVs to check the area the other AUV is going to is also implemented. The results can be seen in section 4.10.

3.4.11 Sim 11

The cross-action is used in this simulation to compare the results from simulation 9 and 10 to the early missions only looking for temperature. Both AUVs are to look for the highest predicted temperature in the grid. The AUVs take into account the other AUVs destination, and the uncertainty cap is set to 0.09 for both vehicles. The results can be seen in section 4.11.

3.4.12 Sim 12

This simulation is the same as in Simulation 11, but this time the AUVs are not communicating. This means that it is not able to check where the other AUV is going. It also means that the AUV does not have access to the other AUVs gathered temperatures or positions. Both AUVs are searching for the highest temperature in the grid, and both AUVs have an uncertainty cap of 0.09. The results can be seen in section 4.12.

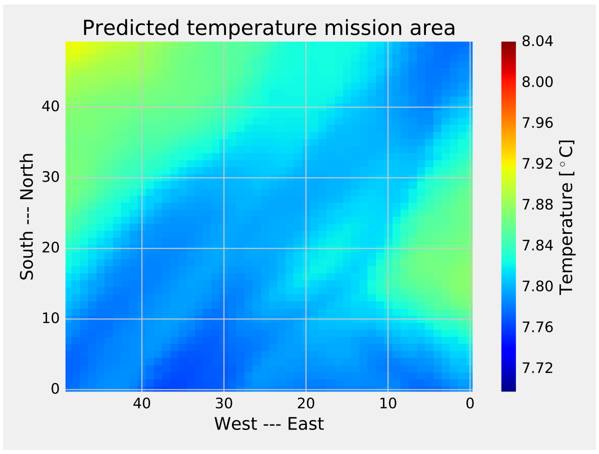
Chapter 4

Results

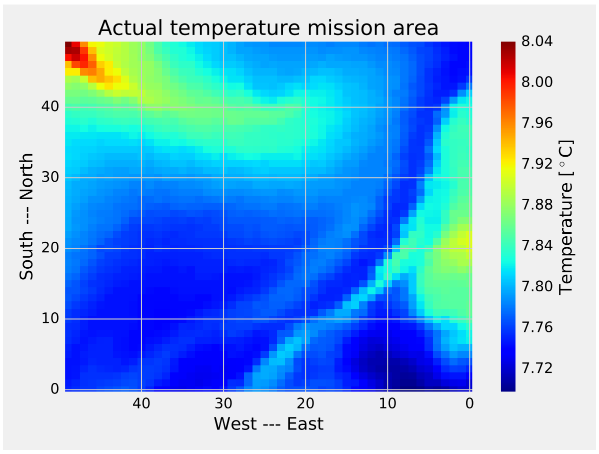
In the majority of the simulations, both AUVs are stopped at the same time. Therefore they have the same uncertainty and prediction plots. This is because both AUVs calculate the uncertainty and prediction based on both vehicles measurements. Therefore, plots only from one AUV will be shown in results. In cases where uncertainty and prediction are different, the plots from both vehicles will be shown. The results from the simulation are plotted in Python. For each simulation, the max and min temperature for all data is checked, and all plots except the uncertainty plot are plotted with the same max and min temperature value. For every simulation, MAE is calculated to evaluate the results, as described in subsection 3.1.3. This error is the sum of the absolute value of the difference between the actual temperature and the predicted temperature, divided by total grid points in the grid. By comparing the number against the sum of the MAE between the actual temperature and the predicted temperature, this tells something about how accurate the predicted field is. The MAE between the actual temperature and the predicted temperature is $0.03046°C$.

4.1 Sim 1

The simulation ran for 10 hours before the reference source in PyCharm timed out. AUV 1 tracked 248 grid points, while AUV 2 tracked 241 points. This resulted in a MAE between true temperature and predicted temperature to be $0.01932°C$.



(a) Predicted temperature used in GP, sim 1



(b) Actual temperature for mission area, sim 1

Figure 4.1: Predicted temp and actual temp for mission area, sim 1

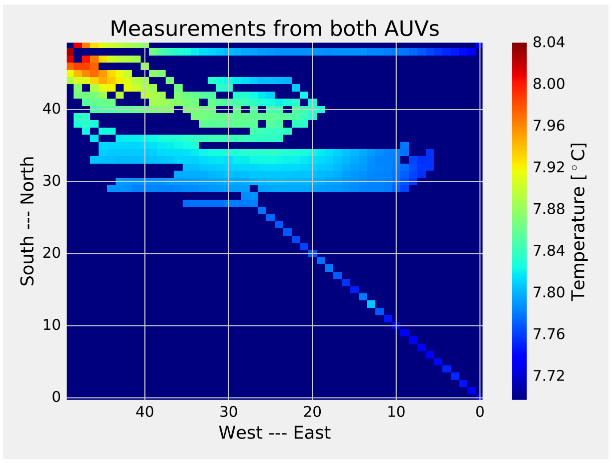
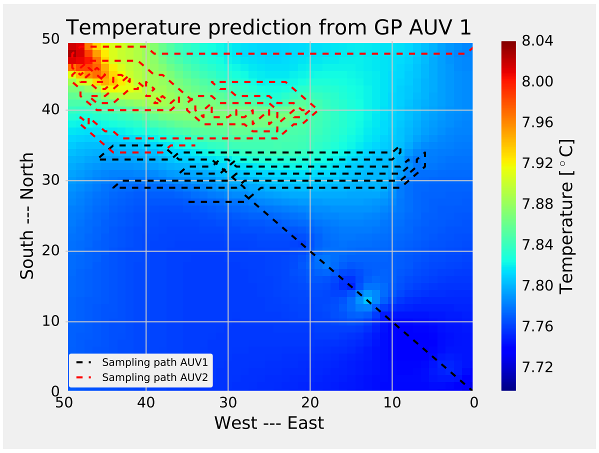
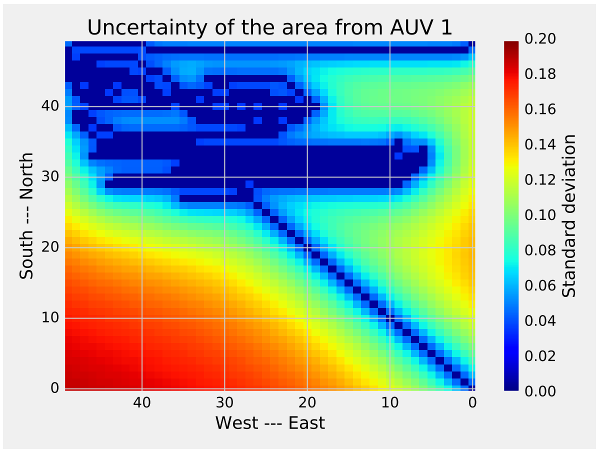


Figure 4.2: Measured temperature from both AUVs, sim 1



(a) Temperature prediction from GP, sim 1

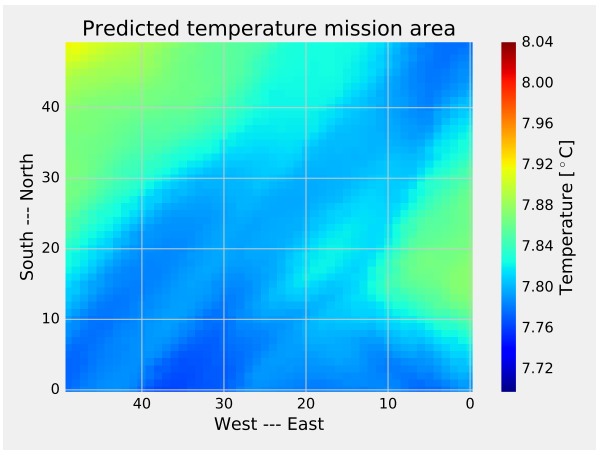


(b) Uncertainty from GP, sim 1

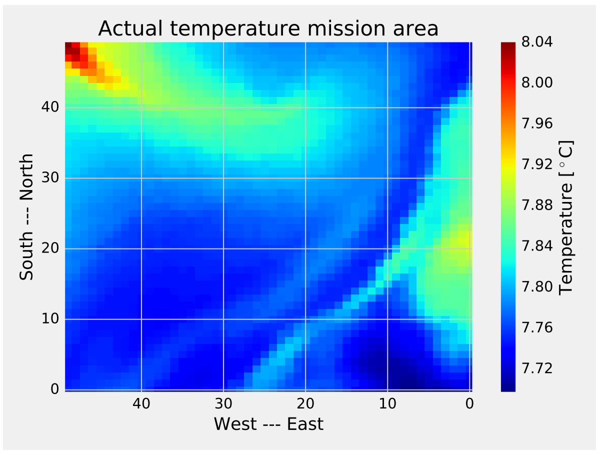
Figure 4.3: The results from the GP, sim 1

4.2 Sim 2

The simulation runs for approximately 12 hours and 30 minutes. AUV 1 run a little longer than AUV 2, and both stopped because of source time out in PyCharm. This resulted in AUV 1 tracking 287 grid points while AUV 2 only tracked 263 grid points. The MAE between true temperature and the predicted temperature for AUV 1 was $0.01825°C$.



(a) Predicted temperature used in GP, sim 2



(b) Actual temperature for mission area, sim 2

Figure 4.4: Predicted temp and actual temp for mission area, sim 2

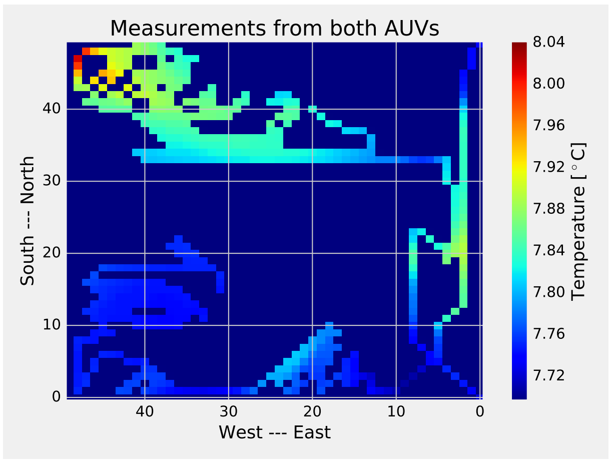
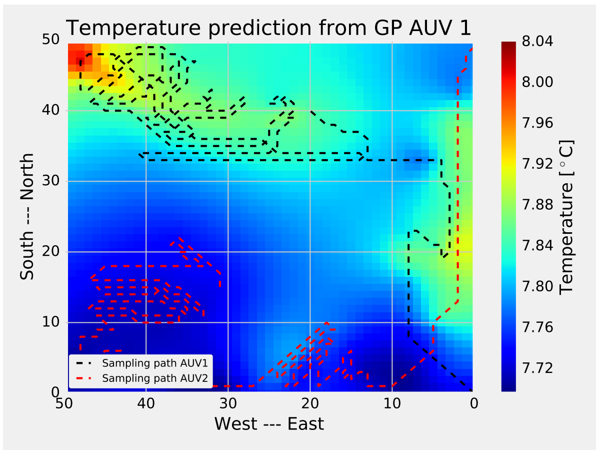
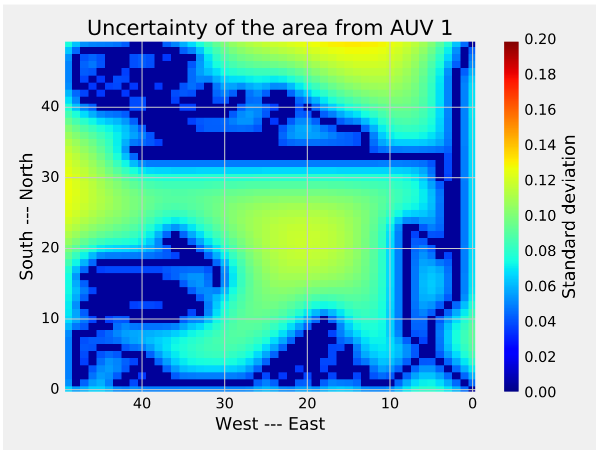


Figure 4.5: Measured temperature from both AUVs, sim 2



(a) Temperature prediction from GP, sim 2

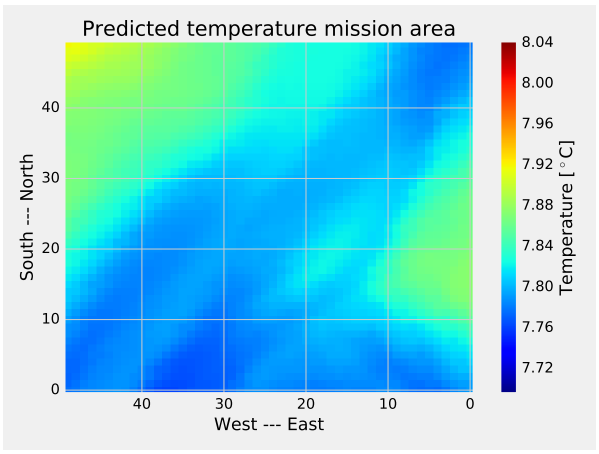


(b) Uncertainty from GP, sim 2

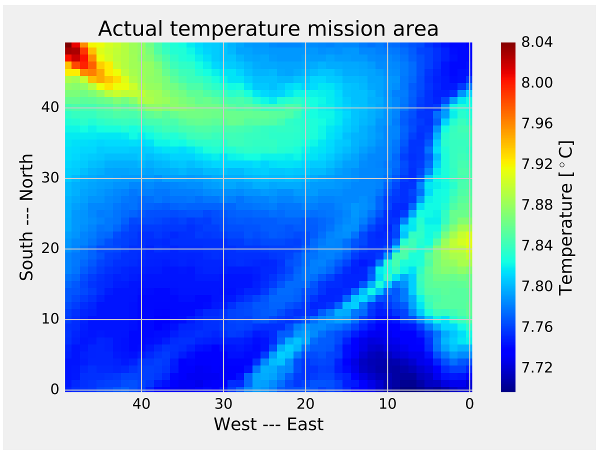
Figure 4.6: The results from the GP, sim 2

4.3 Sim 3

Both AUVs stops after approximately 12 hours due to the reference source timed out. AUV 1 tracked 291 grid points while AUV 2 tracked 284 points. The MAE between true temperature and the predicted temperature was 0.01707°C for AUV 1.



(a) Predicted temperature used in GP, sim 3



(b) Actual temperature for mission area, sim 3

Figure 4.7: Predicted temp and actual temp for mission area, sim 3

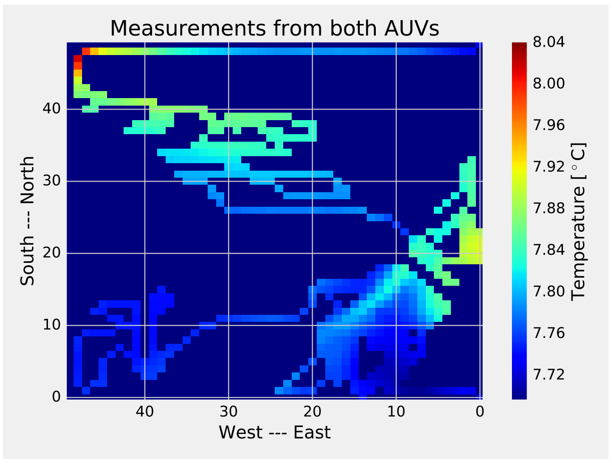
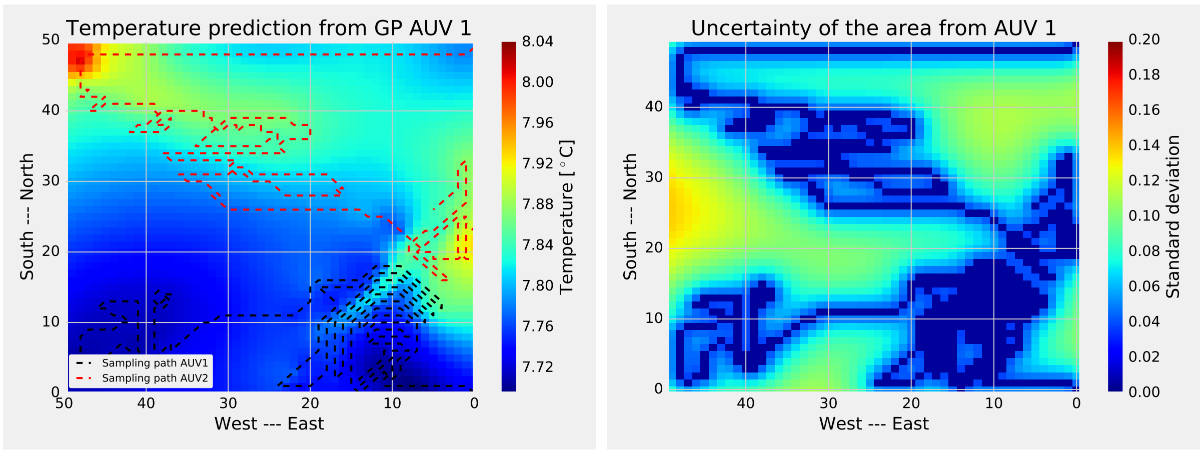


Figure 4.8: Measured temperature from both AUVs, sim 3

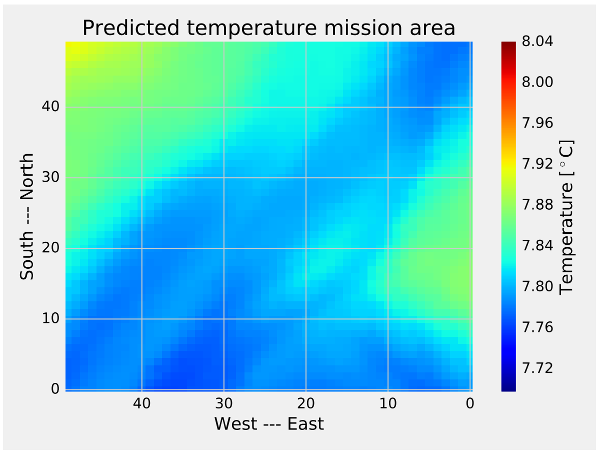


(a) Temperature prediction from GP, sim 3 (b) Uncertainty from GP, sim 3

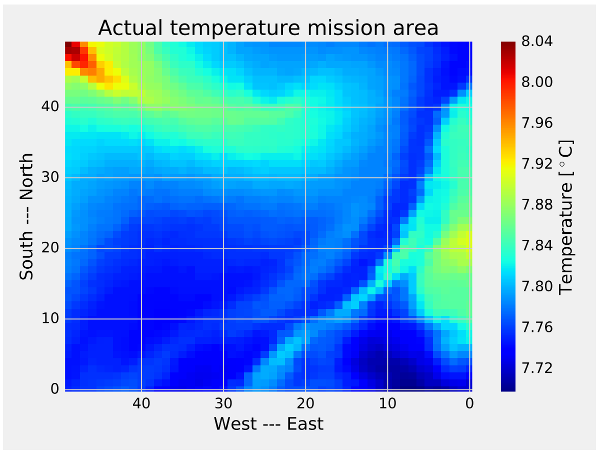
Figure 4.9: The results from the GP, sim 3

4.4 Sim 4

AUV 2 aborted the simulation after 8 hours, while AUV 1 aborted the mission after 11 hours and 45 minutes. This resulted in AUV 1 tracking 262 grid points, while AUV 2 tracked 192 points. The results from AUV 1 shows that the MAE between the true temperature and the predicted temperature was $0.01800°C$, while for the predicted temperature in AUV 2, MAE was $0.01658°C$.

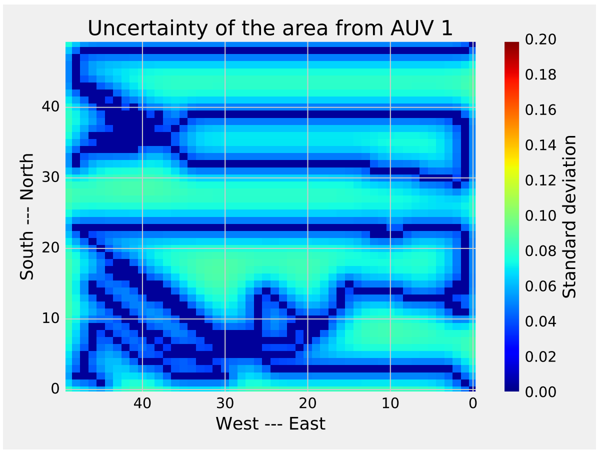


(a) Predicted temperature used in GP, sim 4

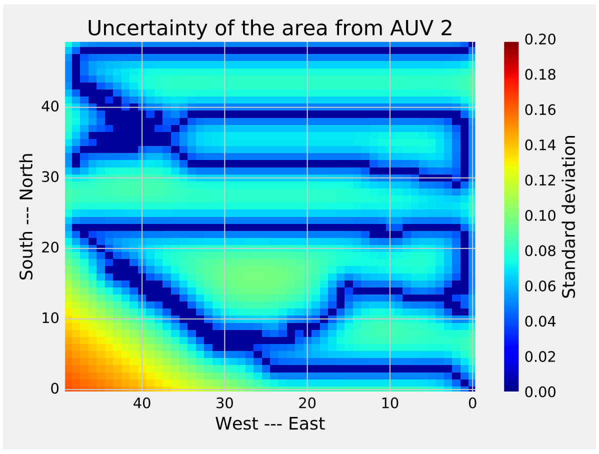


(b) Actual temperature for mission area, sim 4

Figure 4.10: Predicted temp and actual temp for mission area, sim 4

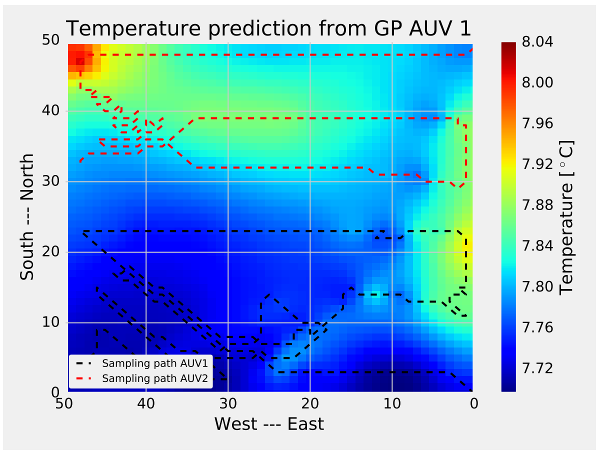


(a) Uncertainty from GP AUV 1, sim 4

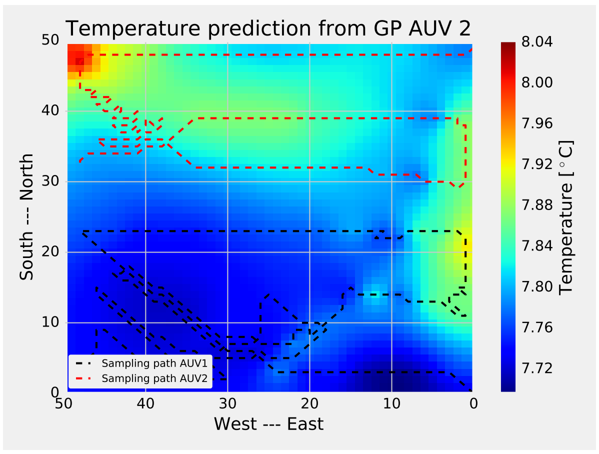


(b) Uncertainty from GP AUV 2, sim 4

Figure 4.11: Uncertainty both AUVs, sim 4



(a) Temperature prediction from GP, sim 4



(b) Temperature prediction from GP, sim 4

Figure 4.12: The results from the GP, sim 4

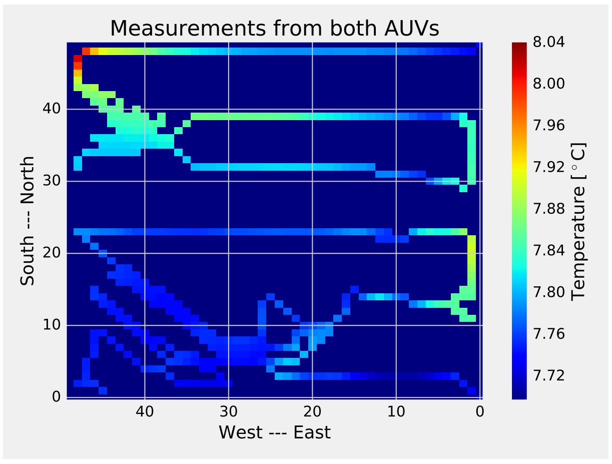
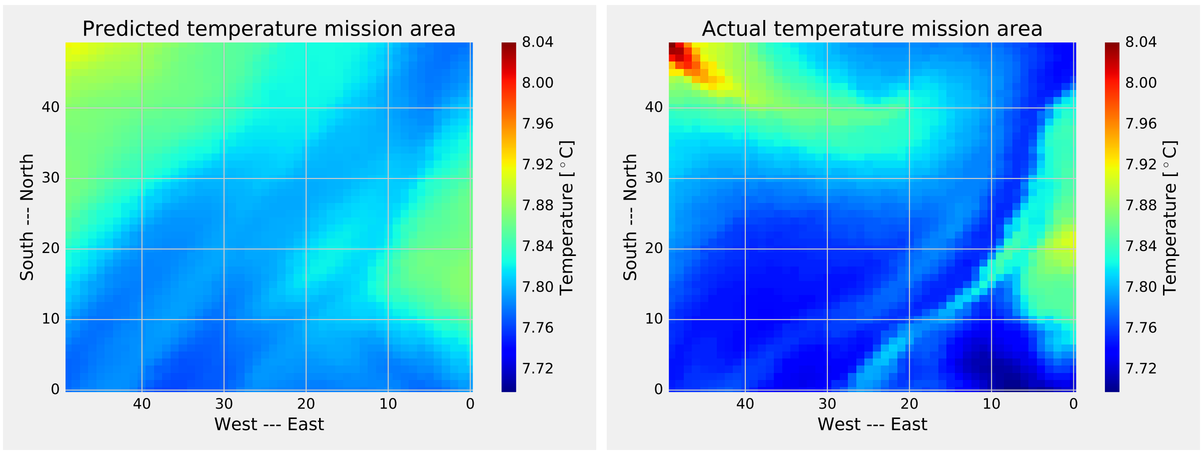


Figure 4.13: Measured temperature from both AUVs, sim 4

4.5 Sim 5

Both AUVs stops after 12 hours because of the reference source time out. AUV 1 tracks 273 temperature grid points while AUV 2 tracks 274 points. This results in the MAE between the true temperature grid and the predicted temperature grid to be $0.01422°C$.



(a) Predicted temperature used in GP, sim 5 (b) Actual temperature for mission area, sim 5

Figure 4.14: Predicted temp and actual temp for mission area, sim 5

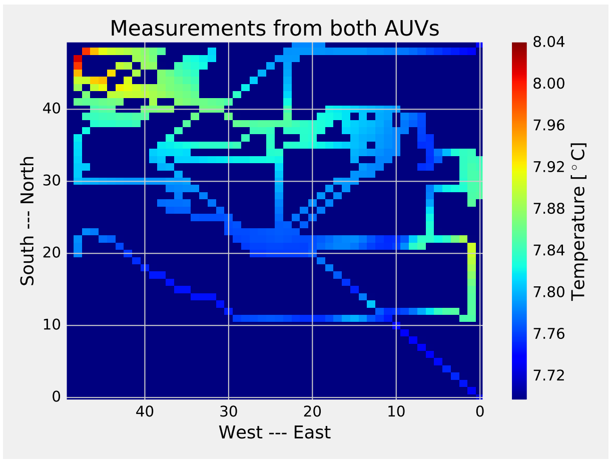
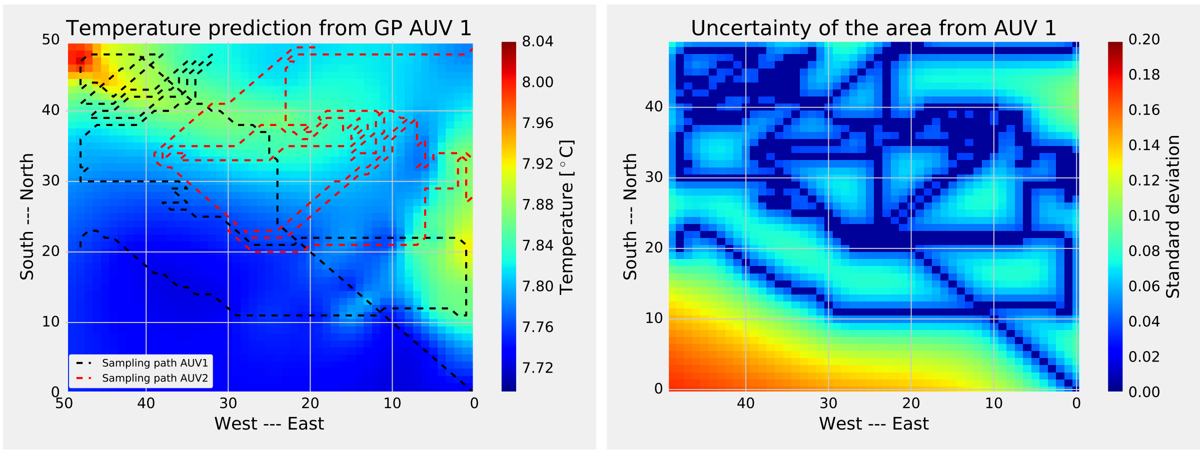


Figure 4.15: Measured temperature from both AUVs, sim 5

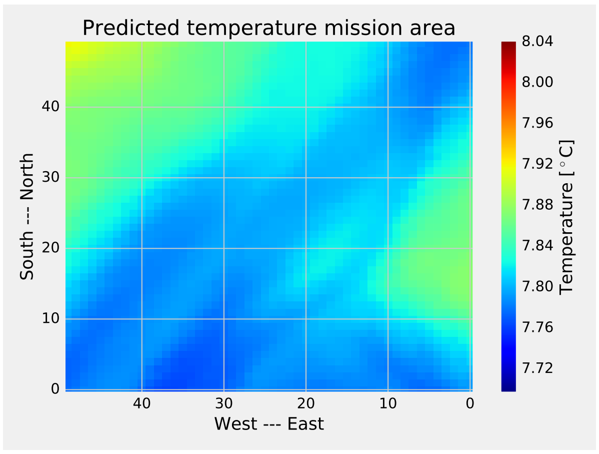


(a) Temperature prediction from GP, sim 5 (b) Uncertainty from GP, sim 5

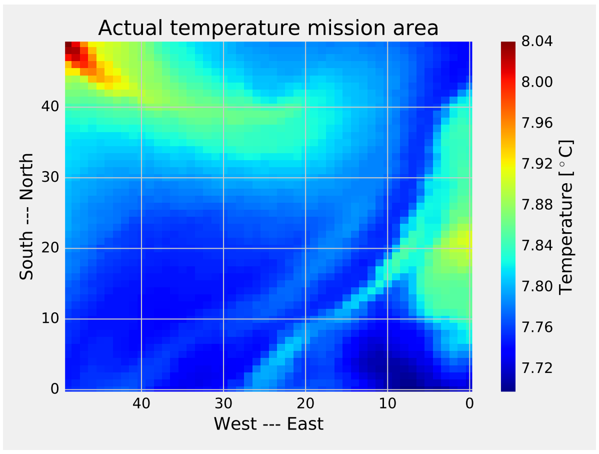
Figure 4.16: The results from the GP, sim 5

4.6 Sim 6

The simulation runs for approximately 13 hours before the code timed out. AUV 1 tracked 290 grid points, and AUV 2 tracked 288 points. The MAE between the true temperature of the area and the predicted temperature was 0.01410°C.



(a) Predicted temperature used in GP, sim 6



(b) Actual temperature for mission area, sim 6

Figure 4.17: Predicted temp and actual temp for mission area, sim 6

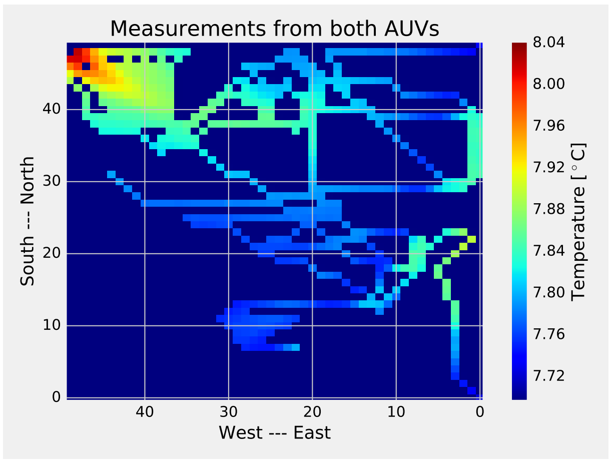
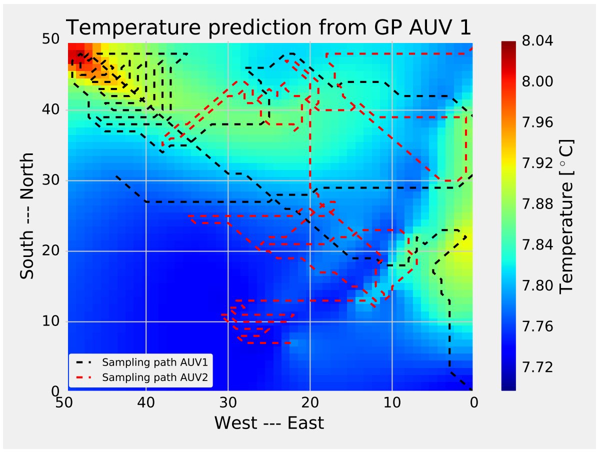
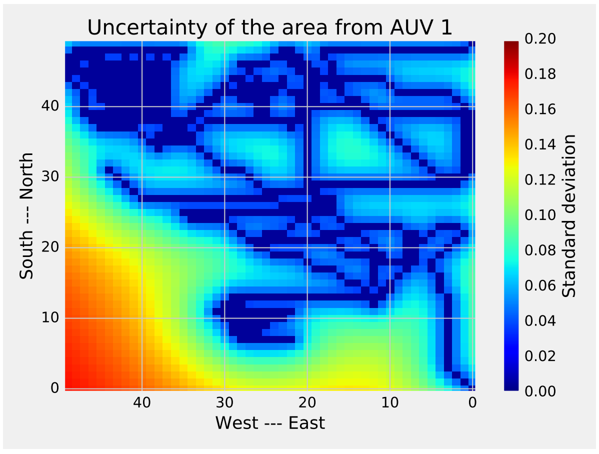


Figure 4.18: Measured temperature from both AUVs, sim 6



(a) Temperature prediction from GP, sim 6

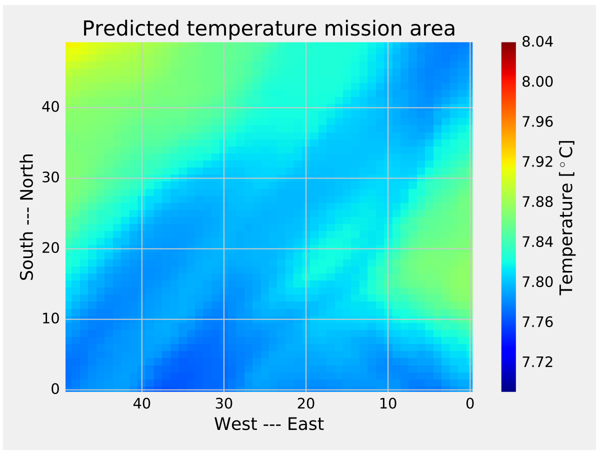


(b) Uncertainty from GP, sim 6

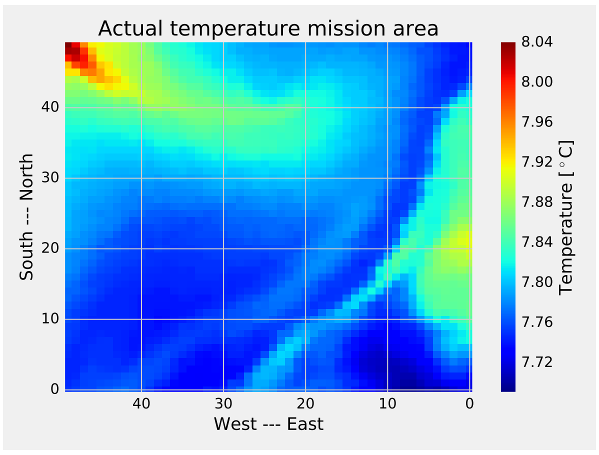
Figure 4.19: The results from the GP, sim 6

4.7 Sim 7

After 9 hours, both AUVs found $[0, 17]$ as the grid point with the highest absolute gradient. None of the AUVs was able to find a path to the grid point, and both vehicles aborted the mission. AUV 1 tracked a total of 225 grid points, and AUV 2 tracked 226. The MAE between the true temperature and the predicted temperature was $0.01728°C$.

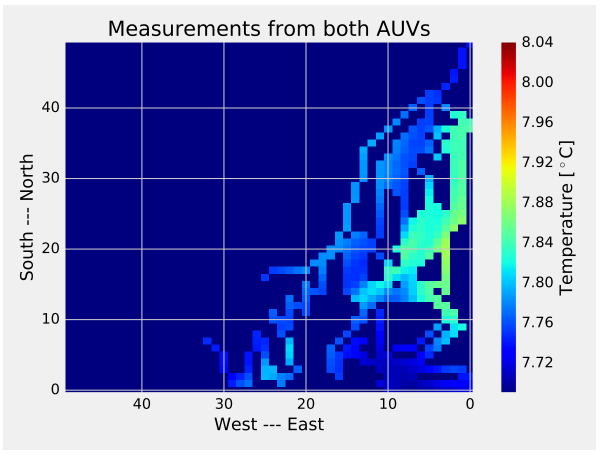


(a) Predicted temperature used in GP, sim 7

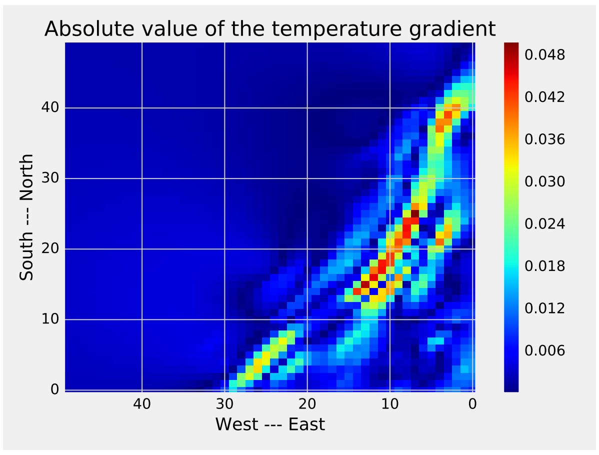


(b) Actual temperature for mission area, sim 7

Figure 4.20: Predicted temp and actual temp for mission area, sim 7

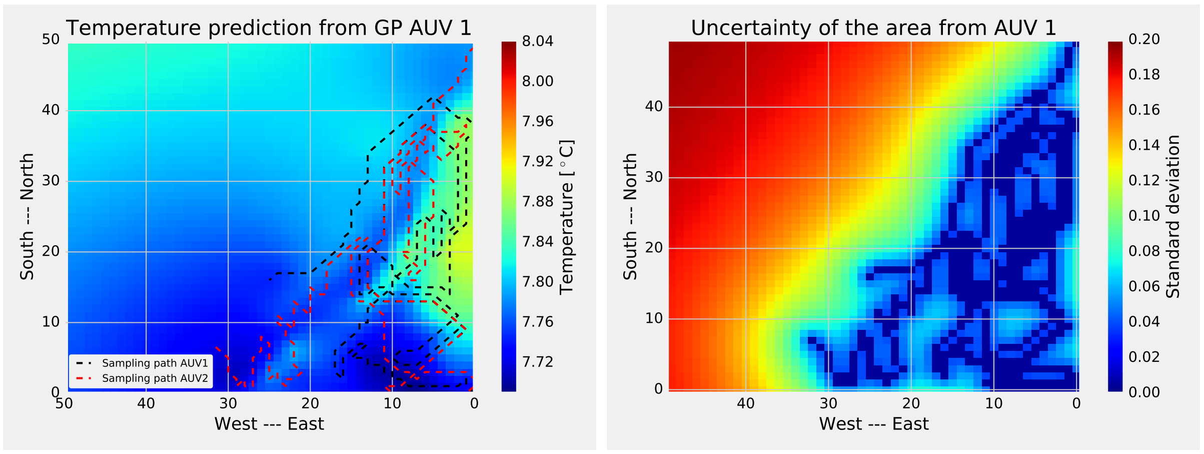


(a) Temperature from both AUVs, sim 7



(b) Absolute temperature gradient, sim 7

Figure 4.21: Measured temperature and temperature gradient, sim 7

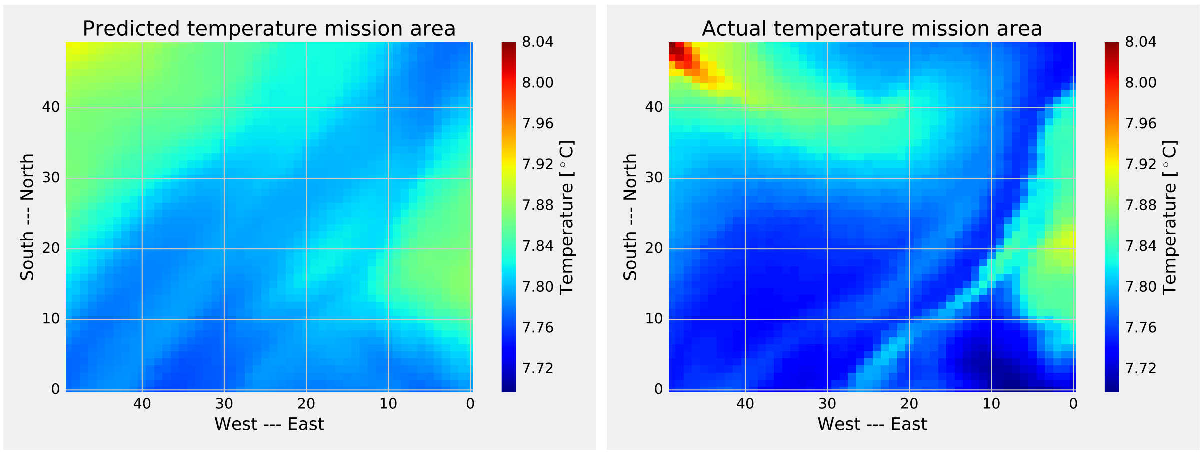


(a) Temperature prediction from GP, sim 7 (b) Uncertainty from GP, sim 7

Figure 4.22: The results from the GP, sim 7

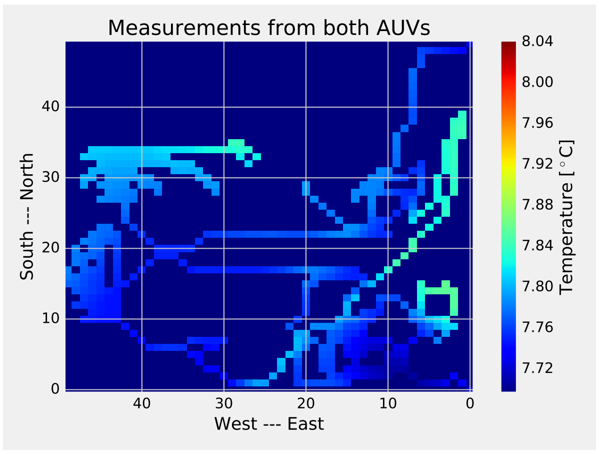
4.8 Sim 8

After 11 hours both AUVs again found grid point [0, 17] as the grid point with the highest absolute gradient. None of the AUVs was able to find a path to the position and the AUVs aborted the missions. Total points covered by AUV 1 was 257, and for AUV 2 it was 256. The MAE between the true temperature and the predicted temperature was $0.01956°C$.

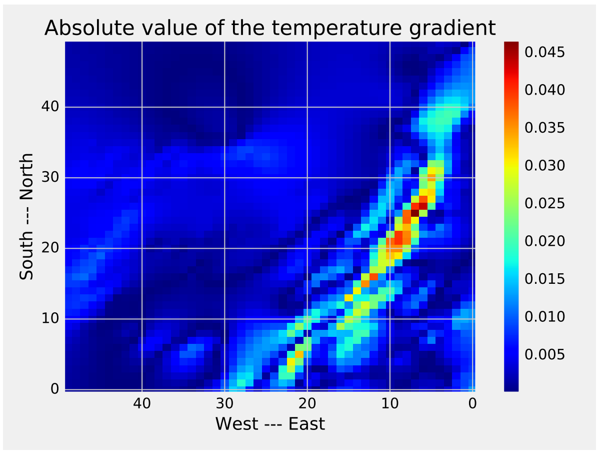


(a) Predicted temperature used in GP, sim 8 (b) Actual temperature for mission area, sim 8

Figure 4.23: Predicted temp and actual temp for mission area, sim 8

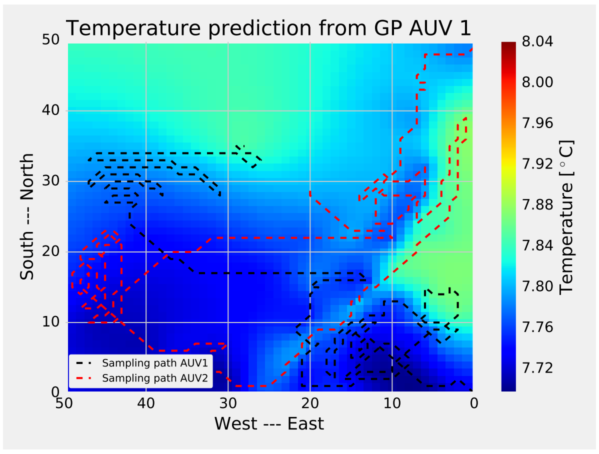


(a) Temperature from both AUVs, sim 8

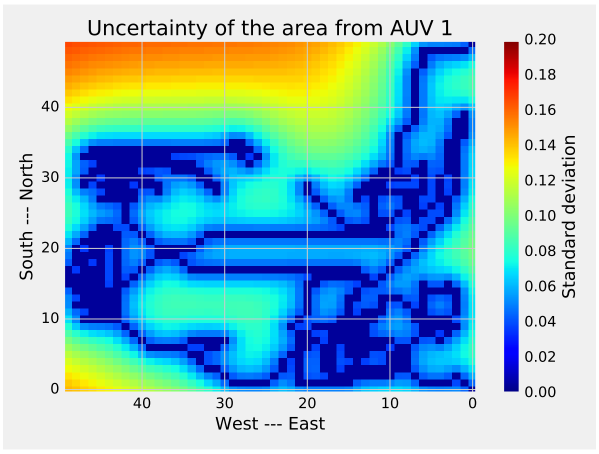


(b) Absolute temperature gradient, sim 8

Figure 4.24: Measured temperature and temperature gradient, sim 8

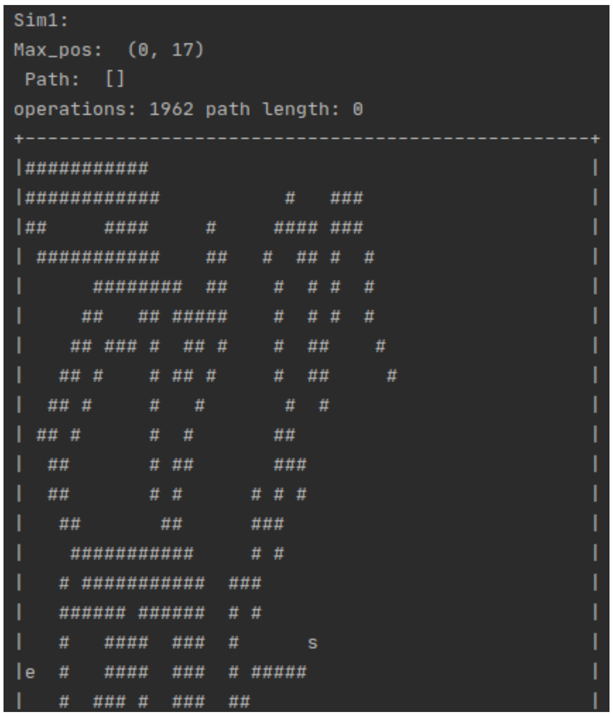


(a) Temperature prediction from GP, sim 8



(b) Uncertainty from GP, sim 8

Figure 4.25: The results from the GP, sim 8

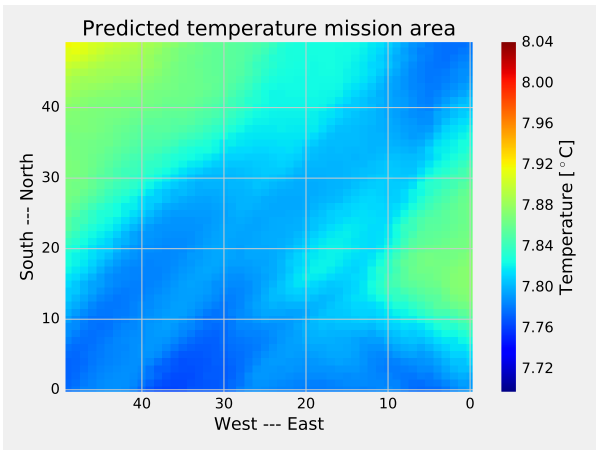


```
Sim1:
Max_pos:  (0, 17)
 Path:  []
operations: 1962 path length: 0
+--------------------------------------------------+
|###########                                       |
|############           #   ###                    |
|##     ####     #     #### ###                    |
| ###########    ##   #  ## #  #                   |
|      ########  ##    #  # #  #                   |
|     ##   ## #####    #  # #  #                   |
|    ## ### #  ## #    #  ##    #                  |
|   ## #    # ## #     #  ##     #                 |
|  ## #     #   #       #  #                       |
| ## #      #  #       ##                          |
|  ##       # ##       ###                         |
|  ##       # #      # # #                         |
|   ##       ##      ###                           |
|    ###########     # #                           |
|   # ###########  ###                             |
|   ###### ######  # #                             |
|   #   ####  ###  #      s                        |
|e  #   ####  ###  # #####                         |
|   #  ### #  ###  ##                              |
```

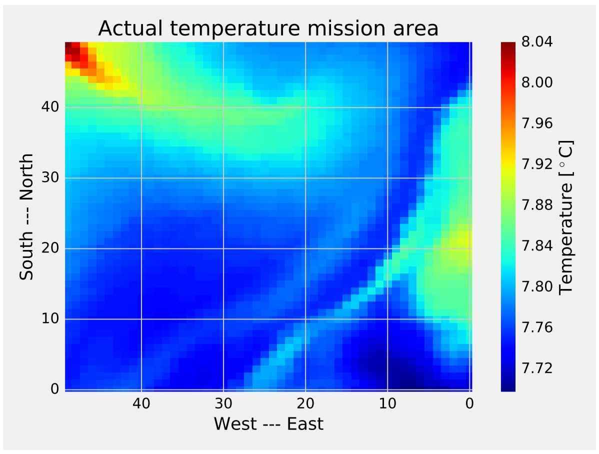
Figure 4.26: Showing the printout of the A* method trying to find a path to [0, 17] in simulation 8. Where s is the start position and e is the end position.

4.9 Sim 9

AUV 1 was looking for the lowest gradient, but after 10 hours AUV 1 aborted the mission when it was unable to find the path to next waypoint. AUV 2 looking for highest gradient run for 14 hours before the reference source in PyCharm timed out. This resulted in AUV 1 tracking 226 grid points giving a MAE between true temperature and predicted temperature at $0.01956°C$. AUV 2 tracked 354 grid points resulting in a MAE of $0.01886°C$.

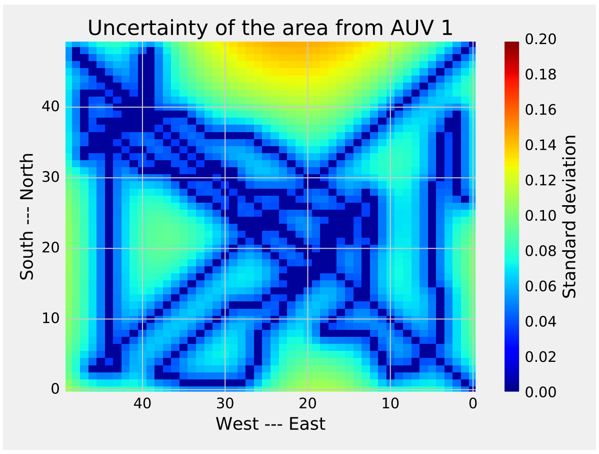


(a) Predicted temperature used in GP, sim 9

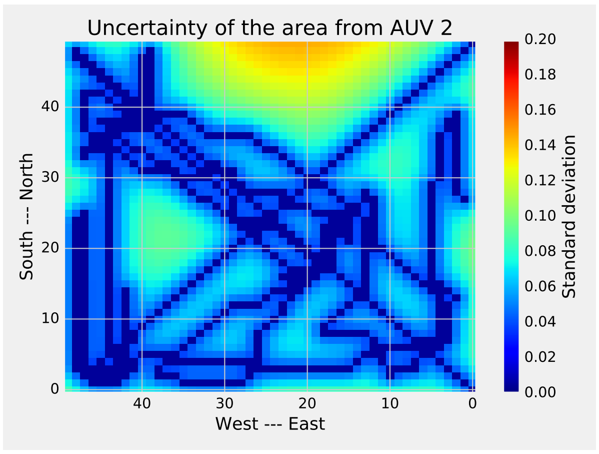


(b) Actual temperature for mission area, sim 9

Figure 4.27: Predicted temp and actual temp for mission area, sim 9

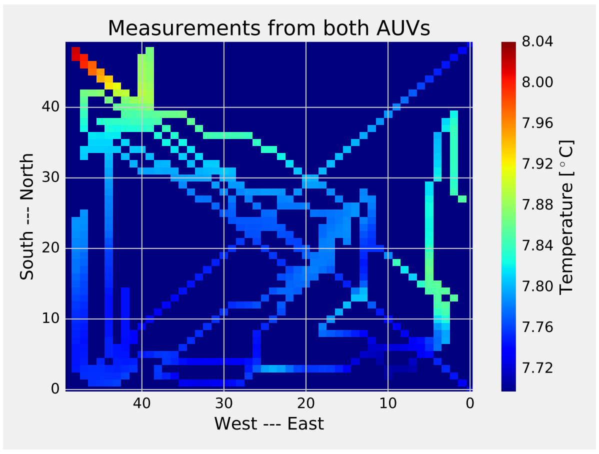


(a) Uncertainty from GP AUV 1, sim 9

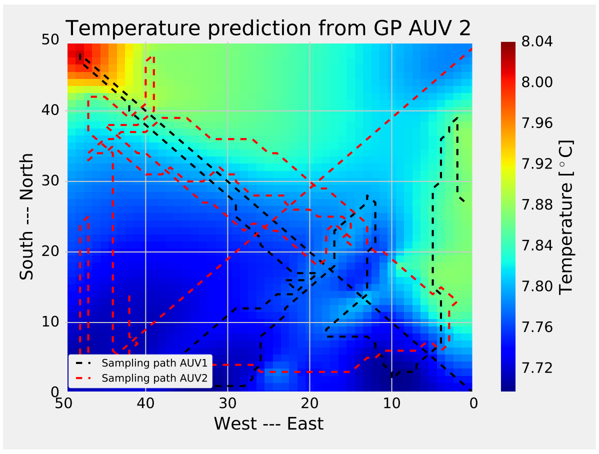


(b) Uncertainty from GP AUV 2, sim 9

Figure 4.28: Uncertainty both AUVs, sim 9



(a) Measured temperature both AUVs, sim 9



(b) Temperature prediction from GP, sim 9

Figure 4.29: The results from the GP, sim 9

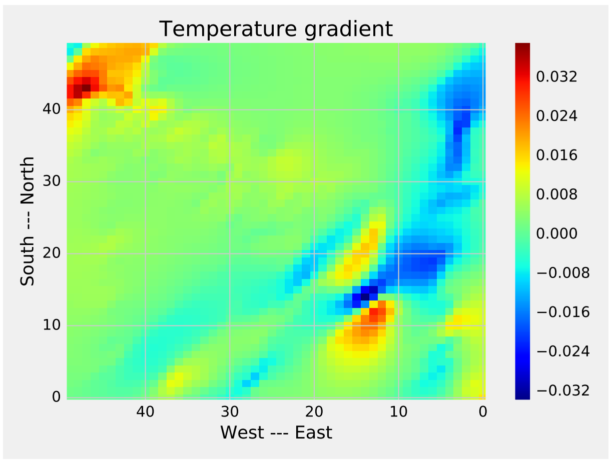
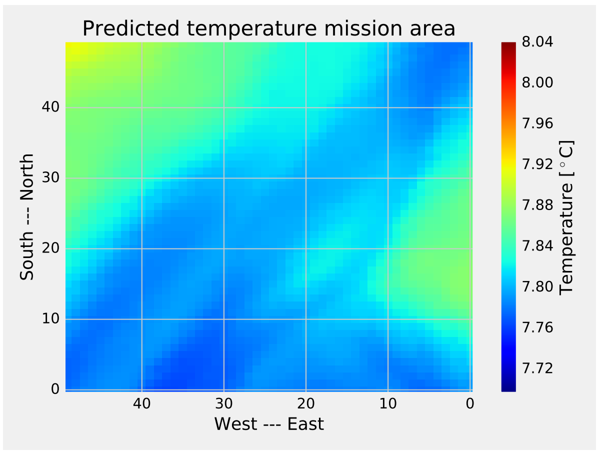


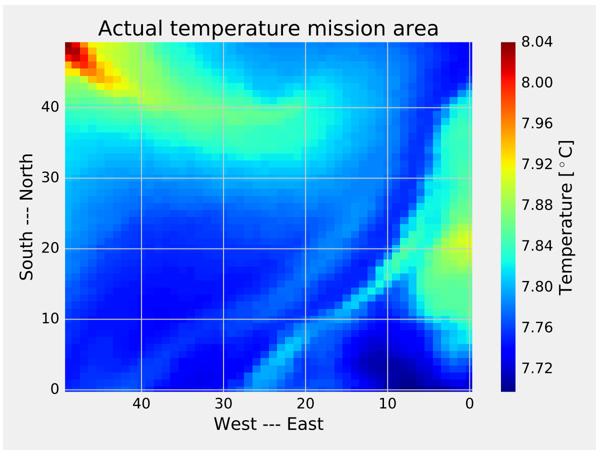
Figure 4.30: Temperature gradient, sim 9

4.10 Sim 10

The simulation ran for almost 13 hours before the source timed out. This resulted in AUV 1 tracking 280 grid points and AUV 2 tracking 285. The MAE between true temperature and the predicted temperature grid was 0.01911°C for AUV 2.

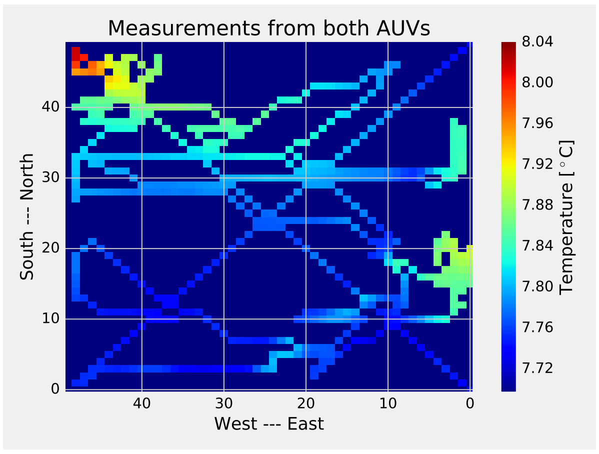


(a) Predicted temperature used in GP, sim 10

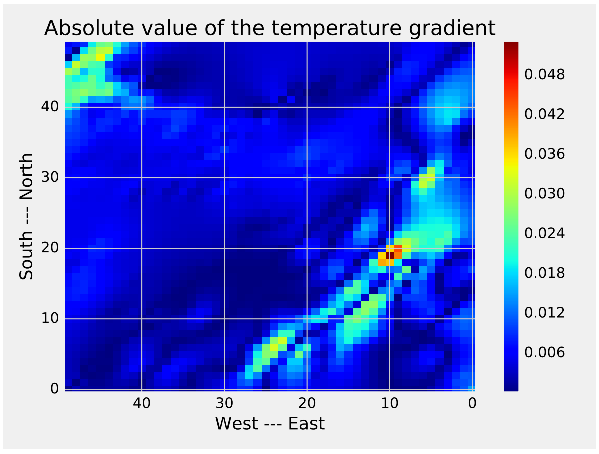


(b) Actual temperature for mission area, sim10

Figure 4.31: Predicted temp and actual temp for mission area, sim 10

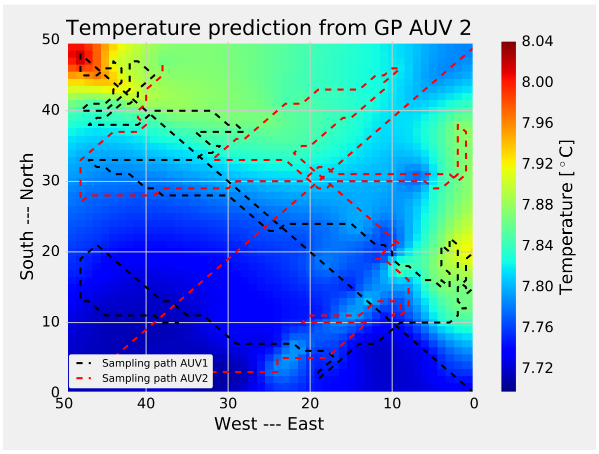


(a) Temperature from both AUVs, sim 10

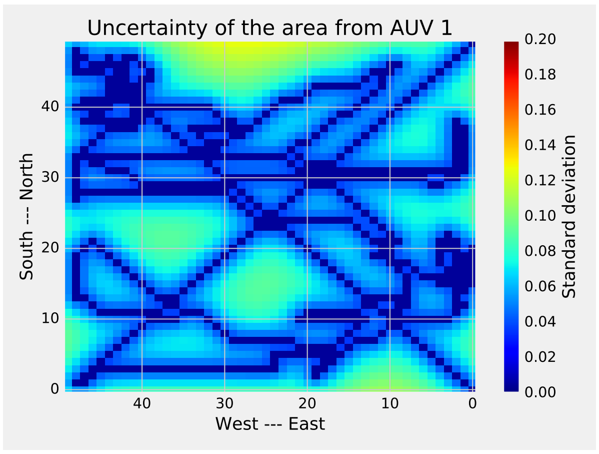


(b) Absolute temperature gradient, sim 10

Figure 4.32: Measured temperature and temperature gradient, sim 10



(a) Temperature prediction from GP, sim 10

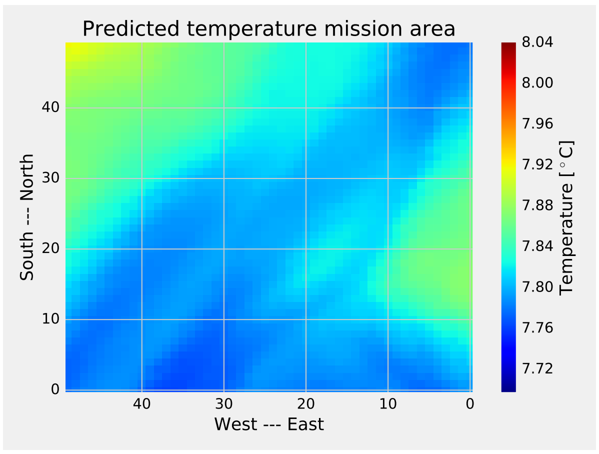


(b) Uncertainty from GP, sim 10

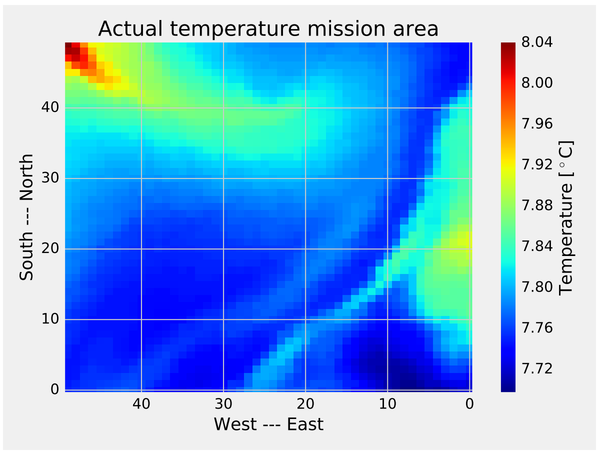
Figure 4.33: The results from the GP, sim 10

4.11 Sim 11

The simulation ran for 12 hours before the reference source timed out. AUV 1 was able to track 276 grid points while AUV 2 tracked 268 points. The MAE between true grid temperature and the predicted temperature was 0.01838°C from AUV 1.



(a) Predicted temperature used in GP, sim 11



(b) Actual temperature for mission area, sim11

Figure 4.34: Predicted temp and actual temp for mission area, sim 11

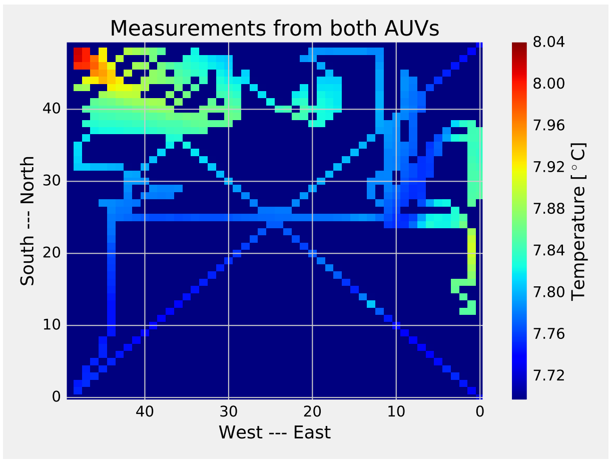
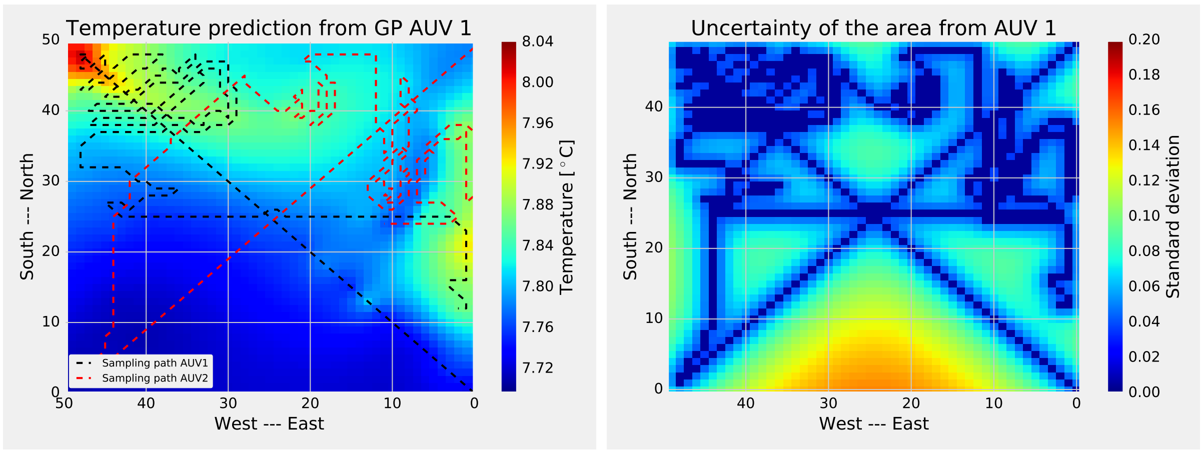


Figure 4.35: Measured temperature from both AUVs, sim 11

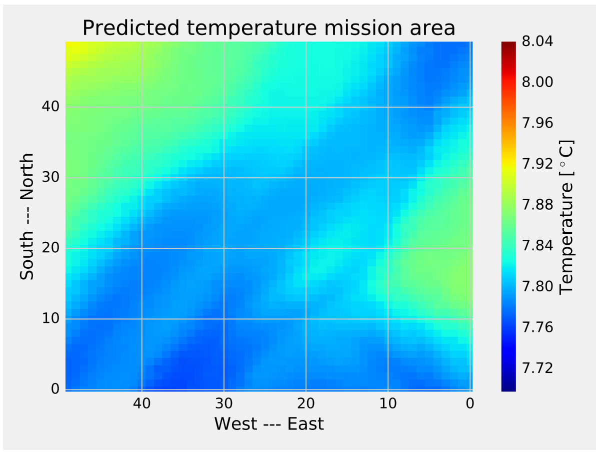


(a) Temperature prediction from GP, sim 11 (b) Uncertainty from GP, sim 11

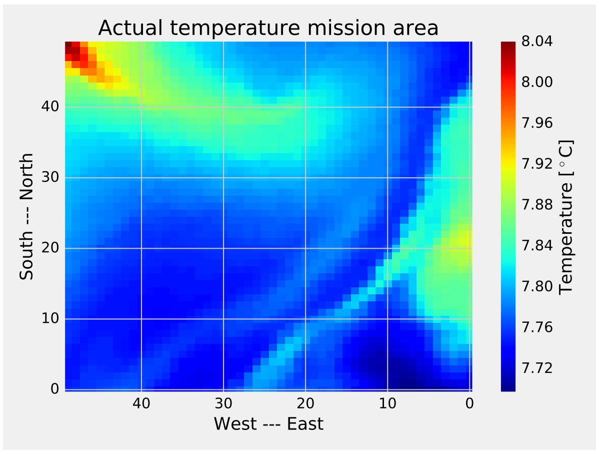
Figure 4.36: The results from the GP, sim 11

4.12 Sim 12

The simulation ran for a total of 25 hours. AUV 1 ran for only 13 hours before it aborted the mission when it was not able to find the path to max temperature position. AUV 1 tracked 377 grid points, while AUV 2 ran for 25 hours tracking 550 points, AUV 2 was stopped manually. Total points checked by the two AUVs are 771, resulting in 156 points that both AUVs have checked. The MAE between true grid temperature and the predicted temperature was $0.02062°C$ for AUV 1 and $0.01920°C$ for AUV 2.

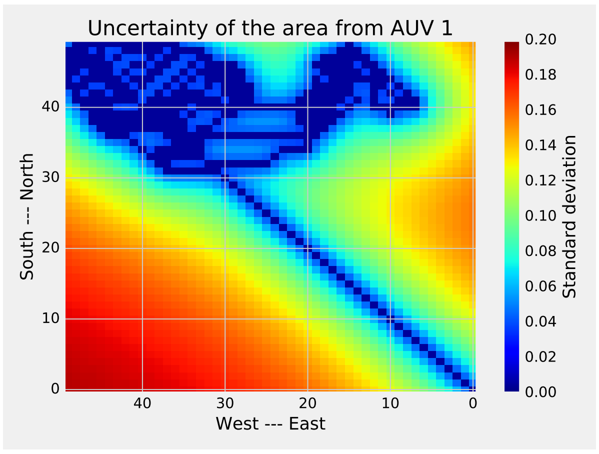


(a) Predicted temperature used in GP, sim 12

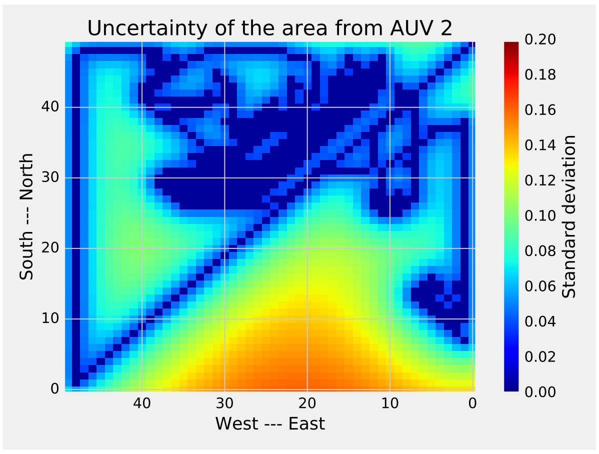


(b) Actual temperature for mission area, sim12

Figure 4.37: Predicted temp and actual temp for mission area, sim 12

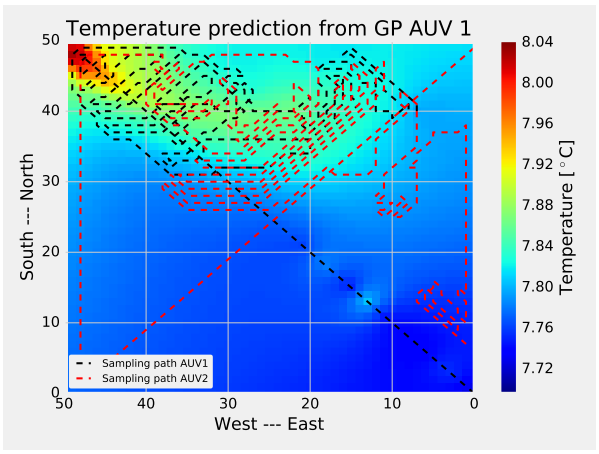


(a) Uncertainty from GP AUV 1, sim 12

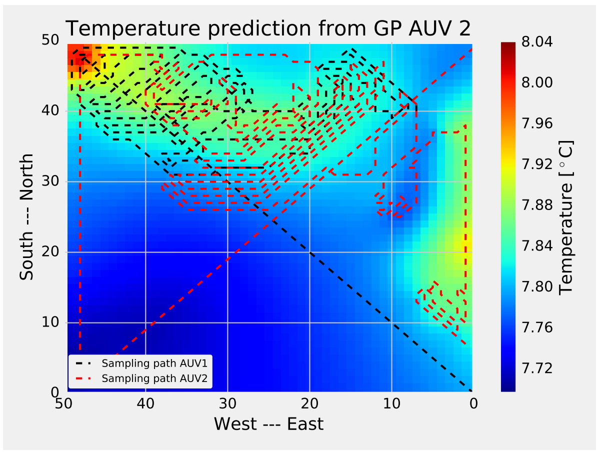


(b) Uncertainty from GP AUV 2, sim 12

Figure 4.38: Uncertainty both AUVs, sim 12



(a) Temperature prediction from GP, sim 12



(b) Temperature prediction from GP, sim 12

Figure 4.39: The results from the GP, sim 12

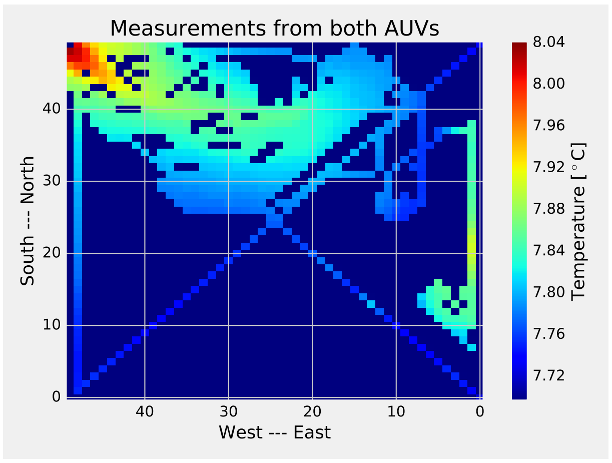
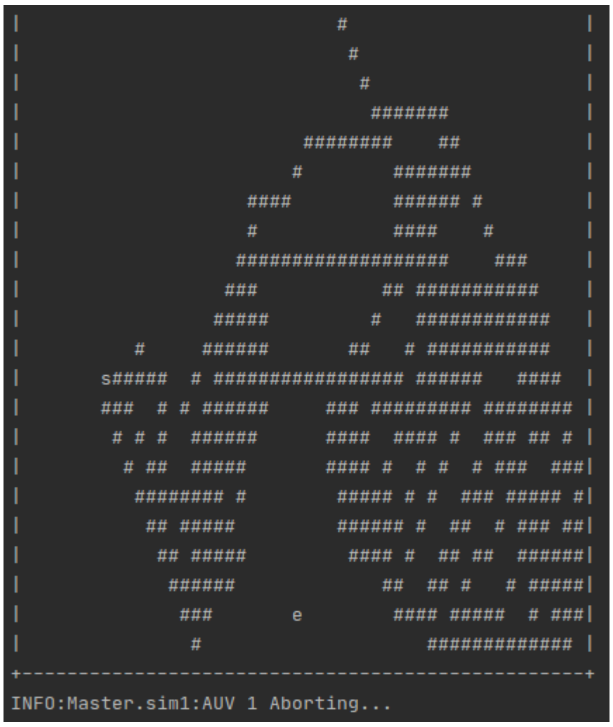


Figure 4.40: Measured temperature from both AUVs, sim 12



```
|                            #                     |
|                             #                    |
|                              #                   |
|                               #######            |
|                         ########    ##           |
|                        #        #######          |
|                    ####         ###### #         |
|                    #            ####    #        |
|                   ###################    ###     |
|                  ###           ## ###########    |
|                 #####         #   ############   |
|          #     ######       ##   # ###########   |
|       s#####  # ################# ######   ####  |
|       ###  # # ######     ### ######### ######## |
|        # # #  ######      ####  #### #  ### ## # |
|         # ##  #####       #### #  # #  # ###  ###|
|          ######## #        ##### # #  ### ##### #|
|           ## #####         ###### #  ##  # ### ##|
|            ## #####         #### #  ## ##  ######|
|             ######             ##  ## #   # #####|
|              ###       e        #### #####  # ###|
|               #                    ############# |
+--------------------------------------------------+
INFO:Master.sim1:AUV 1 Aborting...
```

Figure 4.41: Showing the printout of the A* method trying to find a path for AUV 1 in simulation 12. Where s is the start position and e is the end position.

Chapter 5

Discussion

5.1 Sim 1

In the first simulation, the results show that both AUVs are adapting to the predicted temperature area. Both AUVs started with the northeast grid point $[49, 49]$ as the highest temperature point. This seems reasonable, based on Figure 3.8b. AUV 2 reaches the corner when AUV 1 is around grid point $[32, 32]$ and Figure 4.3a shows that AUV 1 then changes course. AUV 1 ends up going back and forth over the bottom area of the temperature hotspot, while AUV 2 stays more inside the hotspot. The main difference here is the different uncertainty cap for the two AUVs. From the difference in the AUVs path, it is possible to see AUV 2 mapping closer points before changing direction. AUV 1, on the other hand, finds maximum value points on the east side before finding the next one on the west side, going back and forth. The obstacle method in the A* pathfinding seems to work, preventing the AUVs tracking grid points already visited. By looking at the uncertainty plot in Figure 4.3b, the coverage of the area is not satisfying, and the uncertainty is still high in the southwest area. With this method, the AUVs also misses the hotspot in the middle east area. The simulation has a MAE of $0.01932°C$. This is probably because the simulation does not explore the hotspot in the eastern part of the grid.

5.2 Sim 2 and Sim 3

Simulation 2 and simulation 3 is similar, with one AUV looking for the highest temperature while the other looks for the lowest. In simulation 2 AUV 1 is looking for high temperatures while AUV 2 is looking for low temperatures, in simulation 3, it is opposite. When comparing the results from the two simulations, it is possible to see that the coverage of the area is similar. The MAE is slightly different though, simulation 2 has an MAE of $0.01825°C$, while the MAE is only $0.01707°C$ for simulation 3. The difference can not be seen in the plots and can be because simulation 3 has slightly more tracked points than simulation 2. In Simulation 2, the adaptivity of AUV 1 is seen when it changes its course when AUV 2 finds the hotspot in the eastern part. AUV 1 does less covering in the hotspot area before continuing up to the top west corner, compared to simulation 3 where AUV 2 covers the area more thoroughly. This results in a better representation of the hotspot area in the prediction. It also results in better ocean data in the hotspot area, which can be of interest for biologists and oceanographers.

5.3 Sim 4

In this simulation, the grid area is split into an upper and a lower part. This resulted in both AUVs eventually aborting because they were not able to find any more grid point to track due to the uncertainty cap. The results from the uncertainty plots in Figure 4.11 shows that the general uncertainty of the total grid is low, and for AUV 1, the uncertainty is below 0.09 for all grid points. AUV 2 also has an uncertainty below 0.09 for the upper area. The simulation time is relatively short, only 8 hours for AUV 2 and almost 12 hours for AUV 1. Still, the results from the prediction plot in Figure 4.12 shows a good representation of the temperature compared to the true temperature. The MAE between the predicted temperature and true temperature is $0.01658°C$ for AUV 2 after approximately 380 (both AUVs) gird points checked. For AUV 1 the MAE is larger at $0.01800°C$ after approximately 450 grid points checked. The difference in error is not visible in the prediction plots in Figure 4.12. This shows that more points tracked do not mean a better prediction necessarily for this GP. The results could have been different if the hyperparameters in the GP was optimized. The results also show the efficiency of covering a large area for two AUVs when dividing the area into smaller parts.

5.4 Sim 5 and Sim 6

Simulation 5 and 6 are the first simulations to consider the other AUVs destination. The difference between the two simulations is which one of the vehicles that consider the other vehicles destination first. AUV 1 starts considering AUV 2s destination at the third iteration in simulation five and second iteration in simulation 6, and opposite for AUV 2 considering AUV 1s destination. In both simulations, AUV 1 tracks the hot spot in the northwest corner thoroughly, slightly more in simulation 6. This will give a lot of ocean data in an interesting area. AUV 2 can be seen checking other areas while AUV 1 is in the northwest square. Ideally, AUV 2 should have explored the other hotspot in the eastern area more accurately. Instead, AUV 2 maps the eastern tip of the large hotspot in simulation 5 and a colder spot in the middle south in simulation 6. The tip of the hotspot can be interesting for oceanographers and biologists and is not a bad area to explore. The MAE in both simulations is very similar, being $0.01422°C$ for simulation 5 and $0.01410°C$ in simulation 6. This seems reasonable when looking at the prediction plots, which looks very similar. Both plots seem to be a reasonable estimation of the actual temperature.

5.5 Sim 7

The first simulation with gradient as the deciding factor for the AUVs did not result in a good result. Since the first prediction is based on the μ in Figure 3.8b, the gradient is almost the same in every grid point. When the AUVs register their temperature values, these grid points get the highest gradient. The result is that AUV 1 starts checking around itself while AUV 2 starts to go south. When AUV 2 starts tracking the hotspot in the east, the gradient values in this area rises, resulting in both AUVs exploring this area. Since the uncertainty is lowered to 0.08 in this simulation for both AUVs, and they do not take into the other AUVs destination, both AUVs explore the hotspot area thoroughly. None of the AUVs goes to the other hotspot in the northwest corner. The result is a very thoroughly explored east side but a large area with high uncertainty in the west. The resulting MAE is $0.01728°C$, comparing true temperature with predicted temperature. This error is surprisingly low, considering only half the grid is explored. This can be due to the predicted temperature in the upper left corner have a higher value even though it is not mapped.

5.6 Sim 8

Even though the uncertainty cap is raised and the AUVs are taking the other AUVs destination into account, they both find $[0, 17]$ as the maximum gradient point after 11 hours. From the print out in PyCharm, it seems that AUV 2 is not able to find the path to $[0, 17]$ and then aborts. When AUV 2 aborts the destination is not updated, leaving the textfile with the destination of AUV 2 to be $[22, 34]$ instead of $[0, 17]$. Because of this, AUV 1 finds $[0, 17]$ as maximum gradient position, resulting in AUV 1 also aborting. Figure 4.26 shows how AUV 1 is unable to reach "e" from "s" because of the double obstacle ("#") at the left side. The prediction plot in Figure 4.25a shows that none of the AUVs is able to explore neither of the two hotspots. This is reflected in the MAE being as high as $0.01956°C$.

5.7 Sim 9

In this simulation, it is four hours that differs the two AUVs, and AUV 2 has explored 128 more grid points than AUV 1. The mission starts with the AUVs travelling to the opposite grid point from their start point before starting to explore for high and low gradient grid points. In Figure 4.29b, the predicted field shows a higher temperature where the hotspots are, but none of the hotspots is explored thoroughly. It seems like AUV 1 has followed the border of the hot spot in east and AUV 2 has followed the border of the northwest hot spot. This is correct based on them looking for temperature gradient and not the maximum temperature. Still, the prediction is far from the actual temperature with a MAE of $0.01956°C$ for AUV 1 and $0.01886°C$ for AUV 2. AUV 2 having a slightly lower error than AUV 1 after a four hours longer mission is expected, though the error could have been lower with more concentrated exploring in the most critical regions.

5.8 Sim 10, sim 11 and sim 12

Simulation 10, 11 and 12 is run to be compared. All three simulations start with the AUVs crossing the grid to the opposite grid point before the autonomy starts. In simulation 10 the AUVs are looking for the maximum absolute value of the temperature gradient. Simulation 11 is set up, so the two AUVs are both looking for maximum temperature in the grid. The same constraints hold for simulation 12, but the AUVs can not communicate. Comparing the mission time, the three simulations are somewhat equal. Simulation 10 ran for 13 hours, simulation 11 ran for 12 hours, and AUV 1 in simulation 12 ran for 13 hours. AUV 2 in simulation 12 run for 25 hours. This was not the intention but resulted in the opportunity to compare AUV 2 almost as a separate result. Looking at the error for the four missions show that simulation 11 has the best results, as shown in Table 5.1. This also seems reasonable when looking at the prediction plots for the four cases. Both AUVs in simulation 12 has a higher error than the other two simulations. The error of AUV 2 is not far from the error in simulation 10, but the mission ran for 25 hours compared to 13 hours. This clearly shows the benefit of the communication between the AUVs.

Table 5.1: Error results from simulation 10, 11 and 12.

AUV	Grid points	MAE [°C]
AUV 1 (Sim 10)	280+280	0.01951
AUV 2 (Sim 10)	285+280	0.01911
AUV 1 (Sim 11)	276+268	0.01838
AUV 2 (Sim 11)	268+268	0.01851
AUV 1 (Sim 12)	377	0.02062
AUV 2 (Sim 12)	550	0.01920

5.9 Summary

Looking at all the simulations compared to each other, simulation 6 has the lowest error. The first six simulations tracking temperature has an overall lower error and maps the hot spots better. Depending on what the goal of the mission is and what data the scientists are looking for, the simulations searching for temperature gradient shows the possibilities to track along hotspots in a good way, following the gradient line. For the majority of the simulations, the total time is around 12 hours, with several of the missions ending because of the error "reference source timed out". The reason for this is not fully understood, but since AUV 2 in simulation 12 was able to run for 25 hours, the error can be due to the GP or the A* pathfinding using to much time when the amount of data points reaches a certain level. The best result from each simulation can be seen in Table 5.2.

Table 5.2: Best result from all simulations.

Simulation	Total grid points	Total time [hours]	Looking for	MAE [°C]
1	489	10	Temperature	0.01932
2	550	12	Temperature	0.01825
3	575	12	Temperature	0.01707
4	454	12	Temperature	0.01658
5	547	12	Temperature	0.01422
6	578	13	Temperature	0.01410
7	451	9	Gradient	0.01728
8	513	11	Gradient	0.01956
9	580	14	Gradient	0.01886
10	565	13	Gradient	0.01911
11	544	12	Temperature	0.01838
12	550	25	Temperature	0.01920

Chapter 6

Conclusion and Further work

6.1 Conclusion

This master thesis presents an algorithm for doing adaptive sampling for two AUVs using GP to model an area. The results show the advantage of using two AUVs with communication compared to using two AUVs without communication or only one AUV. By testing several myopic decision methods, the MAE of the 50×50 grid area was reduced from $0.03046°C$ to $0.01410°C$ in the best simulation. Still, the results from the evaluation method show some inconsistency, where the prediction field has a higher error with more covered grid points than for the same mission with less covered grid points.

The chosen ocean data to explore is temperature. This was chosen based on the available data in the SINMOD file and because it is seen as more informative for biologists and oceanographers than salinity. The simulations only show how temperature data is gathered. The results do not take into account how the other ocean data in the explored positions compare to the temperature, and the real value of the gathered ocean data.

The simulations show promising results for the algorithm, and it can be suitable as a base for doing the adaptive sampling. Real-life performance has to be tested before this can be said for certain.

As mentioned in the literature review in subsection 2.4.4, there are ways of improving and simulating communication when doing subsea missions. This was not a problem here since everything was simulated and the AUVs was at the surface. If the system is to be used in a real-life test, this needs to be done to be able to achieve similar results for multiple vehicles.

Simulation 10, 11 and 12 shows that adaptive sampling with communication is better than without, even when one AUV runs twice the time. Even though the results were as expected, there was done a mistake when calculating the total error. These simulations were supposed to be a comparison of the best simulation against a simulation without communication. The error was not discovered before after running the last simulation. Simulation 10 was, therefore chosen to be the best simulation. Since the AUVs in this mission was looking for the temperature gradient, the simulation was also done for temperature. This showed even better results and simulation 12 without communication was, therefore, ran with temperature and not the temperature gradient. After calculating the correct error, simulation 5 or 6 should have been tested with gradient and without communication since the mistake did not alter the total result to much, the conclusion stand.

The design parameters (hyperparameters) σ^2 (variance) and ℓ (correlation distance) in the kernel of the GP model was set equal to the default values from Trygve Olav Fossums program. These parameters could have been optimised for this thesis. The optimisation could have been done by several different methods, as mentioned in subsection 2.3.3. This could have improved the prediction in the GP, resulting in better adaptive sampling and more informative ocean data.

In this thesis, the temperature is used to simulate the ocean data. The algorithm presented makes it possible to survey an area both time-efficient and with a finer resolution. By optimising the hyperparameters in the GP and making a robust communication model for two or more AUVs, this algorithm should be able to work with any ocean data, not only temperature. This can give a unique opportunity to biologists and oceanographers. The use of adaptive sampling can broaden the possibilities to discover data sets never gathered before. By using GP to predict the ocean data, it is possible to do adaptive sampling even with little or no previous data for a given area.

6.2 Further work

For further work, a field test should be carried out. This was not possible for this thesis due to the Corona virus. A field test would show how the proposed algorithm works in a dynamic environment. Without too much effort, the program should also be able to use the SINMOD temperatures, as shown in Appendix B to simulate a more dynamic ocean by changing the time the data are gathered. Before eventually launching a field test, several issues need to be looked further into.

Studying some replays of the simulations in Neptus, there are some situations where the AUVs are close to colliding. A situation where the AUVs collide was not seen, but because of navigation errors subsea, the AUVs should prevent being to close. This is not an urgent problem while doing simulations, since there is never a risk of loss of equipment, and is therefore not prioritised in this thesis. Still, this is something that needs to be avoided if a field test is to be carried out.

The cause for the error ”reference source timed out”, resulting in the missions ending, is not known as this thesis concludes. If the program is to be used in further work, this should be investigated. Possible reasons for the time out is due to the GP or the A* pathfinding using to much time or computational power. This needs to be looked into before doing a field test.

There is still some more decision methods that can be tested. A test where one AUV is looking for maximum temperature while the other AUV search for the absolute value of the temperature gradient could prove to be efficient. It could also be interesting to do more changes to the uncertainty cap after the design parameters in the GP model is optimised.

As technology will continue to develop the coming years, making more computational power lighter and more available, the possibilities with AUVs will increase. More computational power allows for implementing more advanced decision methods. In this thesis, several myopic decision methods were tested, but the opportunity to use a non-myopic method looking several steps ahead could have improved the results further. With a lot more computational power the possibility to use GP on several ocean data fields simultaneously, together with a non-myopic decision model could take ocean sampling as we know it a lot further, giving scientists even better data.

Bibliography

Bailey, Herbert S. (1953). "The Voyage of the "Challenger"". In: *Scientific American* 188.5. Publisher: Scientific American, a division of Nature America, Inc., pp. 88–95. ISSN: 0036-8733. URL: https://www.jstor.org/stable/24944225 (visited on 05/19/2020).

Belkin, I. et al. (Nov. 2018). "Marine robotics exploration of a large-scale open-ocean front". In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV), pp. 1–4. DOI: 10.1109/AUV.2018.8729725.

Bresser, Andreas (2020). *pathfinding: Pathfinding algorithms (based on Pathfinding.JS)*. Version 0.0.4. URL: https://github.com/brean/python-pathfinding (visited on 06/03/2020).

Buadu, Stephanie (2018). "Advanced Mission Planner for Cooperative Underwater Vehicles". In: URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2564473 (visited on 12/15/2019).

Costa, M. J. et al. (Nov. 2018). "Field Report: Exploring Fronts with Multiple Robots". In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV), pp. 1–7. DOI: 10.1109/AUV.2018.8729780.

Crow, Edwin L and Shimizu (1987). *Lognormal distributions*. Marcel Dekker New York.

Darwin, Charles (June 29, 1989). *The Voyage of the Beagle*. Google-Books-ID: ZdXEsblf9vQC. Penguin Adult. 452 pp. ISBN: 978-0-14-043268-8.

Eckstein, Sebastian, Thomas Glotzbach, and Christoph Ament (June 2013). "Towards innovative approaches of team-oriented mission planning and mission languages for multiple unmanned marine vehicles in event-driven mission". In: *2013 MTS/IEEE OCEANS - Bergen*. 2013 MTS/IEEE OCEANS - Bergen. ISSN: null, pp. 1–8. DOI: 10.1109/OCEANS-Bergen.2013.6608065.

Ferreira, António Sérgio et al. (Aug. 1, 2019). "Advancing multi-vehicle deployments in oceanographic field experiments". In: *Autonomous Robots* 43.6, pp. 1555–1574. ISSN: 1573-7527. DOI: 10.1007/s10514-018-9810-x. URL: https://doi.org/10.1007/s10514-018-9810-x (visited on 10/09/2019).

Fossum, Trygve Olav (2019). *Adaptive Sampling for Marine Robotics.* NTNU. ISBN: 978-82-326-3989-2. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2608426 (visited on 12/15/2019).

– (2020). *RESEARCH PAGE - TRYGVE O. FOSSUM.* Library Catalog: sites.google.com. URL: https://sites.google.com/view/research-trygve (visited on 06/02/2020).

Fossum, Trygve Olav, Jo Eidsvik, et al. (2018). "Information-driven robotic sampling in the coastal ocean". In: *Journal of Field Robotics* 35.7, pp. 1101–1121. ISSN: 1556-4967. DOI: 10.1002/rob.21805. URL: https://www.onlinelibrary.wiley.com/doi/abs/10.1002/rob.21805 (visited on 10/08/2019).

Fossum, Trygve Olav, Glaucia Moreira Fragoso, et al. (Feb. 20, 2019). "Toward adaptive robotic sampling of phytoplankton in the coastal ocean". In: *4:Eeaav3041.* ISSN: 2470-9476. DOI: 10.1126/scirobotics.aav3041. URL: https://sintef.brage.unit.no/sintef-xmlui/handle/11250/2609710 (visited on 10/08/2019).

Glotzbach, Thomas, Sebastian Eckstein, and Christoph Ament (Jan. 1, 2015). "An Approach for Planning a Safe Mission Begin and End for Teams of Marine Robots". In: *IFAC-PapersOnLine.* 4th IFAC Workshop onNavigation, Guidance and Controlof Underwater VehiclesNGCUV 2015 48.2, pp. 100–106. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2015.06.017. URL: http://www.sciencedirect.com/science/article/pii/S2405896315002566 (visited on 12/14/2019).

Guestrin, Carlos, Andreas Krause, and Ajit Singh (2005). "Near-optimal sensor placements in Gaussian processes". In: vol. 119. ICML '05. ACM, pp. 265–272. ISBN: 978-1-59593-180-1. DOI: 10.1145/1102351.1102385.

Insaurralde, Carlos C. and Yvan R. Petillot (Sept. 2013). "Intelligent autonomy for collaborative intervention missions of unmanned maritime vehicles". In: *2013 OCEANS - San Diego.* 2013 OCEANS - San Diego. ISSN: 0197-7385, pp. 1–6. DOI: 10.23919/OCEANS.2013.6741354.

Kemna, Stephanie (2018). *Multi-Robot Strategies for Adaptive Sampling with Autonomous Underwater Vehicles.* undefined. Library Catalog: www.semanticscholar.org. (Visited on 05/30/2020).

Krause, A. et al. (2006). "Near-optimal sensor placements: maximizing information while minimizing communication cost". In: vol. 2006. IEEE, pp. 2–10. ISBN: 978-1-59593-334-8. DOI: 10.1145/1127777.1127782.

Light Autonomous Vehicle LAUV - AUVAC (2019). URL: https://auvac.org/configurations/view/229 (visited on 12/12/2019).

Low, K. H., J. M. Dolan, and P. Khosla (2008). "Adaptive multi-robot wide-area exploration and mapping". In: *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS.* Vol. 1. ISSN: 1548-8403. International Foundation for Autonomous Agents and Multiagent Systems IFAAMAS, pp. 24–31. ISBN: 978-1-60560-470-1.

Low, Kian Hsiang (2009). "Multi-robot adaptive exploration and mapping for environmental sensing applications". AAI3374752 ISBN-13: 9781109357585. PhD thesis. USA: Carnegie Mellon University. 180 pp.

Madureira, Luís et al. (June 2013). "The light autonomous underwater vehicle: Evolutions and networking". In: *2013 MTS/IEEE OCEANS - Bergen.* 2013 MTS/IEEE OCEANS - Bergen. ISSN: null, pp. 1–6. DOI: 10.1109/OCEANS-Bergen.2013.6608189.

Martins, Ricardo et al. (May 2009). "IMC: A communication protocol for networked vehicles and sensors". In: *OCEANS 2009-EUROPE.* OCEANS 2009-EUROPE, pp. 1–6. DOI: 10.1109/OCEANSE.2009.5278245.

NOAA (July 11, 2018). *How much of the ocean have we explored?* Library Catalog: oceanservice.noaa.gov. URL: https://oceanservice.noaa.gov/facts/exploration.html (visited on 05/19/2020).

Paull, Liam et al. (Jan. 2014). "AUV Navigation and Localization: A Review". In: *IEEE Journal of Oceanic Engineering* 39.1, pp. 131–149. ISSN: 2373-7786. DOI: 10.1109/JOE.2013.2278891.

Pinto, J. et al. (June 2013). "The LSTS toolchain for networked vehicle systems". In: *2013 MTS/IEEE OCEANS - Bergen.* 2013 MTS/IEEE OCEANS - Bergen, pp. 1–9. DOI: 10.1109/OCEANS-Bergen.2013.6608148.

PyCharm (2019). *PyCharm: the Python IDE for Professional Developers by JetBrains.* JetBrains. URL: https://www.jetbrains.com/pycharm/ (visited on 12/17/2019).

Rasmussen, Carl Edward and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning.* Adaptive Computation and Machine Learning. Cambridge, Mass: The MIT Press. ISBN: 978-0-262-18253-9. URL: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=156015&site=ehost-live (visited on 05/22/2020).

Rego, Francisco C. et al. (Apr. 1, 2019). "Cooperative path-following control with logic-based communications: theory and practice". In: *Navigation and Control of Autonomous Marine Vehicles.* Ed. by Sanjay Sharma and Bidyadhar Subudhi. Institution of Engineering and Technology, pp. 187–224. DOI: 10.1049/PBTR011E_ch8. URL: https://digital-library.theiet.org/content/books/10.1049/pbtr011e_ch8 (visited on 06/04/2020).

Singh, Amarjeet et al. (2006). "Efficient Planning of Informative Paths for Multiple Robots". In: Publisher: eScholarship, University of California. URL: https://escholarship.org/uc/item/4r48w3bb (visited on 05/30/2020).

SINMOD (2020). SINTEF. Library Catalog: www.sintef.no. URL: http://www.sintef.no/ocean/satsinger/sinmod/ (visited on 05/20/2020).

Slagstad, Dag and Thomas A. McClimans (Oct. 1, 2005). “Modeling the ecosystem dynamics of the Barents sea including the marginal ice zone: I. Physical and chemical oceanography”. In: *Journal of Marine Systems* 58.1, pp. 1–18. ISSN: 0924-7963. DOI: 10.1016/j.jmarsys.2005.05.005. URL: http://www.sciencedirect.com/science/article/pii/S0924796305001296 (visited on 05/19/2020).

Sørensen, Asgeir J. and Martin Ludvigsen (Jan. 1, 2015). “Towards Integrated Autonomous Underwater Operations”. In: *IFAC-PapersOnLine.* 4th IFAC Workshop onNavigation, Guidance and Controlof Underwater VehiclesNGCUV 2015 48.2, pp. 107–118. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2015.06.018. URL: http://www.sciencedirect.com/science/article/pii/S2405896315002578 (visited on 06/04/2020).

Sture, Øystein (Dec. 1, 2019). *oysstu/pyimc*. original-date: 2017-06-02T13:27:40Z. URL: https://github.com/oysstu/pyimc (visited on 12/15/2019).

Walt, Stefan van der, S. Chris Colbert, and Gael Varoquaux (Mar. 2011). “The NumPy Array: A Structure for Efficient Numerical Computation”. In: *Computing in Science Engineering* 13.2. Conference Name: Computing in Science Engineering, pp. 22–30. ISSN: 1558-366X. DOI: 10.1109/MCSE.2011.37.

Appendix A

Code

Together with this master thesis follows a folder with Python files. These are structured as described in the associated file named `readme.txt`. The content in the file can be seen on next page:

```
Date:       08.06.2020
Author:     Fredrik Elgsaas Alnæs

---------------------------------------------------------------------------------
The files in the .zip-folder are a part of the master thesis, written as the
final part of the five years integrated master program Marine Technology,
with specialisation in underwater engineering at Norwegian University of
Science and Technology, Trondheim.
---------------------------------------------------------------------------------

main.py                 The main program deleting and creating text files and
                        starting the two simulation vehicles.

sim_1.py                Simulation code for "LAUV-SIMULATOR-1" with decision
                        model and Gaussian Process for doing adaptive sampling.

sim_2.py                Simulation code for "LAUV-SIMULATOR-2" with decision
                        model and Gaussian Process for doing adaptive sampling.

Pre_mission.py          The code finds start position of "LAUV-SIMULATOR-2" and
                        creats the temperature grid for actual temperature
                        based on wanted time and day, and predicted temperature
                        the mean of all temperatures considered.

plot.py                 The code loads all relevant text files and adjust them
                        so that the data can be plotted. It also calculates
                        the total error of actual temperature and predicted
                        temperature.

oceanserver.py          Creats a socket for the ocean data and sends ocean
                        data to the sim files based on lat, lon, time, depth
                        (received by Trygve Olav Fossum).

sinmoddata.py           Needed together with oceanserver.py, treated like a
                        black box (received by Trygve Olav Fossum).

samples_2017.05.11.nc   The ocean sample file containing all ocean info. This
                                    is read and separated in the above codes.

Licence                 Licence for using Trygve Olav Fossums Gaussian Process
                        model inside sim_1 and sim_2.

---------------------------------------------------------------------------------
```

Appendix B

All temperatures mission area

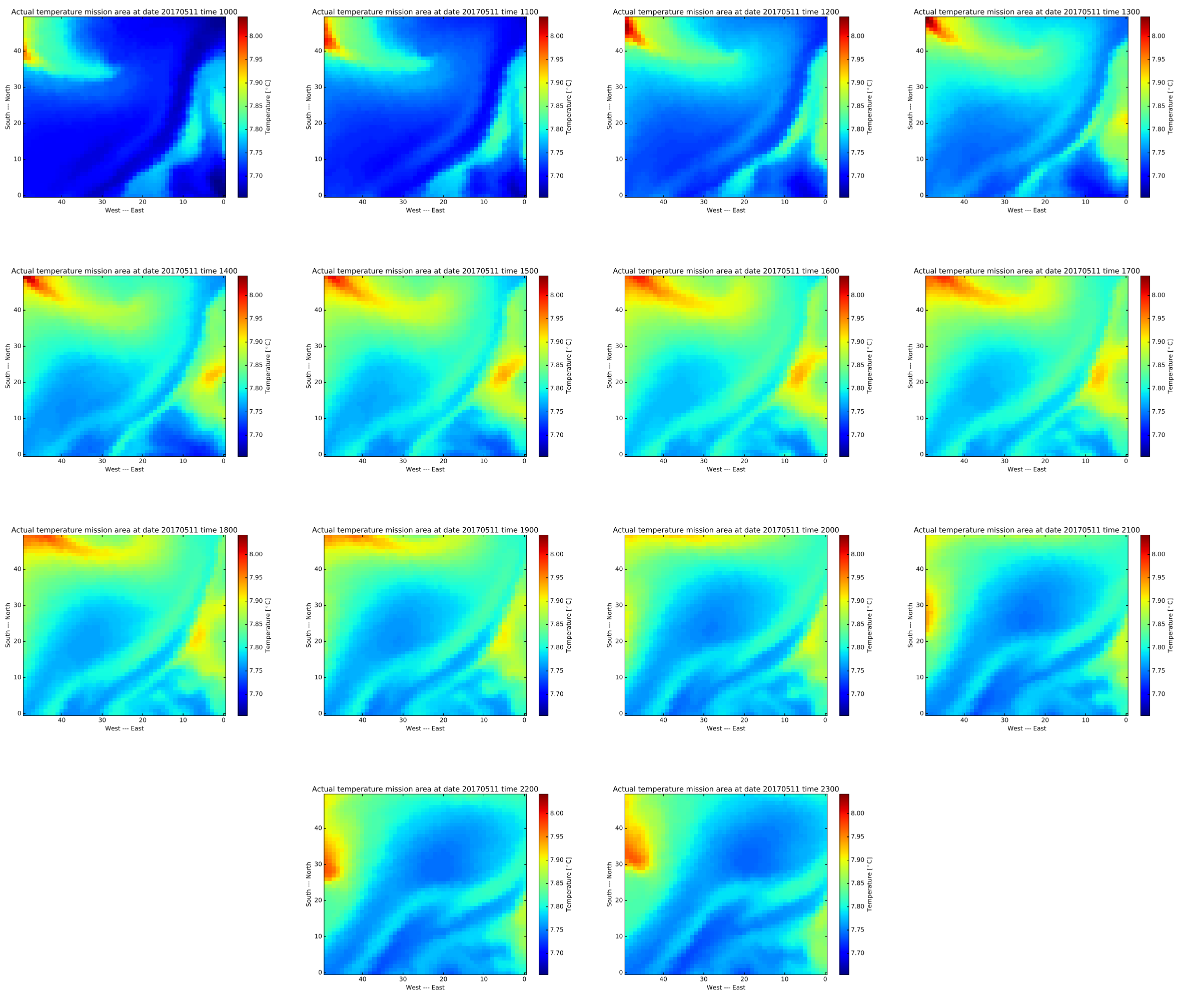


Figure B.1: Actual temperature for every timestep 10:00 to 23:00 day 1.

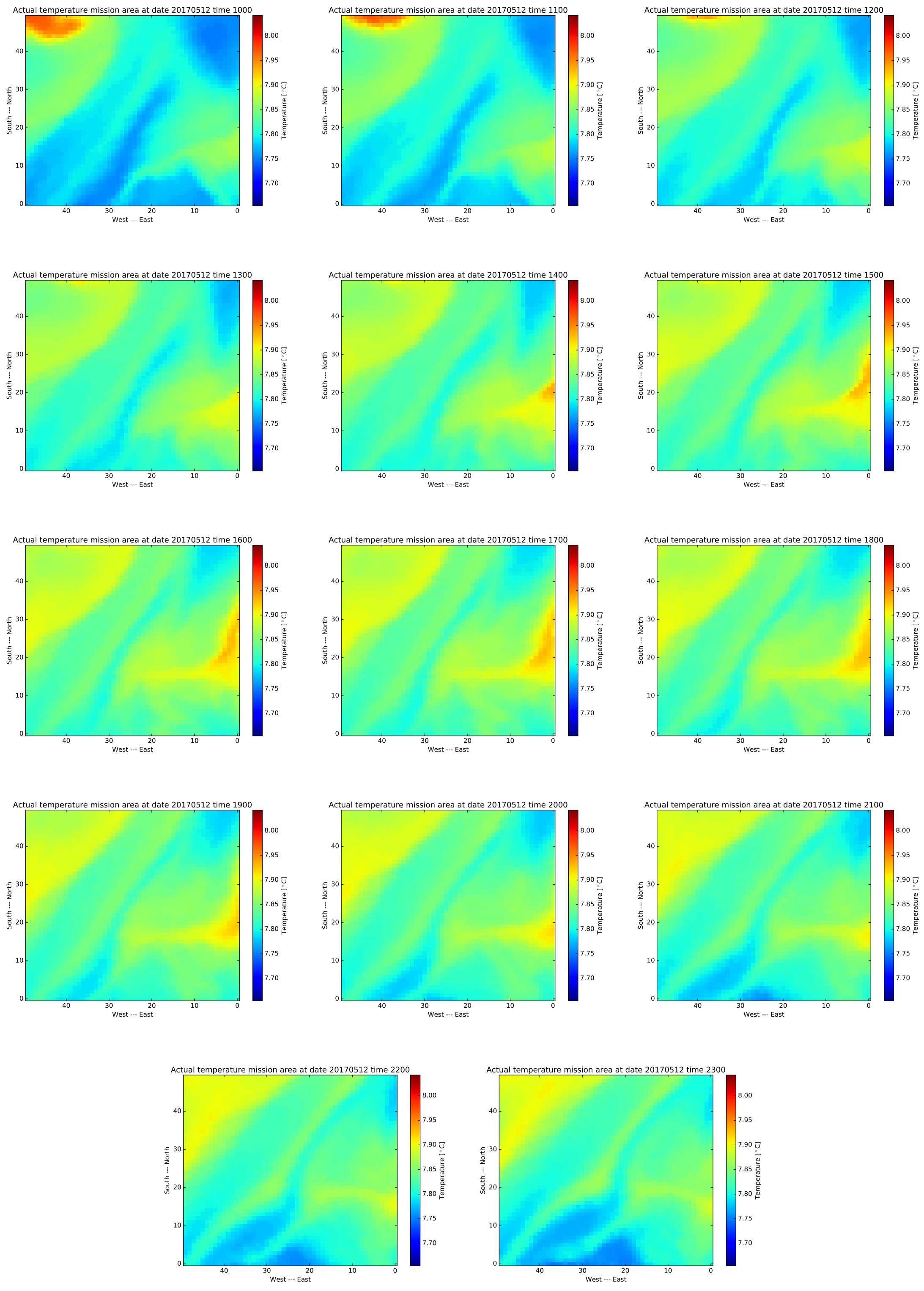


Figure B.2: Actual temperature for every timestep 10:00 to 23:00 day 2.