NTNU

Norwegian University of
Science and Technology

Optimization of Coordinated Path Planning for Autonomous Vehicles in Ice Management

Martin Hals

Martin Skjønhaug

Master of Science in Cybernetics and Robotics
Submission date: June 2017
Supervisor: Tor Arne Johansen, ITK
Co-supervisor: Frederik Leira, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Task Description

An application case study should consider iceberg threat detection and tracking for protecting offshore installations in arctic conditions. While satellite images might provide some infrequent updates of the positions of the largest icebergs, unmanned aerial vehicles can be deployed to check their current location, and also search for other icebergs not captured by satellite images.

The project will consider the use of several fixed wing unmanned aerial vehicles (UAVs) for such a search and tracking application, i.e simultaneously search an area to detect and identify unknown moving objects of interest, while at the same time tracking the motion of already detected objects. Due to the spatio-temporal extent of the problem, the task will involve coordinated control of a network of multiple UAVs to autonomously allocate resources within the fleet of UAVs at a given instant in time.

The following items should be considered:

- Survey the literature for relevant methods and applications.
- Propose algorithms for searching and tracking in a centralized, coordinated control network using a fleet of UAVs.
- Propose a method of target location estimation in a dynamic environment, using data gathered by the fleet of UAVs.
- Implement and simulate the system with a number of UAVs for iceberg search and tracking.
- If time permits, implement the system using the existing DUNE software framework from NTNU, and evaluate the system using software in the loop simulations as a preparation for future field testing.

Preface

This thesis is our final work of the five year Engineering Cybernetics program at the Norwegian University of Science and Technology (NTNU).

We would like to thank our supervisor, Tor Arne Johansen, for his guidance and inspiration throughout the process of this thesis. Furthermore, we would like to thank our co-supervisor, Frederik Leira, for his valuable support during many hours of helpful discussion.

Martin Skjønhaug and Martin Hals,
Trondheim, June 2017

Abstract

An important part of offshore operations in arctic climates is the management of drifting sea ice due to the dangers posed on ships and offshore structures. This thesis investigates the utilization of multiple unmanned aerial vehicles for ice monitoring. A cooperative search algorithm for a team of agents to autonomously monitor and search for sea ice in a dynamic environment is presented. The algorithm reduces the continuous search problem to a sequence of online, receding horizon optimizations on a finite, dynamically updated graph whose vertices represent potential waypoints for the agents to follow. Estimates of target locations are iteratively updated by agent measurements through Bayesian recursive estimation. The main contributions of this thesis are the incorporation of a predictive step in the iterative target estimation, enabling a dynamic environment for an unknown number of targets to be accounted for. Using the estimated target topology enables the formulation of a search objective with the intention of reducing the probability of target collision with an offshore installation. The system has been implemented using MATLAB, and software in the loop testing conducted to investigate the real time performance when integrated in a fully operational system. Simulations of a variety of scenarios have been conducted, revealing a significant increase in performance by increasing the size of the search party, both with regards to search time and threat detection. The proposed search objective yields an efficient cooperative behavior for teams of multiple agents, while additionally yielding good results for a single searching agent. However, simulations show that the predictive step is sensitive to errors in the estimation of the target dynamics, which could cause problems for practical applications as the calculation of the search objective relies heavily on the expected dynamics in the search region.

Sammendrag

En viktig del av offshore-operasjoner i arktiske strøk er håndtering av havis på grunn av farene den utgjør mot skip og oljeplattformer. Denne oppgaven undersøker bruken av opptill flere ubemannede fly til hensikt å monitorere havis. En samarbeidende søksalgoritme for et lag med agenter som monitorerer og søker etter havis i et dynamisk miljø blir presentert. Algoritmen forenkler det kontinuerlige søksproblemet til en sekvens av online, receding horizon optimaliseringer på en dynamisk oppdatert graf av endelig størrelse, hvor nodene representerer potensielle veipunkter som agentene kan følge. Estimater av lokasjonene til målene blir iterativt oppdatert av agentenes målinger gjennom Bayesisk rekursiv estimering. Hovedbidragene i denne oppgaven er innkorporeringen av et prediktivt steg i den iterative estimeringen til målenes posisjon, noe som muliggjør dynamiske søkskart for et ukjent antall mål. Ved å benytte den estimerte mål-topologien blir det mulig å formulere et søksobjektiv med intensjon om å redusere sannsynligheten for at mål skal kollidere med en oljeplatfrom. Systemet har blitt implementert ved hjelp av MATLAB, og tester med software in the loop har blitt gjennomført for å undersøke sanntidskapasitetene til programmet i et fullt operasjonelt system. Simuleringer av en rekke scenarioer har blitt gjennomført. Disse avdekket en betydelig økning av ytelsen ved å øke størrelsen på søkslaget, både med hensyn på søkstid og trusseldetektering. Det foreslåtte søksobjektivet leder til en effektiv, samarbeidende oppførsel for lag bestående av flere agenter, i tillegg til å gi gode resultater for kun én agent. Likevel viser simuleringer at det prediktive steget er sensitivt med tanke på feil i estimeringer av måldynamikken, noe som kan føre til problemer for praktiske applikasjoner siden kalkuleringen av søksobjektivet i stor grad baserer seg på den forventede dynamikken i søksområdet.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The motivation for this thesis is to use multiple unmanned aerial vehicles (UAVs), hereafter referred to as *agents*, for surveillance and tracking of ice to aid ice management for offshore installations in arctic conditions. Ice management is a crucial part of operations in arctic regions due to the dangers the drifting ice pose on ships and offshore structures. The following definition of ice management is proposed in [6]:

> Ice management is the sum of all activities where the objective is to reduce or avoid actions from any kind of ice features, hereafter referred to as *targets*. This will include, but is not limited to:
>
> - Detection, tracking and forecasting of sea ice, ice ridges and icebergs
> - Threat evaluation
> - Physical ice management such as ice breaking and iceberg towing
> - Procedures for disconnection of offshore structures applied in search for or production of hydrocarbons

Since ships are maneuverable, modern solutions for collision avoidance can be based on the use of unmanned aerial systems (UAS) to scout ahead, locate and track potentially dangerous icebergs and escort the ships in safe directions. On the other hand, offshore rigs are stationary and can not perform evasive maneuvers. Instead, information about approaching icebergs are gathered through a variety of means and icebergs posing a threat are usually towed away (Figure 1.1). If detected too late, chances are drilling operations will have to be stopped to brace for impact. Even though modern offshore installations are built to withstand tremendous forces, the cost of disconnecting and terminating drilling operations are significant. It is therefore important to detect potential threats at an early stage.

A common way to monitor ice and icebergs is by using Satellite Synthetic Aperture Radar (SAR) which is able to provide images day and night and in rough weather conditions. By the use of satellites, one can discover large icebergs, but this method is not sufficient in terms of detecting the smaller icebergs which can be missed due to too coarse image resolution. Furthermore, should the rate of provided satellite images be low, the information gained will be outdated as time passes. Helicopters and platform support vessels are also currently used as a method of collecting information about iceberg positions. Including the use of UAS in offshore drilling operations could be a great way to supply

already existing means of iceberg detection. A group of agents performing a cooperative search for potential threats to the platform would save both time and resources.

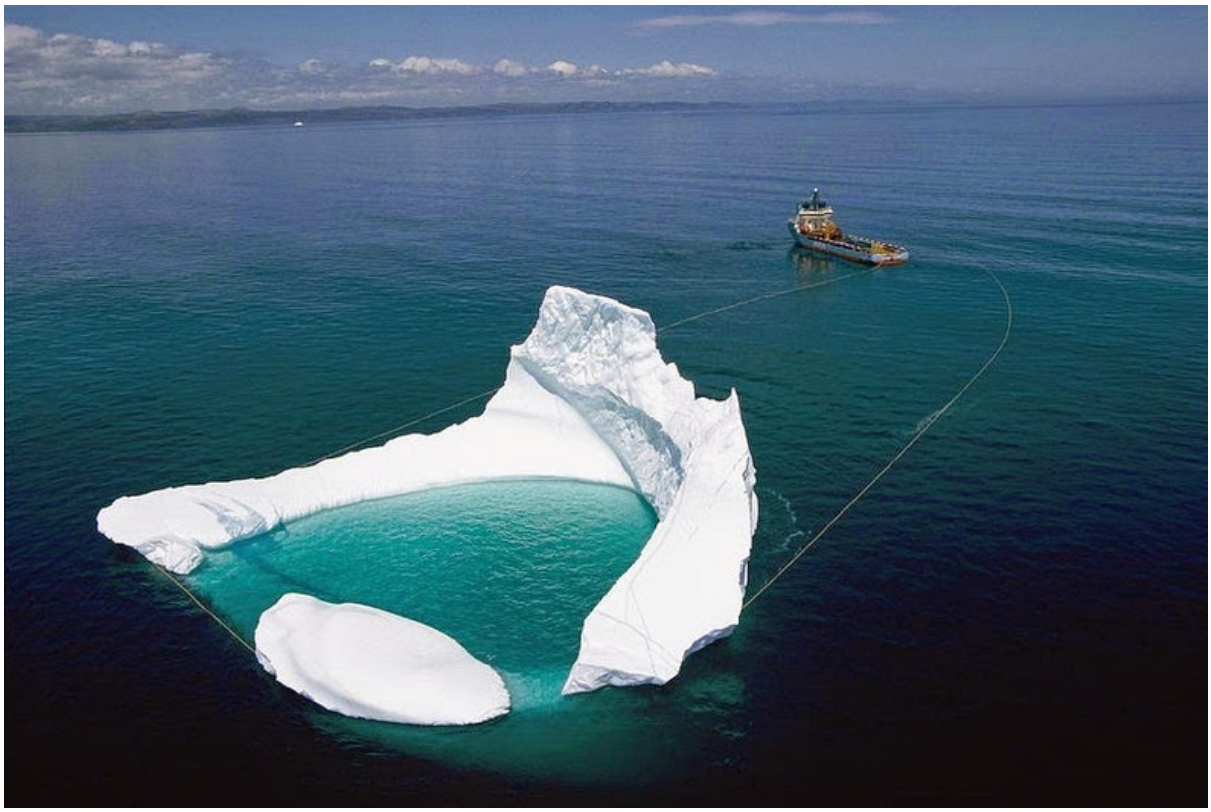


Figure 1.1: Towing an iceberg from a collision course with the Hibernia oil platform located in Newfoundland, Canada. Photo: Randy Olson.

1.1 Problem Description

The use of UAS would mainly contribute in two areas of modern ice management: detection and tracking of sea ice. The problem of autonomously locating and tracking moving targets is a large field of research, but when examining to what extent the use of UAS will contribute to ice management we will focus on the following questions:

- Would such a system add value to modern ice management operations?
- How could a reasonable search policy be formulated for the use case of ice management
- What is the benefit of deploying multiple agents compared to just one?
- How should the target dynamics in the operational area influence the behavior of the searching agents?
- How robust would the system be in terms of errors in the estimates of the target dynamics?
- How can the system account for prior information of iceberg locations?

- How will search strategies be influenced when dealing with both targets with and without prior information?
- How could the system handle both localization and tracking?
- Would the complexity of such a system make it feasible for online calculations in practical applications?

In this thesis we will try to answer these questions by designing a system capable of deploying multiple agents in a cooperative search in a given search region. The system should take into account probabilistic prior information of targets that have already been detected in addition to account for unknown targets with no prior information. Due to the size of the search region, the system should make strategic choices with regards to which areas to prioritize.

To accomplish this, we assume all agents are equipped with state of the art autopilots capable of navigating by waypoints. This reduces the search problem to a problem of cooperative path planning in terms of optimal waypoint placement. To disregard unimportant regions of the search area, we will propose a method of constructing a search priority grid map which will be dynamically updated throughout the search. Based on this grid map, a dynamically updated graph will represent the waypoints to consider when deciding paths for the agents. The system will be cooperative in the sense that the participating agents will share all information in a centralized fashion, in order to plan paths that will maximize the team's chances of detecting threats to the platform. The optimal paths will be calculated online using a receding horizon approach similar to the principles used in model predictive control (MPC).

1.2 Previous Work

Modern search theory was pioneered by Bernard Koopman and was motivated by the problem of finding enemy marine vessels [10]. Since then, the number of applications have greatly increased and now encompass fields like search and rescue, surveillance and exploration. In the years following the declassification and release of Koopman's work, multiple variations and extensions to his problems were explored, mostly solved by a dynamic programming approach [12]. Stephen M. Pollock introduced a Bayesian approach where decisions are made sequentially based on all previous measurements to minimize a search cost [16]. He also considered the case of a moving target under the assumption that a target moves from one region to another in a Markovian fashion and with known parameters. A useful simplification in search theory is to discretize the search space and formulate the search problem with a graph. This will reduce the path of the searching agent to a piecewise linear path, but the problem is still computationally heavy. DasGupta et al. showed that the search problem for a stationary target is NP-hard, but also presented approximate algorithms for solving it in polynomial time using graph partitioning [4].

All of the mentioned works are of a single searching agent. Once we add multiple agents to cooperatively search for targets the problem will increase considerably in complexity. Several approaches have been made to reduce this complexity. In [3], hierarchical decompositions were performed on the tasks, and the searching agents were divided into sub-teams. Another approach is to simplify the problem by restricting the optimizations to a finite, receding horizon. This will reduce the complexity of the optimization problem,

but we are still required to solve an NP-hard search problem at every time step, which puts limitations on the length of the prediction horizon. To address this problem, [9] proposed a method of performing the receding horizon optimization on a dynamically changing graph where the nodes are placed only in regions with high target probability. This way the searching agents only perform detailed searches in areas of high target probability and can evaluate long paths just as efficiently as short paths.

This thesis extends the authors' work [17], where the use of a dynamically updated search graph was introduced in the context of protecting an offshore installation from incoming targets. However, knowledge of the target dynamics was not incorporated in the estimation of prioritized search areas prior to the search mission. Instead, the search priority for unknown targets was linearly decreasing from the offshore rig, and the direction of the search area of interest from the rig was assumed given. A clear mathematical problem description was not formulated, and the search policy of the agents was based on intuition rather than optimization in the sense of avoiding sea ice collision.

1.3 Structure of the Thesis and Main Contributions

The reminder of this thesis is structured as follows:

Chapter 2: This chapter presents the relevant background theory used in this thesis, including already established knowledge and new deviations. This chapter presents the mathematical fundaments the system implementation is built on, and is mainly divided into three sections:

> **Estimation of Target Location:** An iterative Bayesian estimation algorithm based on [19] is used to estimate the target locations from sensor measurements. The main contribution is the proposal of a predictive step in the iterative estimation algorithm to account for the search map dynamics. The prediction is found to be expressed as a convolution between all posterior estimates of target locations and a matrix encapsulating the the target dynamics.
>
> **Search Objective Formulation:** In this section we create and propose an iterative procedure to estimate the probabilities of collision from individual positions in the search region. The probability estimates are based on the matrix encapsulating the target dynamics. An optimization problem with respect to the search team configurations is formulated to minimize the probability of collision between targets and the offshore installation. The optimization objective function is based on the probabilities of collision from individual positions in the search region.
>
> **Model Predictive Search:** In order to relate the search problem to the task of finding a set of optimal waypoints for the team of agents to follow, a receding horizon search algorithm first presented in [9] will be described. By proposing a method of incorporating the objective function from the previous section, the receding horizon optimization will choose agent paths minimizing the probability of collision. The main contribution of this section is the incorporation of the proposed objective function in the model predictive search, enabling an optimal search for an unknown number of targets in a dynamic environment.

Chapter 3: This chapter presents the system architecture of the implementation of the system described in Chapter 2, and a brief description of the different key modules will be

given. A software in the loop (SITL) test environment has been developed to test the real time performance of the implemented system. The architecture of the test environment and the different software components used in the SITL simulations will be described.

Chapter 4: Simulation results from application based test scenarios and parameter adjustment sensitivity are presented by using the simulation environment created and presented in Chapter 3. The simulation results are analyzed to give a brief indication of the performance and scalability of the algorithm, and what kind of estimation errors it is prone to.

Chapter 5 and 6: Discussion of theory and results, concluding remarks and future perspectives are presented in the final chapters.

Chapter 2

Theory

2.1 UAV and Sensor Model

As opposed to a multi-rotor UAV, a fixed wing UAV would be the platform of choice for ice management due to the requirements on range and flight time. Since a fixed wing aircraft is non-holonomic, there are certain constraints given by the dynamics of the UAV. These can be modeled as a set of coupled nonlinear differential equations, but for the purpose of this thesis a detailed aircraft model will not be outlined, as the low level aircraft dynamics is outside the scope of this thesis. It is however important to keep in mind that certain constraints like maximum bank angle, maximum thrust and minimum speed will need to be taken into account to assign feasible waypoints for the UAV. However, the principles of our system can be tested regardless of the specific aircraft constraints. We will therefore make the simplistic assumption that the UAVs at all times move at constant speed in a piecewise linear path constituted by the line segments between the waypoints assigned.

In order to detect sea ice, the UAVs are equipped with sensors capable of taking measurements of the environment at a given frequency. This could for instance be a thermal imaging camera which have proven to be a suitable choice of sensor when attempting to detect ice features. To be able to cover as much ground as possible, the camera is usually mounted on a stabilized gimbal to ensure sufficient camera movement. The region that the camera can view at a particular time instant is called the *field of view* (FOV) and in order to relate features in captured images to the physical world, a projection model dependent on the UAV and gimbal pose is required. This means the optimization problem of searching for moving targets can be extended to also finding the optimal direction to aim the camera at all times throughout the search. This thesis will only consider the path planning aspect of the search and therefore assume the camera is stabilized and always pointing straight downwards relative to the UAV. To be able to automatically classify ice features captured by the camera, computer vision software is required in order to detect the sea ice. Image processing and camera calibration is however outside the scope of this thesis, and we will assume the computer vision software is fully implemented.

2.2 Estimation of Target Location

A key assumption in this thesis is that the prior location of a target is either not known or only partially known. This means we have an approximate location of the target with a varying degree of uncertainty, which needs to be updated as the agents proceed into

the search. In other words we need an estimator that characterize the target probability density function (pdf) in a discrete approximation. One approach is to use particle filters as target location estimators [1], but one is also able to incorporate nonlinear dynamic and non-Gaussian distributions using a grid based probabilistic map. Estimating a map from a moving platform is a challenging task in the field of robotics. Map estimation cannot be separated from the localization problem and difficulty arises when errors in the estimates of the agent location is incorporated into the map. This problem is referred to as simultaneous localization and mapping (SLAM). For this thesis we will use the simplified approach where we assume the poses of the agents are fully known at all time which is a reasonable assumption as most UAVs are equipped with accurate GNSS and IMU sensors in order to estimate position and attitude.

2.2.1 Iterative Estimation of Occupancy Grid Map

For our application of sea ice management, we will represent the search area as a simplified world fixed grid map referred to as an occupancy grid map. An occupancy grid map is a representation of the search area which is divided into grid cells, where each cell either contains or does not contain a target. The occupancy grid map represents the true state of the search area and is thus unknown to the searching agents. However, the agents will in a probabilistic way try to estimate the state of the grid cells by continuous measurement. The iterative Bayesian estimation algorithm presented in this section is based on the work in [19] with inspiration from [7].

Let m_t denote the occupancy grid map

$$m_t = \{\mathbf{m}_{i,t}\} \tag{2.1}$$

where $\mathbf{m}_{i,t}$ represents grid cell i with an associated binary occupancy value. We wish to estimate the posterior probability density of all possible maps given all past measurements $z_{1:t} = \{z_1, z_2, \ldots, z_t\}$ and agent poses $x_{1:t} = \{x_1, x_2, \ldots, x_t\}$. This is commonly referred to as the *belief* of the state of the map at time t

$$bel(m_t) = p(m_t|z_{1:t}, x_{1:t}) \tag{2.2}$$

An important assumption about the stochastic process m_t is that it is Markov. What this essentially means is that given the current state m_t, no additional information is gained by considering previous states and measurements when trying to predict m_{t+1}. In other words, m_t is complete in the sense that it summarizes everything that has happened in the past. The Markov assumption can be expressed by the conditional independence

$$p(m_t|m_{t-1}, z_{1:t-1}, x_{1:t-1}) = p(m_t|m_{t-1}) \tag{2.3}$$

It is often convenient to calculate a posterior before taking the most recent measurement into account. This is called the predictive belief and will be denoted as

$$\overline{bel}(m_t) = p(m_t|z_{1:t-1}, x_{1:t-1}) \tag{2.4}$$

A recursive Bayesian estimation algorithm, or Bayesian filter, calculates the posterior $bel(m_t)$ from a prior $bel(m_{t-1})$ and a new measurement z_t. First, by marginalizing over all previous states, (2.4) can be rewritten as

$$\begin{aligned} \overline{bel}(m_t) &= \int p(m_t|m_{t-1}) p(m_{t-1}|z_{1:t-1}, x_{1:t-1}) dm_{t-1} \\ &= \int p(m_t|m_{t-1}) bel(m_{t-1}) dm_{t-1} \end{aligned} \tag{2.5}$$

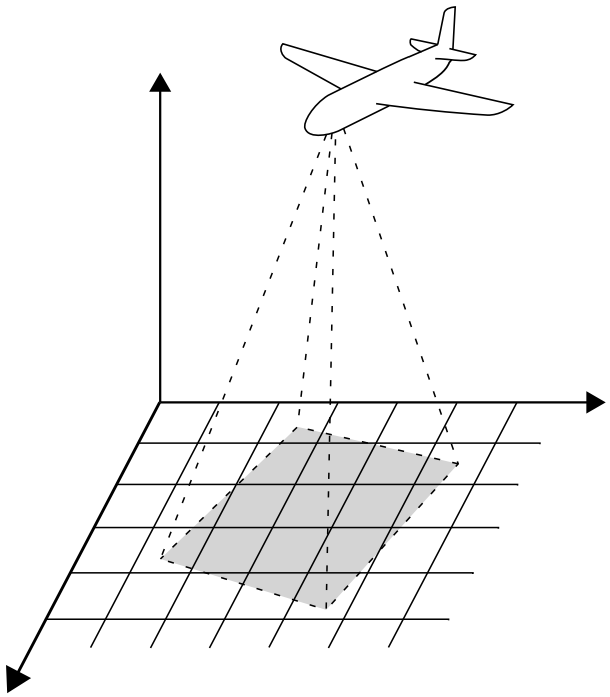


Figure 2.1: An agent performing a sensor measurement z_t of all occupancy grid cells in the agent's field of view.

By applying Bayes' rule such that

$$p(m_t, z_t|z_{1:t-1}, x_{1:t}) = p(z_t|m_t, z_{1:t-1}, x_{1:t})p(m_t|z_{1:t-1}, x_{1:t}) \tag{2.6}$$
$$= p(m_t|z_t, z_{1:t-1}, x_{1:t})p(z_t|z_{1:t-1}, x_{1:t}) \tag{2.7}$$

and by setting (2.6) and (2.7) equal to each other we get the following expression for the belief

$$p(m_t|z_{1:t}, x_{1:t}) = \frac{p(z_t|m_t, z_{1:t-1}, x_{1:t})p(m_t|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \tag{2.8}$$

Now we exploit the assumption that our state m_t is complete in the sense of Markov. We know that when trying to predict the measurement z_t, no additional information is gained by considering previous measurements given that we know the state m_t. Furthermore, even though the agent pose x_t does not predate the state m_t, it will not influence the state of the map without its accompanying measurement and can therefore be omitted. This can all be expressed by the following conditional independence

$$p(z_t|m_t, z_{1:t-1}, x_{1:t}) = p(z_t|m_t) \tag{2.9}$$

Hence (2.8) can be reduced to

$$\begin{aligned} p(m_t|z_{1:t}, x_{1:t}) &= \frac{p(z_t|m_t)p(m_t|z_{1:t-1}, x_{1:t})}{p(z_t|z_{1:t-1}, x_{1:t})} \\ &= \eta p(z_t|m_t)p(m_t|z_{1:t-1}, x_{1:t}) \end{aligned} \tag{2.10}$$

where η can be considered a normalizing term. From (2.5) and (2.10) we get the following recursive update equations

$$\overline{bel}(m_t) = \int p(m_t|m_{t-1})bel(m_{t-1})dm_{t-1} \tag{2.11}$$

$$bel(m_t) = \eta p(z_t|m_t)\overline{bel}(m_t) \tag{2.12}$$

Since an occupancy grid map consists of multiple grid cells $\mathbf{m}_{i,t} \in \{0, 1\}$, the marginalization of all possible occupancy map permutations in (2.11) will in most applications be too computationally heavy. Thus it makes sense to consider each grid cell individually. By doing this we will first assume the grid cell densities are independent, and neighboring cells will not affect each other. This can be explained in mathematical terms as

$$p(m_t|z_{1:t}, x_{1:t}) = \prod_i p(\mathbf{m}_{i,t}|z_{1:t}, x_{1:t}) \tag{2.13}$$

By considering individual grid cells we reduce the marginalization over all possible maps in (2.11) to the much simpler marginalization over the previous states of the single grid cell

$$\overline{bel}(\mathbf{m}_{i,t}) = \sum_{\mathbf{m}_{i,t-1}} p(\mathbf{m}_{i,t}|\mathbf{m}_{i,t-1})bel(\mathbf{m}_{i,t-1}) \tag{2.14}$$

where $p(\mathbf{m}_{i,t}|\mathbf{m}_{i,t-1})$ is a state transition probability encoding the dynamics of the map. Due to our assumption of independence between grid cells, map dynamics is included only by considering the state of a single grid cell given its previous state. One might argue this is overly simplistic, and section 2.2.2 of this thesis is solely dedicated to the predictive step $\overline{bel}(\mathbf{m}_{i,t})$, proposing a slightly more sophisticated approach to map dynamics. We get the final step of the Bayesian recursion by applying Bayes' rule

$$\begin{aligned} bel(\mathbf{m}_{i,t}) &= p(\mathbf{m}_{i,t}|z_{1:t}, x_{1_t}) \\ &= \frac{p(z_t|\mathbf{m}_{i,t}, x_t)p(\mathbf{m}_{i,t}|z_{1:t-1}, x_{1:t-1})}{p(z_t|z_{1:t-1}, x_{1:t})} \\ &= \frac{p(z_t|\mathbf{m}_{i,t}, x_t)\overline{bel}(\mathbf{m}_{i,t})}{p(z_t|z_{1:t-1}, x_{1:t})} \end{aligned} \tag{2.15}$$

where $p(z_t|\mathbf{m}_{i,t}, x_t)$ is the probability relating the most recent measurement to the state of the cell and the agent pose. This probability density might however be very hard to define as we are required to describe every possible measurement given the agent pose and cell state. In our case the measurement is much more complex than the simple binary state of $\mathbf{m}_{i,t}$. It is therefore easier to specify a distribution over the binary cell state as a function of the measurement. This is referred to as an *inverse measurement model* and is often much easier to implement than the traditional forward model. By applying Bayes' rule to the measurement model we obtain

$$\begin{aligned} p(z_t|\mathbf{m}_{i,t}, x_t) &= \frac{p(\mathbf{m}_{i,t}|z_t, x_t)p(z_t|x_t)}{p(\mathbf{m}_{i,t}|x_t)} \\ &= \frac{p(\mathbf{m}_{i,t}|z_t, x_t)p(z_t|x_t)}{p(\mathbf{m}_{i,t})} \end{aligned} \tag{2.16}$$

Next, inserting this into (2.15) yields

$$bel(\mathbf{m}_{i,t}) = \frac{p(\mathbf{m}_{i,t}|z_t, x_t)p(z_t|x_t)\overline{bel}(\mathbf{m}_{i,t})}{p(\mathbf{m}_{i,t})p(z_t|z_{1:t-1}, x_{1:t})} \tag{2.17}$$

By taking advantage of the fact that the grid cell states are binary, we can calculate the probability of the negated state by

$$p(\neg\mathbf{m}_{i,t}) = 1 - p(\mathbf{m}_{i,t}) \tag{2.18}$$

We define the *odds ratio* to be the fraction between the belief and the negated belief. By assuming the belief has the same property as in (2.18) we can cancel out a few terms not dependent on the cell state to get

$$\frac{bel(\mathbf{m}_{i,t})}{bel(\neg\mathbf{m}_{i,t})} = \frac{\overline{bel}(\mathbf{m}_{i,t})p(\mathbf{m}_{i,t}|z_t, x_t)p(\neg\mathbf{m}_{i,t})}{\overline{bel}(\neg\mathbf{m}_{i,t})p(\neg\mathbf{m}_{i,t}|z_t, x_t)p(\mathbf{m}_{i,t})} \tag{2.19}$$

Defining $l_{i,t}$ to be the log of the odds ratio we get the additive update equation

$$l_{i,t} = \log\frac{\overline{bel}(\mathbf{m}_{i,t})}{1-\overline{bel}(\mathbf{m}_{i,t})} + \log\frac{p(\mathbf{m}_{i,t}|z_t, x_t)}{1-p(\mathbf{m}_{i,t}|z_t, x_t)} - \log\frac{p(\mathbf{m}_{i,t})}{1-p(\mathbf{m}_{i,t})} \tag{2.20}$$

The first term in this update equation includes the predictive belief to account for map dynamics. The second term includes the most recent measurement through the inverse measurement model $p(\mathbf{m}_{i,t}|z_t, x_t)$. The return value of the inverse sensor measurement model is a design parameter decided by how reliable the sensor measurements are, and enables us to account for the possibility of false detection. The last term is the prior log odds ratio and can be seen as a quantity returned by the inverse measurement model when nothing is added through a measurement. For grid cells not in the perceptual field of any agents there are no measurements to influence the belief. Thus, for all grid cells outside the field of view of the searching agents, the belief is updated only by the predictive step

$$l_{i,t} = \log\frac{\overline{bel}(\mathbf{m}_{i,t})}{1-\overline{bel}(\mathbf{m}_{i,t})} \tag{2.21}$$

By using the log odds representation in the Bayesian recursion, as opposed to the regular probability representation, we avoid instabilities for probabilities close to zero. The probabilities can be recovered at any time from the log odds ratio by

$$p(\mathbf{m}_{i,t}|z_{1:t}, x_{1:t}) = \frac{\exp(l_{i,t})}{1+\exp(l_{i,t})} \tag{2.22}$$

2.2.2 Map Dynamics

In order to calculate the belief of each individual grid cell according to (2.15) we made the assumption that all occupancy grid cells are independent. However, in the case of a dynamic occupancy grid map it arguably makes sense that a grid cell to some degree is dependent of the state of other cells. Due to external forces in the search environment there is a probability that the state of one grid cell will transition over to a neighboring cell. We will now propose a way to account for this transitional probability by relaxing the constraint of independent grid cells during the calculation of the predictive belief $\overline{bel}(\mathbf{m}_{i,t})$.

In order change the calculation of the predictive belief for the entire search map (2.11) to that of a single grid cell (2.14), we assumed all cells independent in order to get the simple marginalization over one binary state. For the purpose of calculating the predictive belief, let us instead assume the state of a grid cell $\mathbf{m}_{i,t}$ is *dependent* on the previous state of all grid cells in a neighborhood D. This means $\mathbf{m}_{i,t}$ is independent of all previous grid cell states except the set of states

$$\{\mathbf{m}_{j,t-1}\}, \quad \forall j \in D \tag{2.23}$$

To calculate the predictive belief we now need to marginalize over all previous states of all grid cells in D. We also assume the contributions from the grid cells j to the single cell i are additive. (2.11) can then be reformulated as

$$\begin{aligned}\overline{bel}(\mathbf{m}_{i,t}) &= p(\mathbf{m}_{i,t}|z_{1:t-1}, x_{1:t-1}) \\ &= \sum_{j\in D}\sum_{\mathbf{m}_{j,t-1}} p(\mathbf{m}_{i,t}|\mathbf{m}_{j,t-1})p(\mathbf{m}_{j,t-1}|, z_{1:t-1}, x_{1:t-1}) \\ &= \sum_{j\in D}\sum_{\mathbf{m}_{j,t-1}} p(\mathbf{m}_{i,t}|\mathbf{m}_{j,t-1})bel(\mathbf{m}_{j,t-1})\end{aligned} \tag{2.24}$$

The term $p(\mathbf{m}_{i,t}|\mathbf{m}_{j,t-1})$ will tell us something about the probability that the previous state of cell j will influence the current state of cell i. Since the grid cell states are binary, we can write out (2.24) as

$$\sum_{j\in D}\Big[p(\mathbf{m}_{i,t}|\mathbf{m}_{j,t-1})bel(\mathbf{m}_{j,t-1}) + p(\mathbf{m}_{i,t}|\neg\mathbf{m}_{j,t-1})bel(\neg\mathbf{m}_{j,t-1})\Big] \tag{2.25}$$

$p(\mathbf{m}_{i,t}|\neg\mathbf{m}_{j,t-1})$ is describing the probability of the state of cell i given the neighbor j had the opposite state. Since we are considering the individual contribution in an additive fashion, there is no way the previous state of cell j alone can give cell i the opposite state. Therefore

$$p(\mathbf{m}_{i,t}|\neg\mathbf{m}_{j,t-1}) = 0 \tag{2.26}$$

This leaves us with the following expression for the predictive belief

$$\overline{bel}(\mathbf{m}_{i,t}) = \sum_{j\in D} p(\mathbf{m}_{i,t}|\mathbf{m}_{j,t-1})bel(\mathbf{m}_{j,t-1}) \tag{2.27}$$

By letting the cells of the neighborhood D be a square section of the occupancy grid map with $2n+1$ cells in each direction and cell i in the center, we can represent the neighborhood D in the matrix form

$$K_{(u,v),t} \in \mathbb{R}^{2n+1\times 2n+1} \tag{2.28}$$

Here we replace the notation of a given cell i by its equivalent row position u and column position v in the context of an occupancy grid map in matrix form. The elements in $K_{(u,v),t}$ are the state transition probabilities from the respective cells in D

$$K_{(u,v),t} = \begin{bmatrix} p(\mathbf{m}_{(u,v),t}|\mathbf{m}_{(u-n,v-n),t-1}) & \cdots & p(\mathbf{m}_{(u,v),t}|\mathbf{m}_{(u-n,v+n),t-1}) \\ \vdots & \ddots & \vdots \\ p(\mathbf{m}_{(u,v),t}|\mathbf{m}_{(u+n,v-n),t-1}) & \cdots & p(\mathbf{m}_{(u,v),t}|\mathbf{m}_{(u+n,v+n),t-1}) \end{bmatrix} \tag{2.29}$$

The representation of the transitional probabilities in matrix form enables us to write (2.27) as

$$\overline{bel}(\mathbf{m}_{(u,v),t}) = \sum_{k=-n}^{n}\sum_{l=-n}^{n} p(\mathbf{m}_{(u,v),t}|\mathbf{m}_{(u-k,v-l),t-1})bel(\mathbf{m}_{(u-k,v-l),t-1}) \tag{2.30}$$

In this sense it is the kernel $K_{(u,v),t}$ that determines the effect of map dynamics on the posterior probabilities of target locations. Thus, we need a way to approximate the

elements of $K_{(u,v),t}$. In the scenario of drifting sea ice, the term *metocean data* refer to the main three effects that will influence target movement: wind, ocean currents and waves. However, the size and shape of icebergs will influence the dynamics such that two different targets might not get the same trajectories for the same metocean data. Due the the fact that we necessarily do not know the target shape prior to detection and due to error sources in the metocean data, there naturally is an element of uncertainty to the target dynamics. It is outside the scope of this thesis to employ metocean data to analytically estimate the dynamics of drifting sea ice, but for the purpose of this thesis we will make the following assumption:

> *Based on information like metocean data, weather forecasts and historical target trajectories, a probability density function representing the target velocity vector at position* (x, y) *can be found.*

This means all locations (x, y) in the search area have a corresponding target vector represented by a probability density function, as illustrated in Figure 2.2. For more

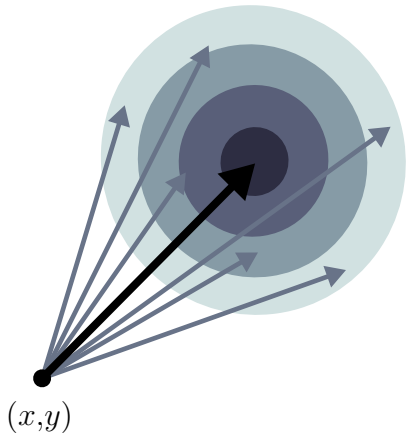


Figure 2.2: Target velocity vector modeled with a probability density function.

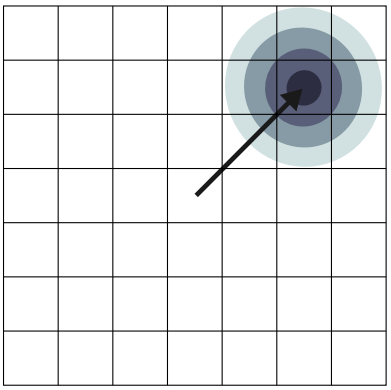
detailed information on uncertainty in vector fields, see [13]. Next, let the probability density function of the target velocity vector of an occupancy grid cell (u, v) be discretized as the grid cell matrix

$$\mathcal{K}_{(u,v),t} \in \mathbb{R}^{2n+1 \times 2n+1} \tag{2.31}$$

which is of the same dimesions as $K_{(u,v),t}$. Figure 2.3 illustrates an example of how $\mathcal{K}_{(u,v),t}$ is approximated by the pdf of a target velocity vector. It should now be apparent that $\mathcal{K}_{(u,v),t}$ is directly related to the transitional probabilities. However, in contrast to $K_{(u,v),t}$ which contains the probabilities $p(\mathbf{m}_{i,t}|\mathbf{m}_{j,t-1})$, $\mathcal{K}_{(u,v),t}$ contains the elements $p(\mathbf{m}_{j,t}|\mathbf{m}_{i,t-1})$ which describes the probability of a target transitioning from the center cell i to a neighbor j. This means $\mathcal{K}_{(u,v),t}$ can be written as

$$\mathcal{K}_{(u,v),t} = \begin{bmatrix} p(\mathbf{m}_{(u-n,v-n),t}|\mathbf{m}_{(u,v),t-1}) & \cdots & p(\mathbf{m}_{(u-n,v+n),t}|\mathbf{m}_{(u,v),t-1}) \\ \vdots & \ddots & \vdots \\ p(\mathbf{m}_{(u+n,v-n),t}|\mathbf{m}_{(u,v),t-1}) & \cdots & p(\mathbf{m}_{(u+n,v+n),t}|\mathbf{m}_{(u,v),t-1}) \end{bmatrix} \tag{2.32}$$

In order to relate $\mathcal{K}_{(u,v),t}$ and $K_{(u,v),t}$, we will model the vector field of target velocity vectors in the search region as uniform with the same probability density function for



$$\implies \mathcal{K}_{(u,v),t} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.095 & 0.118 & 0.095 \\ 0 & 0 & 0 & 0 & 0.118 & 0.1478 & 0.118 \\ 0 & 0 & 0 & 0 & 0.095 & 0.118 & 0.095 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 2.3: Conceptual illustration of how the matrix $\mathcal{K}_t$ is a discretization of the probability density function of the target velocity vector.

the entire region, which is possible since we can incorporate uncertainty in the velocity vectors. This means transitional probabilities in a neighborhood relative to a grid cell $\mathbf{m}_{(u,v),t}$ will be the same regardless of the position of $\mathbf{m}_{(u,v),t}$. Thus the matrices $\mathcal{K}_{(u,v),t}$ and $K_{(u,v),t}$ will be identical for all (u, v) in the occupancy grid map and will therefore be referred to as K_t and $\mathcal{K}_t$ for the remainder of this thesis. By considering the two matrices we can see that $\mathcal{K}_t$ describes the transfer of information from the center cell in D to the neighbors, while K_t describes the opposite transfer from the neighbors to the center cell. Furthermore, since the transitional probabilities are assumed constant relative to the center cell i for all neighborhoods in the search area it makes sense that the two matrices $\mathcal{K}_t$ and K_t are flipped versions of each other, both horizontally and vertically. The relation can be described mathematically as

$$K_t = J_{2n+1}\mathcal{K}_t J_{2n+1} \tag{2.33}$$

where $J_{2n+1} \in \mathbb{R}^{2n+1 \times 2n+1}$ is the exchange matrix

$$J_{2n+1} = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \iddots & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{bmatrix} \tag{2.34}$$

(2.34) is often referred to as the reversal matrix due to its property of flipping a matrix around its vertical axis by pre-multiplication and around its horizontal axis by post-multiplication.

Due to the relationship (2.33) it is of interest to note that calculating (2.30) for every cell in the occupancy grid map amounts to a two dimensional spatial convolution. The operation performed on the entire occupancy grid map can therefore be written as the convolution

$$\overline{bel}(m_t) = \mathcal{K}_t * bel(m_{t-1}) \tag{2.35}$$

Drawing parallels to the field of image processing this can be seen as convolution where $bel(m_{t-1})$ represents an image and $\mathcal{K}_t$ a filtering kernel. For example, letting $\mathcal{K}_t$ represent a target vector with a Gaussian distribution, (2.35) would amount to a Gaussian blur, thus smoothing out the posterior occupancy grid map. For more information of the use of convolution in image processing, we refer the reader to [8]. It should now be more clear

why it is important to keep $\mathcal{K}_t$ spatially independent. By letting $\mathcal{K}_t$ change depending on the position of the occupancy grid cell we risk existing posterior information being duplicated or removed. Furthermore, due to the fact that $\mathcal{K}_t$ represents a discretization of a pdf we know

$$\sum_{k=-n}^{n}\sum_{l=-n}^{n} p(\mathbf{m}_{(u-k,v-l),t}|\mathbf{m}_{(u,v),t-1}) = 1 \tag{2.36}$$

which ensures the sum of posterior information is preserved after the operation (2.35). Even though the transitional probabilities are assumed spatially independent it is important to note $\mathcal{K}_t$ can be *time* dependent. As the conditions in the search area change, the probability density function of the target velocity vector should be updated accordingly, causing $\mathcal{K}_t$ to change.

Figure 2.4 shows simulation results for the iterative Bayesian recursion algorithm with map dynamics incorporated by (2.35). The simulation is without sensor measurements, so only the dynamics of the map is regarded. The sum of probabilities in all grid cells were found to be the same before and after every convolution.

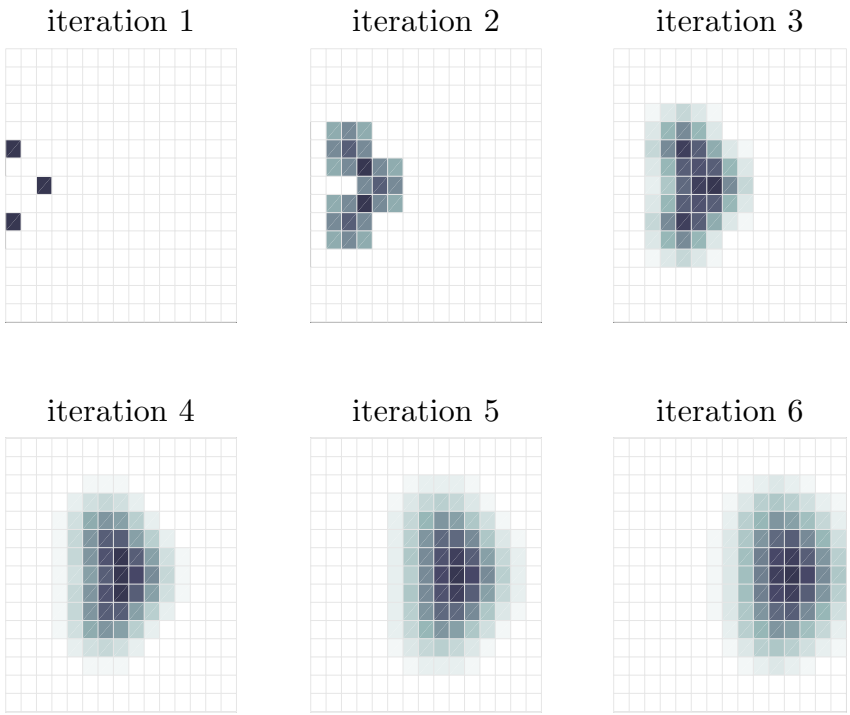


Figure 2.4: Simulation of convolution with a Gaussian kernel $\mathcal{K}_t$. The sum of the belief of all grid cells before and after every iteration is the same.

2.2.3 Algorithm for Iterative Occupancy Grid Map Estimation

As a summary, this section will present the complete iterative Bayesian recursion algorithm with map dynamics included. The algorithm requires a prior estimate of the occupancy grid map $bel(m_{t-1})$ and a transition probability kernel K_t. The posterior

probabilities of the occupancy grid map can be estimated recursively using the algorithm listed in Algorithm 1.

Algorithm 1 Occupancy Grid Mapping by Bayesian Recursion

procedure OccupancyGridMapping(l_{t-1}, K_t, z_t, x_t)
  for all cells i **do**
    $bel(\mathbf{m}_{i,t-1}) = \exp(l_{i,t-1})/(1 + \exp(l_{i,t-1}))$
    Calculate $\overline{bel}(\mathbf{m}_{i,t})$ according to (2.30)
    if cell i in perceptual field of z_t **then**
      Calculate $l_{i,t}$ according to (2.20)
    else
      Calculate $l_{i,t}$ according to (2.21)
    end if
  end for
  return l_t
end procedure

Figure 2.5 illustrates the iterative map estimation described in Algorithm 1 for a single agent. The agent is traveling in a straight line from west to east and the map dynamics are based on a target velocity pointing south.

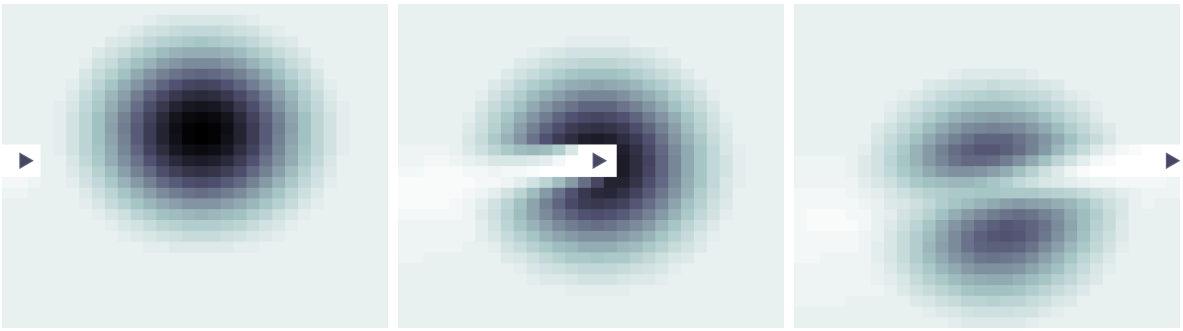


Figure 2.5: Illustrating the full Bayesian estimation process for a single agent traveling from west to east. Map dynamics are included based on a target velocity vector directed south.

2.3 Search Objective Formulation

We define the state of the team of agents at time t as p_t. Furthermore, we define the search policy μ to be a function mapping the current state of the team of agents at time t to the next set of agent positions

$$p_{t+1} = \mu(p_t) \tag{2.37}$$

In other words, the search policy μ defines the next set of controls to be executed by the searching agents. The objective is to find a search policy that minimizes the risk of targets colliding with the offshore platform. Since we wish to continuously evaluate the threat of a collision by performing new sensor measurements, it is of interest to consider

the posterior probability $P(\text{collision}|z_{1:t}, x_{1:t})$. Using the fact that we can either have a collision or no collision at all, we write the posterior as

$$P(\text{collision}|z_{1:t}, x_{1:t}) = 1 - P(\text{no collision}|z_{1:t}, x_{1:t}) \tag{2.38}$$

In order to consider the probability of a collision in the context of an occupancy grid map, we define the event

$\mathcal{C}_{i,t}$ = at least one target in grid cell i at time t causes a future collision

We extend the assumption of independent grid cell from Section 2.2.1, such that the events $\mathcal{C}_{i,t}$ are independent for all cells in the occupancy grid map. This enables us to write (2.38) for the entire search area as

$$P(\text{collision}|z_{1:t}, x_{1:t}) = 1 - \prod_i \Big(1 - P(\mathcal{C}_{i,t}|z_{1:t}, x_{1:t}))\Big) \tag{2.39}$$

By marginalization, we can write the posterior probability of $\mathcal{C}_{i,t}$ as

$$\begin{aligned} P(\mathcal{C}_{i,t}|z_{1:t}, x_{1:t}) &= \sum_{\mathbf{m}_{i,t}} P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t}) p(\mathbf{m}_{i,t}|z_{1:t}, x_{1:t}) \\ &= \sum_{\mathbf{m}_{i,t}} P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t}) bel(\mathbf{m}_{i,t}) \end{aligned} \tag{2.40}$$

Since grid cell i can not contribute to a collision should it be empty, we know

$$P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 0) = 0 \tag{2.41}$$

which enables us to write (2.40) as

$$P(\mathcal{C}_{i,t}|z_{1:t}, x_{1:t}) = P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) bel(\mathbf{m}_{i,t} = 1) \tag{2.42}$$

Inserting into (2.39) yields the following expression for the posterior probability of collision

$$P(\text{collision}|z_{1:t}, x_{1:t}) = 1 - \prod_i \Big(1 - P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) bel(\mathbf{m}_{i,t} = 1)\Big) \tag{2.43}$$

The term $P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)$ essentially states the probability that a target in grid cell i will eventually collide, given there exists at least one target in that cell. Section 2.3.1 of this thesis is dedicated solely to the estimation of this conditional probability.

As a search unfolds, the agents can iteratively update the estimate of (2.43). However, since the only action the agents are capable of performing is observation, they can do nothing more than gradually determine the chances of whether a collision is going to take place or not. In other words, our choice of search policy will in no way guarantee a reduction in the probability of collision. On the contrary, a good search policy might uncover a threat that will greatly increase $P(\text{collision}|z_{1:t}, x_{1:t})$. Thus we make the following assumption:

> *When the position of a target can be determined with a satisfactory level of certainty, the target in question no longer pose a threat.*

This assumption is reasonable in the sense that once we are relatively certain of the state of a target, this target is under control. After all it is outside the scope of the UAS to neutralize a threat. However, the decision whether or not to neutralize a threat should be based on the information *gathered* by the UAS. In the scenario of ice management, this could involve deploying boats for towing an iceberg or performing evasive maneuvers. Thus we need a measure of the importance and priority of a grid cell based on the posterior belief. We define $\delta : bel(\mathbf{m}_{i,t}) \rightarrow [0, 1]$ as a function that maps the posterior belief of a grid cell to some notion of priority. δ can take any shape depending on the desired behavior of the agents, but we found using the Shannon entropy of the belief

$$\begin{aligned} \delta(bel(\mathbf{m}_{i,t})) &= -\sum_{\mathbf{m}_{i,t}} P(bel(\mathbf{m}_{i,t})) \log_2 P(bel(\mathbf{m}_{i,t})) \\ &= -P(bel(\mathbf{m}_{i,t})) \log_2 P(bel(\mathbf{m}_{i,t})) - P(bel(\neg\mathbf{m}_{i,t})) \log_2 P(bel(\neg\mathbf{m}_{i,t})) \end{aligned} \tag{2.44}$$

yielding good results. The shape of (2.44) can be seen in Figure 2.6. In this sense, δ can be seen as a measure of the information gained by visiting each grid cell. After all, we have little interest in investigating cells with a posterior belief close to 0 or 1, since we are fairly certain of what to find. Furthermore, by modeling δ in a manner resembling

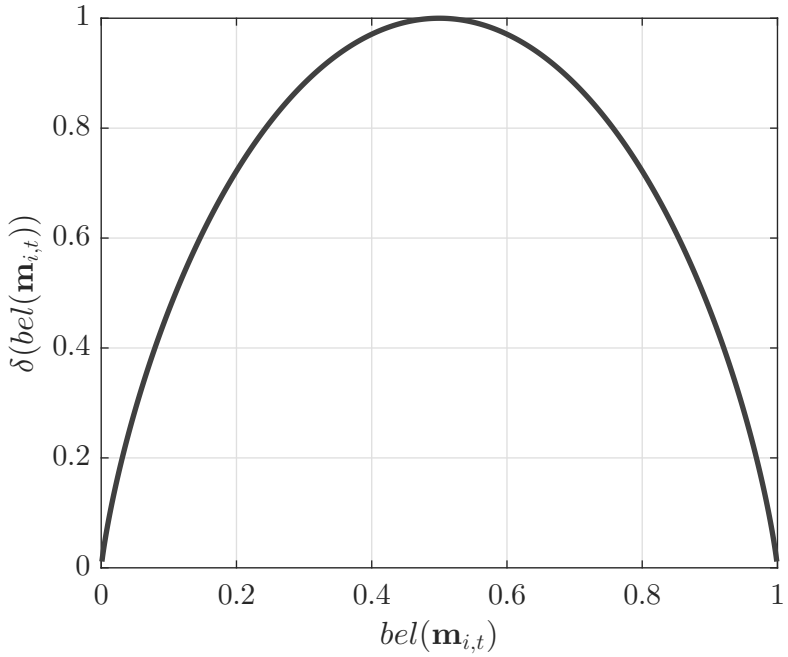


Figure 2.6: Response of $\delta(bel(\mathbf{m}_{i,t}))$ as the Shannon entropy of $bel(\mathbf{m}_{i,t})$.

(2.44), we can include a notion of target tracking. Upon detection of a target, the belief $bel(\mathbf{m}_{i,t})$ will spike causing δ to drop. As time passes, the map dynamics described in Section 2.2.2 will blur the posterior belief of the grid cell across the neighboring cells, causing a gradual increase in δ. Thus it might be of interest to go back and re-investigate the state of a target after some time.

We can now propose the following objective to the search problem:

$$\min_{\mu} f(\mu, t) := 1 - \prod_{i} \Big(1 - P(\mathcal{C}_{i,t} | \mathbf{m}_{i,t} = 1)\delta(bel(\mathbf{m}_{i,t} = 1))\Big) \tag{2.45}$$

subject to

$$p_{t+1} \in \mathcal{P}_t \tag{2.46}$$

where $\mathcal{P}_t$ is the set of all reachable points for the agents from their current state p_t. The constraint (2.46) is a result of the physical limitations of the agents. It should be apparent that the objective (2.45) closely resembles the posterior probability of collision (2.43). The difference is the replacement of the posterior belief with the priority function δ. Due to this, the objective function can be interpreted as a measure of the probability of collision, influenced only by grid cells deemed important. The measure of which grid cells are important is decided by the shape of δ.

2.3.1 Collision Risk Grid Map

In order to calculate the posterior probability of collision (2.43) and the objective function (2.45), we need to determine the conditional probability $P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)$. Since this term essentially states the probability that at least one target in a grid cell will eventually cause a collision given there are at least one target present, it is obvious this probability should be closely linked to the map dynamics. By considering a target in grid cell i, we know the only possible events occurring in a time-step is the target transitioning to a grid cell j in the neighborhood D of cell i, or the target staying put in cell i. Thus, by assuming that the individual grid cells in D contribute to the probability of $\mathcal{C}_{i,t}$ happening in an additive fashion at time $t + 1$, we can express

$$P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) = \sum_{j \in D} \sum_{\mathbf{m}_{j,t+1}} P(\mathcal{C}_{i,t}|\mathbf{m}_{j,t+1}) p(\mathbf{m}_{j,t+1}|\mathbf{m}_{i,t} = 1) \tag{2.47}$$

which is a marginalization over the states of all grid cells in D. Since we are considering the individual contributions from neighboring cells, we make the same assumption of the transitional probabilities as in (2.26), such that

$$P(\mathbf{m}_{j,t+1} = 0|\mathbf{m}_{i,t} = 1) = 0 \tag{2.48}$$

which enables us to express (2.47) as

$$P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) = \sum_{j \in D} P(\mathcal{C}_{i,t}|\mathbf{m}_{j,t+1} = 1) P(\mathbf{m}_{j,t+1} = 1|\mathbf{m}_{i,t} = 1) \tag{2.49}$$

Let us consider the term $P(\mathcal{C}_{i,t}|\mathbf{m}_{j,t+1} = 1)$. This is the probability that a target in cell i will eventually collide given there exists a target in cell j at the next time step. In the context of (2.49), this amounts to the individual contribution to the probability that a target will venture through cell j on its path to collision. Thus it arguably makes sense that we can rewrite (2.49) to

$$P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) = \sum_{j \in D} P(\mathcal{C}_{j,t+1}|\mathbf{m}_{j,t+1} = 1) P(\mathbf{m}_{j,t+1} = 1|\mathbf{m}_{i,t} = 1) \tag{2.50}$$

Right away it is possible to spot the overlapping subproblem $P(\mathcal{C}_{j,t+1}|\mathbf{m}_{j,t+1} = 1)$, which amounts to the same problem as finding $P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)$. Furthermore, (2.50) contains transitional probabilities for future time-steps. The need for predictions of future transitional probabilities would greatly increase the problem complexity, and we will thus assume our current estimate of the dynamics at time t encapsulates the future map dynamics. We can then express (2.50) as

$$P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) = \sum_{j \in D} P(\mathcal{C}_{j,t}|\mathbf{m}_{j,t} = 1) P(\mathbf{m}_{j,t} = 1|\mathbf{m}_{i,t-1} = 1) \tag{2.51}$$

One might argue this is a crude assumption, since a worst case scenario might contain a considerable change in target dynamics should the weather and current conditions change. However, the problem can be addressed by adding sufficient uncertainty to the target vector described by $\mathcal{K}_t$. Next, we subtract both sides of (2.51) by the case where $j = i$, such that

$$\begin{aligned} &P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)\Big(1 - P(\mathbf{m}_{i,t} = 1|\mathbf{m}_{i,t-1} = 1)\Big) \\ &= \sum_{j \in D \setminus \{j=i\}} P(\mathcal{C}_{j,t}|\mathbf{m}_{j,t} = 1)P(\mathbf{m}_{j,t} = 1|\mathbf{m}_{i,t-1} = 1) \end{aligned} \tag{2.52}$$

which yields the following expression

$$P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1) = \frac{\sum_{j \in D \setminus \{j=i\}} P(\mathcal{C}_{j,t}|\mathbf{m}_{j,t} = 1)P(\mathbf{m}_{j,t} = 1|\mathbf{m}_{i,t-1} = 1)}{1 - P(\mathbf{m}_{i,t} = 1|\mathbf{m}_{i,t-1} = 1)} \tag{2.53}$$

By looking at (2.53), we see that the probability of collision from cell i is again given by the probability of collision from its neighbors j. However, the neighbors j are in turn given by the value of cell i. In other words, (2.53) can not be solved explicitly. Thus we will now propose an iterative procedure to approximate $P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)$.

Let C^k denote the grid map of the approximated grid cell probabilities of collision after iteration k

$$C^k = \{\mathbf{c}_i^k = \hat{P}(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)\} \tag{2.54}$$

Let Q denote the subset of grid cell indices in the occupancy grid map containing the offshore platform. Thus a collision is defined as the event of a target transitioning into a grid cell $\mathbf{m}_{i,t}$ for all $i \in Q$. Initiate C^0 such that

$$C^0 = \{\mathbf{c}_i^0 = 0 \mid i \notin Q\} \cup \{\mathbf{c}_i^0 = 1 \mid i \in Q\} \tag{2.55}$$

which essentially is a binary map with cell value 1 for cells defined as containing an offshore platform. After all, should a target happen to be in such a cell, there is a 100% chance of collision. For every iteration k until convergence, all grid cells $i \notin Q$ in C^k are updated according to

$$\mathbf{c}_i^k = \frac{1}{1 - P(\mathbf{m}_{i,t} = 1|\mathbf{m}_{i,t-1} = 1)} \sum_{j \in D \setminus \{j=i\}} \mathbf{c}_j^{k-1} P(\mathbf{m}_{j,t} = 1|\mathbf{m}_{i,t-1} = 1) \tag{2.56}$$

where the transitional probabilities $P(\mathbf{m}_{j,t} = 1|\mathbf{m}_{i,t-1} = 1)$ are the elements of $\mathcal{K}_t$. C^k is said to converge when

$$|\mathbf{c}_i^k - \mathbf{c}_i^{k-1}| < \theta, \quad \forall i \tag{2.57}$$

where θ is a convergence threshold specified by the desired accuracy of the iterative estimation. The full procedure is summarized in in Algorithm 2.

Algorithm 2 Collision Risk Grid Map Creation Algorithm

procedure CREATECOLLISIONGRIDMAP($\mathcal{K}_t$, Q)
  Initiate C^0 according to (2.55)
  $k := 1$
  while C^{k-1} has not converged **do**
    for all grid cells $i \notin Q$ **do**
      Calculate $\mathbf{c}_i^k$ according to (2.56)
    end for
    $k := k + 1$
  end while
end procedure

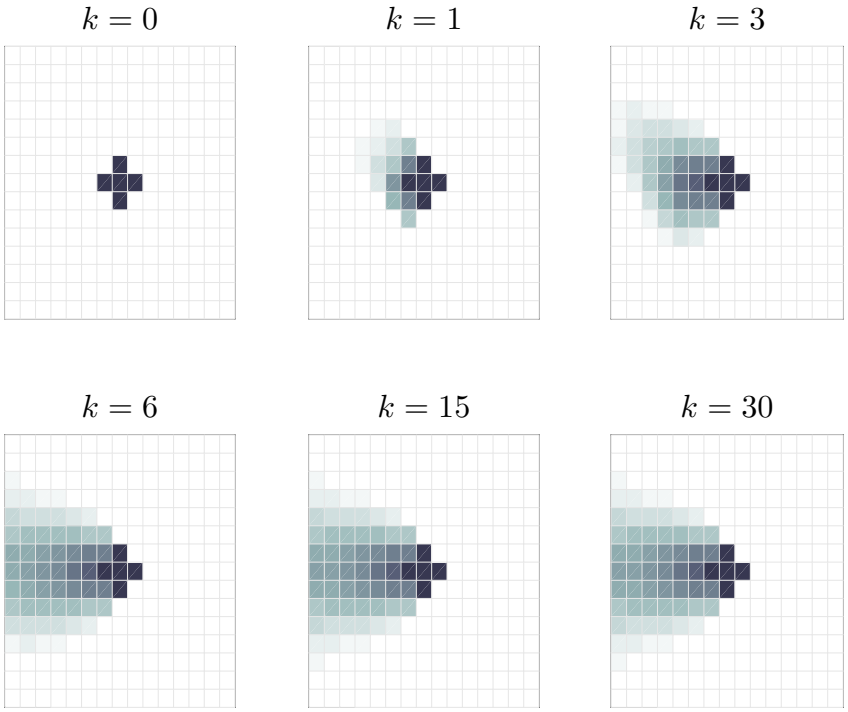


Figure 2.7: Illustration of the iterative process of creating the grid map C^k containing the probabilities of collision.

Although a formal proof of convergence for C^k has not been established, Algorithm 2 has been tested on a broad range of transition kernels $\mathcal{K}_t$, resulting in convergence every time. Figure 2.7 illustrates a simple example of the iterative procedure listed in Algorithm 2. The dark grid cells at iteration $k = 0$ marks the cells defined as critical in terms of collision. It can be seen that for this simple example, the algorithm converges after relatively few iterations.

2.4 Model Predictive Search

2.4.1 Search Policy

The objective function $f(\mu, t)$ proposed in Section 2.3 presents a metric for optimization, but the choosing of agent control actions and trajectories is not specified. During a search each agent is given a path or a trajectory to follow, and the overall goal is to find the set of paths for the team of agents such that $f(\mu, t)$ is minimized as quickly as possible. To find the set of optimal routes for the team to follow, the objective function will be further examined. We define the associated risk $\mathbf{r}_{i,t}$ for a cell i at time t to be

$$\mathbf{r}_{i,t} := P(\mathcal{C}_{i,t}|\mathbf{m}_{i,t} = 1)\delta(bel(\mathbf{m}_{i,t} = 1)) \tag{2.58}$$

such that the search objective can be rewritten as

$$\min_{\mu} \left[1 - \prod_{i}(1 - \mathbf{r}_{i,t})\right] \tag{2.59}$$

The agents are able to alter the objective function by taking a measurement of cell i, thus updating the belief $bel(\mathbf{m}_{i,t})$ associated with the cell. Due to potential measurement noise and the possible events of false negatives, we let $\hat{\mathbf{r}}_{i,t}$ denote the associated risk of cell i after the belief has been updated according to Algorithm (1).

In optimization theory, there are a number of ways of finding a local minimum. In this thesis we employ the *steepest descent method*, which will be looking for paths resulting in the most negative gradient of the objective function. In other words, to minimize $f(\mu, t)$ the team of agents should use the set of paths that, after execution, maximize the difference in the objective function per unit time. Let $P_{i,t}$ be the path associated to agent i and $\mathcal{P}_t$ denote the set of planned paths for the team of agents at time t, i.e.

$$\mathcal{P}_t = \{P_{1,t}, P_{2,t}, \ldots, P_{n,t}\} \tag{2.60}$$

Let $T(\mathcal{P})$ denote the time for the team of agents to traverse their paths in $\mathcal{P}$. The reward $R_T(\mathcal{P})$ for the team is the reduction of the objective function per unit time accomplished by the entire team if the agents follow their respective path in $\mathcal{P}$. As a large decrease in the objective function should result in a large positive team reward, the team reward should be defined as the negative difference in the objective function per unit time. As the team of agents have performed measurements of a set of cells when all paths are traversed, the posterior objective function will be divided into two sets, one consisting of the cells that are not updated, $i \notin \mathcal{P}$, and the other consisting of cells that are updated by the measurements, $i \in \mathcal{P}$. This leads to the following expression for the team reward

$$R_T(\mathcal{P}) := -\frac{\Delta f}{T(\mathcal{P})} = -\frac{f_{\text{post}} - f_{\text{pre}}}{T(\mathcal{P})} \tag{2.61}$$

$$= -\frac{1}{T(\mathcal{P})}\left[1 - \prod_{i\notin\mathcal{P}}(1 - \mathbf{r}_{i,t})\prod_{i\in\mathcal{P}}(1 - \hat{\mathbf{r}}_{i,t})\right] + \frac{1}{T(\mathcal{P})}\left[1 - \prod_{i}(1 - \mathbf{r}_{i,t})\right] \tag{2.62}$$

where f_{pre} is the value of the objective function before and f_{post} after the paths in $\mathcal{P}$ have been executed.

In order to find the optimal set of paths for the team, a few assumptions are made. Firstly, the objective function value f_{pre} will not be affected by the choice of paths $\mathcal{P}$ and

can be considered constant for the optimization problem. Secondly, the knowledge of measurement noise and false negatives and positives are not considered during the path planning, as we assume the agents will gather sufficient measurements of any cell i to let $\hat{\mathbf{r}}_{i,t} \to 0$. From these assumptions, the team reward can be rewritten as

$$R_T(\mathcal{P}) = \frac{1}{T(\mathcal{P})}\left[\prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t}) - \prod_{i}(1-\mathbf{r}_{i,t})\right] \tag{2.63}$$

$$= \frac{1}{T(\mathcal{P})}\left[\prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t}) - \prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t})\prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t})\right] \tag{2.64}$$

$$= \frac{1}{T(\mathcal{P})}\prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t})\left[1 - \prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t})\right] \tag{2.65}$$

Next, to find the optimal set of paths, we are looking for the set of paths $\mathcal{P}^*$ that will maximize the team reward

$$\mathcal{P}^* := \operatorname*{argmax}_{\mathcal{P}} R_T(\mathcal{P}) \tag{2.66}$$

$$= \operatorname*{argmax}_{\mathcal{P}} \frac{1}{T(\mathcal{P})}\prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t})\left[1 - \prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t})\right] \tag{2.67}$$

$$\tag{2.68}$$

Since the objective function value f_{pre} is unaffected by the choice of $\mathcal{P}$, we know that the product

$$f_{\text{pre}} := \prod_{i}(1-\mathbf{r}_{i,t}) = \prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t})\prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t}) \tag{2.69}$$

is constant. This implies that maximizing one of the product terms in (2.69) is equal to minimizing the other. Due to this fact, equation (2.67) can be rewritten as

$$\mathcal{P}^* = \operatorname*{argmax}_{\mathcal{P}} \frac{1}{T(\mathcal{P})}\prod_{i\notin\mathcal{P}}(1-\mathbf{r}_{i,t}) \tag{2.70}$$

By using the fact proposed in (2.69), we can express $\mathcal{P}^*$ as

$$\mathcal{P}^* = \operatorname*{argmax}_{\mathcal{P}} \frac{f_{\text{pre}}}{T(\mathcal{P})\prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t})} \tag{2.71}$$

$$= \operatorname*{argmin}_{\mathcal{P}} T(\mathcal{P})\prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t}) \tag{2.72}$$

As $(1-\mathbf{r}_{i,t})$ will take a value in the interval $[0,1]$, the product can potentially get very small should many cells be included in $\mathcal{P}$. A more convenient approach is thus minimizing the logarithm of the objective function.

$$\mathcal{P}^* = \operatorname*{argmin}_{\mathcal{P}} \log\left[T(\mathcal{P})\prod_{i\in\mathcal{P}}(1-\mathbf{r}_{i,t})\right] \tag{2.73}$$

$$= \operatorname*{argmin}_{\mathcal{P}}\left[\log T(\mathcal{P}) + \sum_{i\in\mathcal{P}}\log(1-\mathbf{r}_{i,t})\right] \tag{2.74}$$

Finally, the team reward value should be positive for any set of paths $\mathcal{P}$ if it decreases the objective function. Hence the optimal set of paths can be expressed as the solution to the following optimization problem

$$\mathcal{P}^* = \underset{\mathcal{P}}{\operatorname{argmax}} \left[-\log T(\mathcal{P}) - \sum_{i \in \mathcal{P}} \log(1 - \mathbf{r}_{i,t}) \right] \tag{2.75}$$

2.4.2 Motivation for the Graph Construction

The notions of a path and a set of paths were introduced in section 2.4.1, but no information was given about how a path is represented, nor how the search for optimal paths is performed. A path is defined as a set of sequential *waypoints*, where a waypoint is a spatial position in the search area.

$$P := \{w_1, w_2, \dots w_n\} \tag{2.76}$$

It should be noted that the path planning in this thesis is simplified to the two-dimensional plane, as the the desired altitude can be predetermined and constant for the searching agents. Thus, a waypoint w_i is represented by the two-dimensional coordinates (x_i, y_i).

As the spatial positions in the search area are continuous, some sort of discretization of the potential waypoints is required to have a finite number of paths to consider. A solution proposed in [9] is to span the search area by a lattice graph $G_l := (V_l, E_l)$ with vertex set V_l and edge set E_l. Each node v in V_l is assigned to a position (x_m, y_n) in the search map (Figure 2.9a), and represents a possible waypoint for the agents to consider as part of their paths. Similarly, the edges represent potential connections between the waypoints. To ensure that it is physically possible for an agent to travel directly from a node i to any connected neighbor node j, the *node spacing* must be large enough to satisfy the aircraft constraints described in Section 2.1. Moreover, a vital part is that the node spacing ensures that *every cell* in the occupancy grid map will be covered by the FOV of an agent from at least one node in the graph, to prevent the case where a region is not observable from any waypoint. It is important to note that any lattice graph will work for this process, and [9] found a degree-3 hexagonal lattice to give particularly good results. However, the principles of the algorithm can be be tested regardless of lattice choice, and we will therefore use the simplistic square grid graph in this thesis.

To be able to use the graph to find an optimal set of trajectories, a *search map* $s_t = \{\mathbf{s}_{i,t}\}$ will be introduced. The search map is a grid map, where each cell $\mathbf{s}_{i,t}$ is represented by the value

$$\mathbf{s}_{i,t} = -\log(1 - \mathbf{r}_{i,t}) \tag{2.77}$$

The graph G_l is placed on top of the search map s_t, and each node is assigned a weight equal to the sum of the cells $\mathbf{s}_{i,t}$ in the neighborhood of the node. The sum of the cells in the neighborhood of a node is found by dividing the entire search area into equally sized regions, where each node has its position at the center of a such region. This region is defined as the *field of region* (FOR) for the correspondingly centered node as shown in Figure 2.8. The size of the FOR is thus dependent on the spacing between the nodes. The weight $W_t(v)$ for a node v is computed as the sum of cells in the search map laying

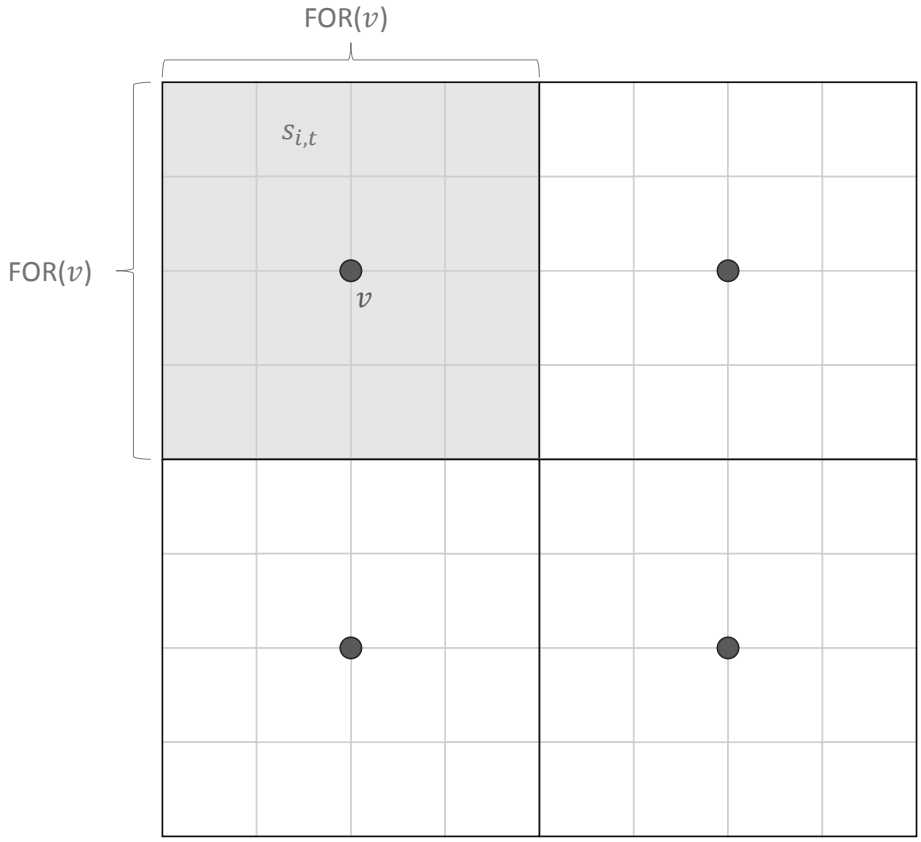


Figure 2.8: Illustration of how the FOR of a node is defined.

inside the FOR of v at time instance t

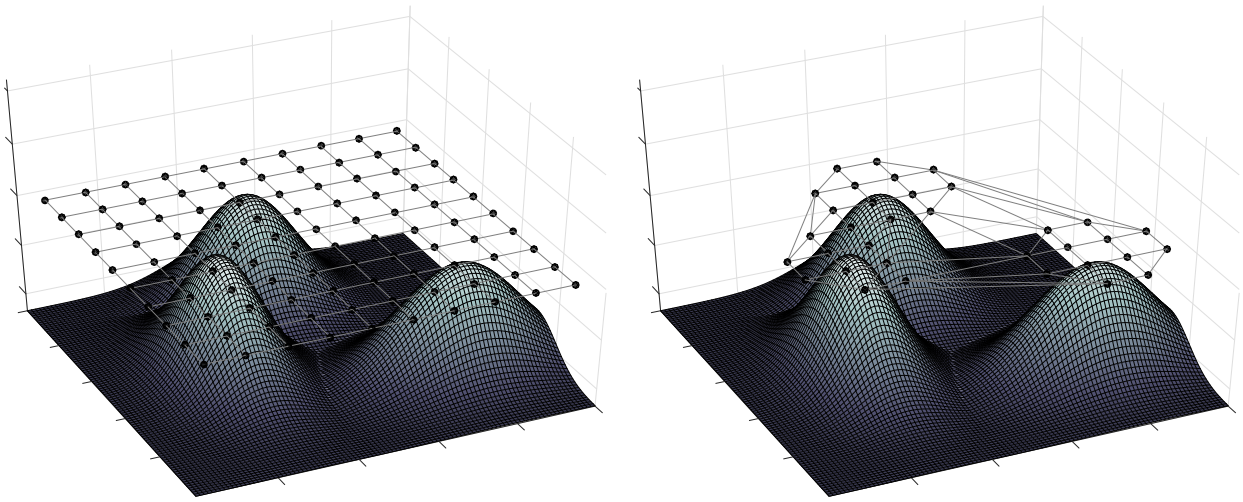
$$W_t(v) := \sum_{i=1}^{n} \mathbf{s}_{i,t} \cdot \text{inFOR}(v, \mathbf{s}_{i,t}) \tag{2.78}$$

$$\text{inFOR}(v, \mathbf{s}_{i,t}) := \begin{cases} 1, & \mathbf{s}_{i,t} \in \text{FOR}(v) \\ 0, & \text{otherwise} \end{cases} \tag{2.79}$$

Note how equations (2.78) and (2.77) are related to (2.75). The weight of node v is closely related to the reduction of the objective function yielded by including v in a path. Since there can potentially be a very large number of nodes in the graph, a horizon length l will be set to limit the maximum number of nodes each path can contain at all times.

It is not necessary to consider every node during the search as some nodes may have a low and negligible weight $W_t(v)$. Since we want to search in areas with high search map cell values, we introduce a threshold τ to remove the nodes with weights below this threshold. After the removal, the remaining nodes will be reconnected by using Delaunay triangulation (DT) to ensure connectivity between possibly separated areas. Figure 2.9b shows an illustration of what a fully constructed graph G could look like. Full details for how the graph is constructed is explained in Section 2.4.3.

The weight assigned to a node can be thought of as a search reward the agents will receive by examining this node. The optimal set of trajectories are found by searching for the paths that results in the highest search reward for the entire team. This algorithm is called *Cooperative Graph-based Model Predictive Search* (CGBMPS) [9] and is described in detail in Section 2.4.4.



(a) Lattice graph over the search map s_t. (b) Fully created graph over the search map s_t.

Figure 2.9: Lattice- and fully created graph placed over the search map s_t.

2.4.3 Graph Construction

The dynamic graph is created as proposed in [9]. Since the search map is constantly changing due to sensor measurements from the team of agents, the graph should be periodically updated to ensure the trajectories only consists of waypoints with high search reward. The graph construction process is stated in Algorithm 3. Figure 2.10 shows the graph construction step-by-step.

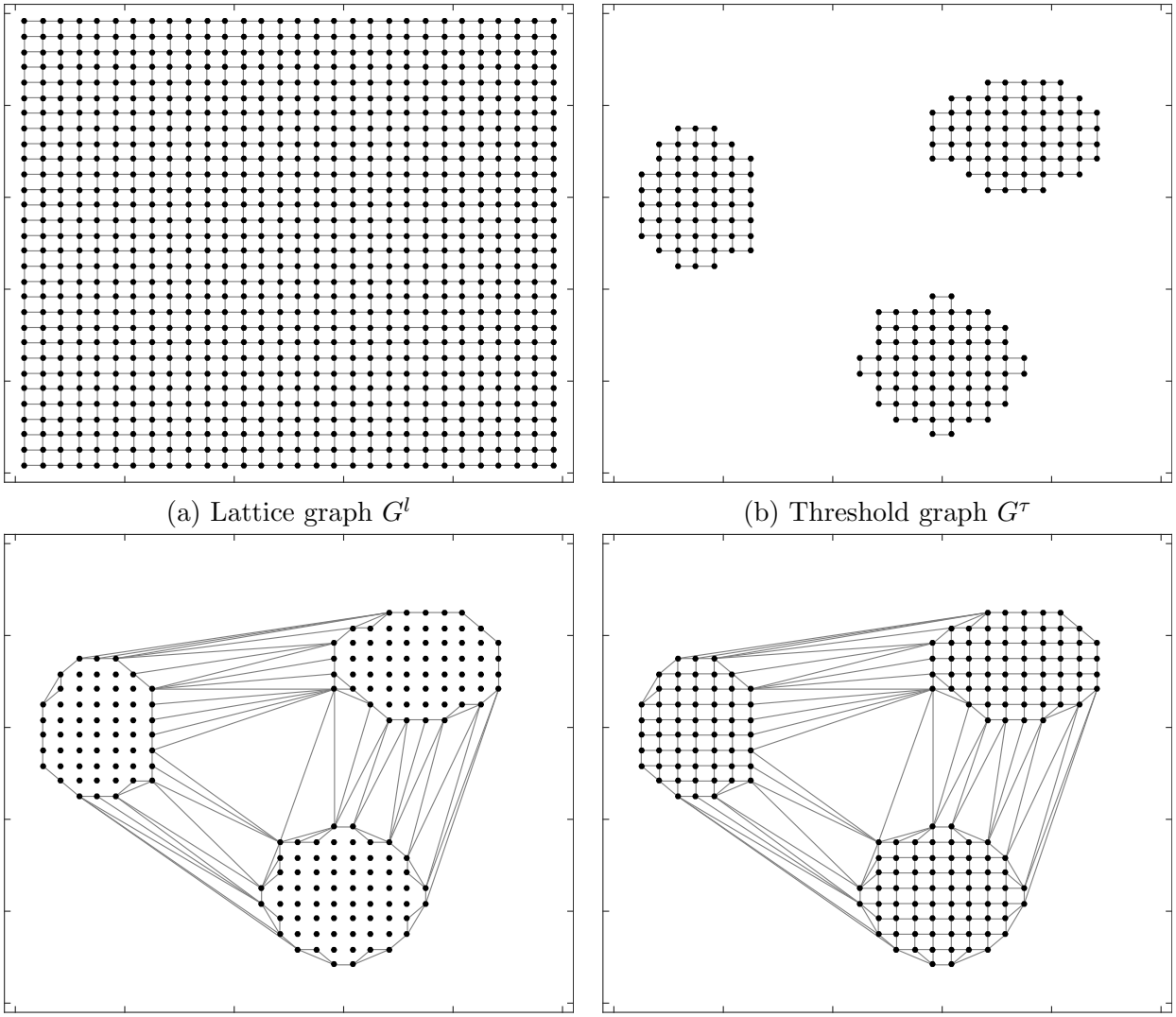
Algorithm 3 Graph Construction at time instance t

1: Construct uniform lattice graph $G^l := (V^l, E^l)$ over the search region
2: Set reward threshold $\tau \geq 0$
3: Let $G_t^\tau := (V_t^\tau, E_t^\tau)$ be the graph where nodes from G^l with values less than τ at time instance t are removed, $V_t^\tau := \{v \in V^l \mid W_t(v) \geq \tau\}$. E_t^τ is the set of edges connecting the nodes in V_t^τ
4: Let $G_t^{Del} := (V_t^\tau, E_t^{Del})$ be the graph generated by a Delaunay triangulation of V_t^τ
5: Let $G_t^{Del<d} := (V_t^\tau, E_t^{Del<d})$ be the subgraph of G_t^{Del} obtained by keeping only the edges in E_t^{Del} connecting the nodes in V_t^τ which has degree less than d in G_t^τ, where d is the degree of the lattice graph G^l.
6: Let the final graph $G_t := (V_t, E_t)$ be the combination of G_t^τ and $G_t^{Del<d}$, that is $V_t := V_t^\tau$ and $E_t := E_t^\tau \cup E_t^{Del<d}$

2.4.4 Cooperative Graph-Based Model Predictive Search

As the dynamic search graph based on the search map s_t is created, an algorithm for the team of agents to cooperatively find a set of paths to follow has to be established. The algorithm presented in this section is a slight generalization of the CGBMPS algorithm presented in [9] to include the use of the search priority map s_t.

Suppose there is a team of M agents searching for an unknown number of targets in a search region. $G_t := (V_t, E_t)$ is the dynamic graph used for path-planning with a vertex set V_t and edge set E_t at time instance t. As already stated, the vertices of G_t serves



(a) Lattice graph G^l

(b) Threshold graph G^τ

(c) Threshold Delaunay graph $G^{Del<d}$

(d) Final graph G

Figure 2.10: Step-by-step illustration of the graph construction. The search map s_t consists of three Gaussian distributions and the nodes with high weights remains after the threshold in (b).

as potential waypoints for the agents. At all times, each individual agent a has a set of waypoints they want to reach, which is a path P_a represented as a sequence of vertices in G_t; $P_a := (v_1^a, v_2^a, \ldots, v_{l-1}^a, v_l^a)$, where l is the horizon length of the receding horizon optimization.

A path traveling cost is defined as

$$T(P_a) := \sum_{i=1}^{l-1} t(v_i^a, v_{i+1}^a) \tag{2.80}$$

where $t(v_i^a, v_{i+1}^a)$ is the time it takes for agent a to travel between node v_i and node v_{i+1}. $T(P_a)$ is thus equal to the total duration for agent a to travel along path P_a. Optimally, each agent should select a path P_a^* that maximizes the *team's* examination of prioritized areas. Therefore it is convenient to find an expression for the team reward R_T, which all of the agents attempt to maximize. From (2.75) and (2.77) we rewrite the team reward

as

$$R_T(\mathcal{P}) = -\log T(\mathcal{P}) + \sum_{i \in \mathcal{P}} \mathbf{s}_{i,t} \tag{2.81}$$

Since we are summing over the set of cells visited by the entire team, we can also sum over set of cells visited by each agent and sum over all agents. The time penalty $T(\mathcal{P})$ is the time it takes the team to finish executing the set of paths P, and could thus be equal to $T(\mathcal{P}) = \max[T(P_a)]\ \forall P_a \in \mathcal{P}$. This definition would make the search party only try to decrease the duration of the most time consuming path, and remove the information about the time spent for the other paths. Instead, our steepest descent search is interpreted as if there were only one agent trying to find and execute M potentially non-connected paths, where the overall time would be the sum of the time it takes to traverse each path. This will make the team work as a unit to optimize the time spent by every agent, and we can thus express the time penalty as

$$T(\mathcal{P}) := \sum_{a} T(P_a) \tag{2.82}$$

enabling us to rewrite the team reward as

$$R_T(\mathcal{P}) = -\log \sum_{a=1}^{M} T(P_a) + \sum_{a=1}^{M} \sum_{i \in P_a} \mathbf{s}_{i,t} \tag{2.83}$$

As the agents are using the nodes in V_t as waypoints, and the FOR for two neighbor nodes border each other, the sum of all cells $\mathbf{s}_{i,t}$ included in path P_a can be approximated by summing all the node weights of all nodes v_i^a in P_a when assuming that the FOV of each agent is approximately as large as the FOR. Thus, we can simplify the team reward to

$$R_T(\mathcal{P}) = -\log \sum_{a=1}^{M} T(P_a) + \sum_{a=1}^{M} \sum_{i=1}^{l} W_t(v_i^a) \tag{2.84}$$

where W_k is calculated by (2.78)

The agents will compute a new path every time they reach a new waypoint. Hence it will only be the *first* node in the path that will be reached before a new path is generated. This receding horizon approach is similar to the principals of model predictive control (MPC) and is referred as model predictive search (MPS).

Obviously, the agents will generally arrive at waypoints at different times since the edges in E_t may have nonuniform lengths. At each time instant we therefore define the set A_{plan} containing all agents that have arrived at a waypoint and A_{en} containing all agents en route to a waypoint. In the CGBMPS algorithm, whenever the set A_{plan} is nonempty, the agents in that set need to compute new paths assuming that the agents in A_{en} will continue on their current paths. The update of the optimal paths at timestep k is determined by

$$\{P_a^* : a \in A_{\text{plan}}(k)\} = \underset{\{P_a : a \in A_{\text{plan}}(k)\}}{\operatorname{argmax}} R_T(\{P_a\}) \tag{2.85}$$

Most often A_{plan} will consist of only a single agent. However, when A_{plan} consists of more than one agent, equation (2.85) is not possible to solve explicitly as the optimal

paths are found by assuming all the other optimal paths are known in (2.84) . The worst case scenario is all agents arriving at a waypoint at the same time instant. This will make the optimization problem computationally heavy so an approximation used in [9] is the sequential optimization

$$P_a^* := \underset{P_a}{\operatorname{argmax}}\ R_T(P_1^*, P_2^*, \ldots, P_a),\ \forall a \in A_{\text{plan}} \tag{2.86}$$

Since all agents in A_{en} already have computed their optimal path, that is $\exists P_i^*\ \forall i \in A_{\text{en}}$, this modification results in the agent with the lowest index in A_{plan} to plan its path first and the remaining agents plan their paths accordingly in a sequential manner. However, this approximation does not care about cases where it might be better that the indexes are shuffled, such that agent n finds its path before agent $n-1$. This may result in a suboptimal behavior by the team, and is solved by calculating (2.86) for all permutations of agent orderings and choose the set of paths that yields the highest team reward.

To prevent multiple agents visiting the same node, the reward of visiting an already taken node is set to zero for the subsequent agents. However, if there are waypoints with high priorities close to an already taken waypoint, then the subsequent agents can choose to pass through this node to reach the high priority node if this is an optimal solution. The complete algorithm can be seen listed in Algorithm 4.

Algorithm 4 CGBMPS

$t := 0$
Construct $G_0 = (V_0, E_0)$ as in Section 2.4.3
For each agent $a \in \{1, \ldots, M\}$ assign initial path $P_a^* = (v_1^a, v_2^a, \ldots, v_l^a)$ on G_0
while search not complete **do**
    Each agent takes sensor measurement
    Update the belief $bel(m_t)$ as in Section 2.2.3.
    Compute the set of agents arriving at a waypoint, A_{plan}
    if the set A_{plan} is nonempty **then**
        Update G_t as in Section 2.4.3
        for each agent a in the set $A_{\text{plan}}(k)$ **do**
            Compute the path P_a^* using (2.86) with permutations
        end for
    end if
    Agents move towards next waypoint in path
    $t = t + 1$
end while

Chapter 3

System Architecture

This chapter describes the system architecture of the simulator implemented to test the performance of the system outlined in Chapter 2. The majority of work done in this thesis have been spent implementing the simulation software, and it is thus natural to give a brief description of its design and capabilities. Even though the system is intended to have a high degree of autonomy capable of beyond line of sight (BLOS) operations, the online optimization is performed in a centralized manner. This means the agents will have on-board computers processing the sensor measurements before sending them to a ground station which makes all decisions on behalf of the team. This design implies the need for a telemetry link providing robust communication between the centralized computer and all agents, leading to a requirement of the on-board computers being able to perform image processing on a substantial amount of raw sensor data, rather than send the data for processing elsewhere. Thus the only image information required to transmit to the centralized computer would be binary detection variables for the occupancy grid cells, which will greatly increase the robustness of the communication links in terms of less data transfer.

The implementation of the system in this thesis is mainly twofold: The software running on the centralized computer calculating the optimal agent trajectories, and a software in the loop (SITL) setup simulating the aircraft dynamics in real time using actual communication links to the centralized computer. Both aspects will be given a brief description in the following sections.

3.1 Centralized Model Predictive Search Software

In essence, the task of the centralized computer is to calculate optimal trajectories for the team of agents based on their poses and sensor measurements of the search area. As described in Section 2.4.2, the problem of calculating trajectories is reduced to the problem of finding a set of optimal waypoints for each agent, as the on-board autopilot will calculate a trajectory accordingly.

MATLAB was chosen as the programming language to implement the model predictive search optimization. This is due to the high amount of built-in functions suited for scientific development and MATLAB's high level of abstraction and possibility for fast prototyping. The program is written using object-oriented programming (OOP), abstracting several core components into objects. Due to the object instances separating the functionality of the program we will refer to the separate objects as modules. In

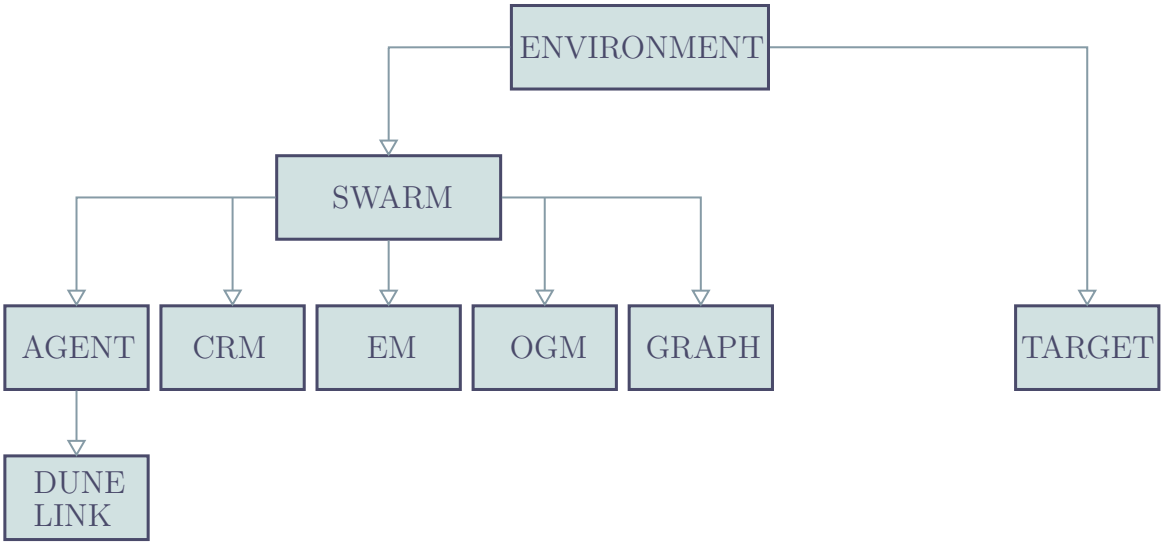


Figure 3.1: Dependencies between the MATLAB modules.

some cases certain variables and methods of a module is accessible by other modules. The program is divided into the following modules:

- Agent
- Collision risk map
- DUNE link
- Entropy map
- Environment
- Graph
- Occupancy grid map
- Swarm
- Target

The modules all have separate responsibilities. For instance, while the graph module is responsible for updating the nodes and edges of the waypoint graph at each time step, the swarm module is responsible for keeping track of the positions and planned paths of all agents. Illustrated in Figure 3.1 are the module dependencies. All the modules are indirectly connected to the environment module, which act as a container for the overall simulation. The swarm module also acts as a container in the sense that it holds a collision risk map-, entropy map-, occupancy grid map- and a graph instance in addition to a list of agent instances. The tree structure ensures minimal dependencies between the modules, which in turn optimizes both the abstraction of the architecture and the code quality.

3.1.1 Module Descriptions

Map Modules

There are three separate map modules; Collision risk-, occupancy grid- and entropy map module. The occupancy grid map (OGM) module is responsible of keeping track of the posterior belief of the occupancy grid map, as well as the creation of an initial prior belief map from recent satellite data. The posterior belief is updated every time a measurement is supplied by an agent. The occupancy grid map module is also responsible for calculating the next iteration in the Bayesian filter presented in Section 2.2.3.

The collision risk map (CRM) module creates and stores the collision risk grid map presented in Section 2.3.1. The collision risk grid map is not updated during the search except if (or when) the kernel is changed.

The entropy map (EM) module is very similar to the occupancy grid map module, but contains the Shannon entropy of all the belief cells instead of probability. Every time the occupancy grid map is updated, the entropy map module will recalculate the Shannon entropy of each updated belief cell.

Table 3.1 shows the three map modules and some of their methods

Table 3.1: The map modules and some of their methods.

Module	**Methods**
`occupancy_grid_map`	`bayes_estimation(kernel, map)` `predict_belief(kernel, map)`
`collision_risk_map`	`calc_risk_map_from_kernel(kernel)`
`entropy_map`	`calc_entropy_from_OGM(OGM)`

Graph Module

The graph module creates and dynamically updates the graph containing the possible paths to consider for the agents. The graph module uses the element wise product between the entropy- and collision risk grid map to calculate the graph as explained in Section 2.4.3. Table 3.2 shows some of the methods and variables of the module. Every time the planned paths of the agents are to be updated, a new waypoint graph is

Table 3.2: The variables and methods belonging to the graph module.

Variables	**Methods**
`number_of_nodes`	`add_node()`
`node_weights`	`remove_node()`
`reward_threshold`	`create_graph_from_map(map)`
`adjacency_matrix`	

created by calling `create_graph_from_map` to account for recent measurements. The variable `adjacency_matrix` keeps track of the current waypoint graph, represented as an adjacency matrix.

Agent Module

The agent module stores information of one single agent, and every agent in the team is represented by an instance of this module. The module has no methods and Table 3.3 shows the key variables. The position and desired velocity of each agent will be stored within their respective instance of this module. In addition, every agent holds an instance of the DUNE link module. This module keeps all information necessary for the central computer to communicate with the agent in question, regardless of the type of communication link used. Details about the software, DUNE, used as a command and communications link, will be given in Section 3.2.

Table 3.3: Variables in the agent module.

Variables	Methods
ID	
pose	
velocity	
current_node	
FOV_size	
current_path	
dune_link	

Target Module

The target module keeps track of all information of one single target, and thus every simulated target is represented as an instance of this module. Note that the swarm module does not have access to the target module (Figure 3.1), and thus the true target positions will not influence the estimated target positions. The target module does *not* contain information about the team's posterior belief of the target positions, as that is the purpose of the occupancy grid map module. The target module is fairly simple, and its variables and methods are listed in Table 3.4.

Table 3.4: The variables and methods belonging to the target module.

Variables	Methods
ID	move()
position	
velocity_vector_mu	
velocity_vector_sigma	
is_detected	

If a target is positioned within the field of view of an agent while performing a measurement, is_detected is set to true. By considering this variable for all targets in a search area we get a simple metric for the efficiency and thoroughness of a search party.

Swarm Module

The swarm module can be seen as the core component of the centralized program. This module keeps track of all information provided by the searching agents and makes all decisions regarding path choices for the agents. Instances of the waypoint graph, all maps and a list of active agents are stored, and this is the module performing the model predictive search optimization. The variables and methods of the swarm module are listed in Table 3.5. At every time instance, the optimal paths for the team can be found by calling update_agent_paths, which in turn calls model_predictive_search, solving (2.86) by using dynamic programming. For each individual agent, the optimal path is found by calling find_optimal_path recursively on the neighboring nodes, since the optimal path from the current node with horizon l is the same as the maximum of the optimal paths from the neighboring nodes with horizon $l-1$. The method uses depth first search to determine the order of nodes to examine.

Table 3.5: The variables and methods belonging to the swarm module.

Variables	Methods
agents	model_predictive_search(graph)
OGM	update_graph(graph)
CRM	take_step()
EM	update_agent_paths()
graph	measure_and_update_map(map)
kernel	calc_path_reward()
horizon_length	find_optimal_path(graph)
position_log	measure()
search_area_size	

Environment Module

The environment module acts as a facilitator search scenario. It contains an instance of the swarm module and a list of all target module instances in the search region. This abstraction enables us to draw a clear line between the estimated reality, contained in the swarm module, and actual reality, contained in the list of target module instances. The role of the environment module is thus to keep track of the time passed, move the targets in accordance with their true velocity vectors and return the simulated camera measurements to the swarm module. The environment module will flag targets as detected if they are, and it is also capable of visualizing the search as it unfolds, plotting the poses of all the agents along with the positions of the targets. Table 3.6 shows the variables and methods of the environment module.

Table 3.6: Variable and methods of the simulation module.

Variables	Methods
swarm	add_targets(targets)
targets	mark_detected_targets(targets)
time_passed	visualize_search()
	execute_search()
	get_agent_positions()

Listing 3.1 shows a code snippet from the `execute_search` method, which is run after all initializations are complete. `self` is a reference to the environment instance, as the function belongs to this module. In line 2, `measure_and_update_map` calls the swarm method `measure` for all agents, subsequently calling `bayes_estimation` from the occupancy grid map module to update the occupancy grid- and entropy map. `get_agent_positions` reads the position data received from DUNE and updates the `position` variable in the agent instances. In line 5, `update_agent_paths` determines which agents have arrived at their designated waypoint and add them to A_{plan}. If A_{plan} is nonempty, a new graph is created, and the `model_predictive_search` method is called on this graph to update the optimal paths.

```
1 while search_not_complete
2     located_targets = self.swarm.measure_and_update_map();
3     self = self.mark_detected_targets(located_targets);
4     self = self.get_agent_positions();
5     self.swarm = self.swarm.update_agent_paths();
6     self.swarm.send_waypoints_to_agents();
7     self.visualize_search();
8 end
```

Listing 3.1: Snippet from the `execute_search` method belonging to the environment module.

Visualization Tool

A visualization tool has been created to enable the user to view the current status of the search as it unfolds. Figure 3.2 shows a snapshot of the visualization tool. The visualization tool consists of multiple windows displaying information vital to the search. The main window is showing the search priority map s_t with the dynamically created waypoint graph on top. The agents are represented as triangles and their position logs are displayed as trailing lines. The upper right window displays the posterior belief of the occupancy grid map, based on the measurement gathered by all the agents. The middle window to the right shows the collision risk map created based on the target dynamic kernel, while the lower right window shows the kernel. All plots in the visualization tool are made three dimensional to make it possible to view the search from any suitable angle.

3.1.2 Features and Capabilities

There are a few parameters left for the user to specify when initializing the centralized model predictive search. Among those, the most important are

- Geographical boundaries of the search area
- Resolution of the occupancy grid map
- Position of the offshore installation in the search area
- Horizon length in each step of the model predictive search
- Inverse sensor measurement model return values

It is however important to note that choosing a higher grid map resolution or horizon length increase the system requirements of the computer used to run the program. Should the centralized computer spend too much time on each calculation, the consequence would be agents waiting for further instructions would be left loitering their previously assigned waypoint in the meantime, which obviously is a waste of time.

The modular design allows us to keep the true target positions stored in the environment module, separated from the actual optimization process. During simulation, this means when an agent reports its pose back to the ground station, a field of view will be calculated from the target topology creating a simulated measurement. However, this

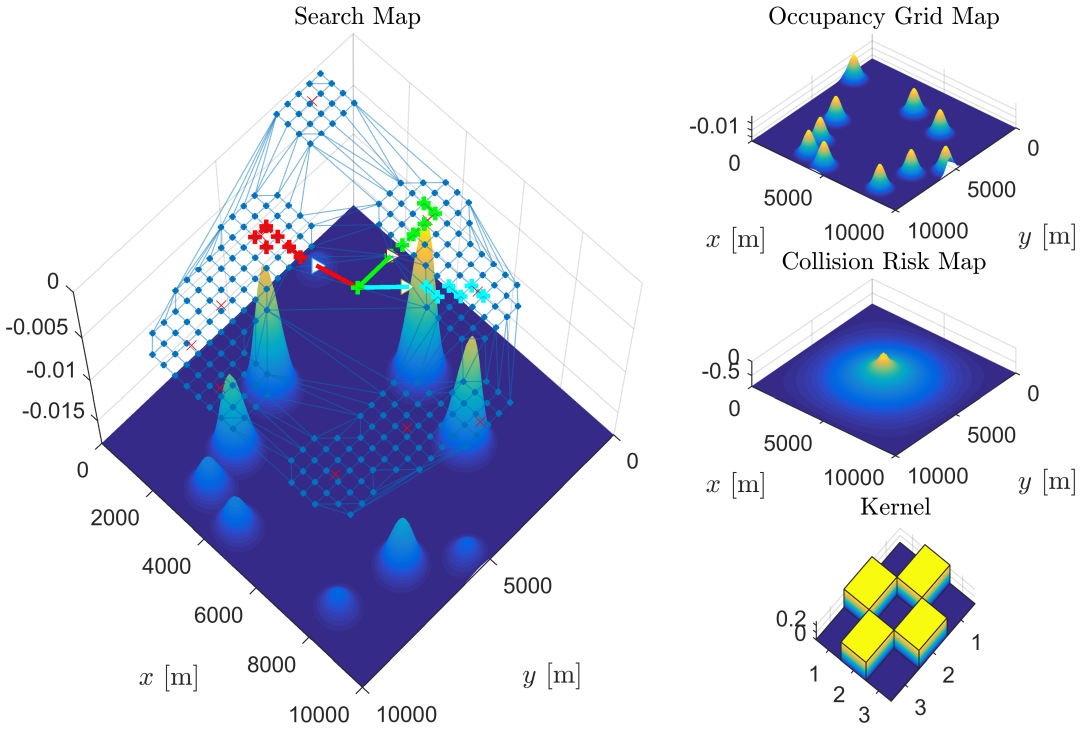


Figure 3.2: Visualization tool.

approach does not lead to loss of generality as the same structure is applicable to actual field tests. The advantage is the separation of the image processing problem, allowing sole attention to the development of the path planning and target position estimation.

3.2 Software in the Loop

In order to make sure the software from Section 3.1 performs well when integrated in a fully operational system, a SITL test environment was set up. SITL is a useful tool for developers as it simulates hardware, physical dynamics and data flow in real time, enabling extensive testing of the software without physically connecting the hardware.

In order to simulate the flight dynamics of the UAVs acting as agents for the system, the open source flight dynamics model JSBSim [2] is used. JSBSim is a standalone program capable of compiling and running under many operating systems. It takes input through a script file and various configuration files and can relatively easily be incorporated in a larger flight simulator system. The specific airframe used for simulations in this thesis is the Skywalker X8 Flying Wing (Figure 3.3) as this frame is available at NTNU for potential field testing.

ArduPilot [18] is used as the closed loop autopilot for the agents. Even though ArduPilot has many powerful features in terms of autonomy, it will serve our purpose by simply navigating the UAV by the waypoints supplied by the centralized ground computer. ArduPilot passes the output of the aircraft actuators to JSBSim, and JSBSim will use the flight dynamics to reply with the new states of the agents, including their new pose. One of the powers of ArduPilot is the possibility to run it on numerous different hardware systems, including a regular laptop. This makes it possible to simulate an aircraft with an associated autopilot on a laptop, while the interaction possibilities are exactly the same as if it was running on an on-board computer.

DUNE Unified Navigation Environment (DUNE) [14] is used as a command and communications link between the agents and the centralized computer. Dune is a run-time environment for unmanned systems on-board software, and will pass the data from ArduPilot and JSBSim to MATLAB and vice versa. DUNE can be used to write specific embedded software at the heart of a system, such as features of control, navigation, communication, sensors and actuators. For the purpose of this thesis, DUNE enables all active agents to be initiated as individual *tasks* that communicate through a well defined protocol. The tasks are event driven, and all messages between an agent and the centralize computer are sent as predefined Inter-Module Communication (IMC) [11] messages. MATLAB and DUNE communicate through a socket, uniquely defined by an IP address and a port. This network interface encourages the centralized MATLAB software to not care about the agents hardware setup nor the physical communication medium as long as the socket is properly created.

To monitor the state of the agents during flight, the open-source ground control software, Neptus [15], is used. Neptus can be used for a variety of tasks, including mission planning, simulation, control execution, supervision, data logging and post mission analysis. Neptus can communicate directly with DUNE using the IMC protocol, making it the natural choice of ground control software for this thesis. A snapshot of Neptus' command and control GUI during SITL simulation can be seen in Figure 3.5b. The corresponding live MATLAB visualization of the mission at the same time instance can be seen in Figure 3.5a.

The system architecture of the complete SITL test environment can be seen in Figure 3.4. All agent instances were initiated on separate computers connected by a local area network (LAN) to the centralized computer. The fact that DUNE works as a pipeline between the ground station and the agents using IMC messages, means the exact same software tested in the SITL environment will be fully functional for an operational system

Figure 3.3: Skywalker X8 Flying Wing airframe. Photo: NTNU Unmanned Aerial Vehicles Laboratory (UAV-Lab).

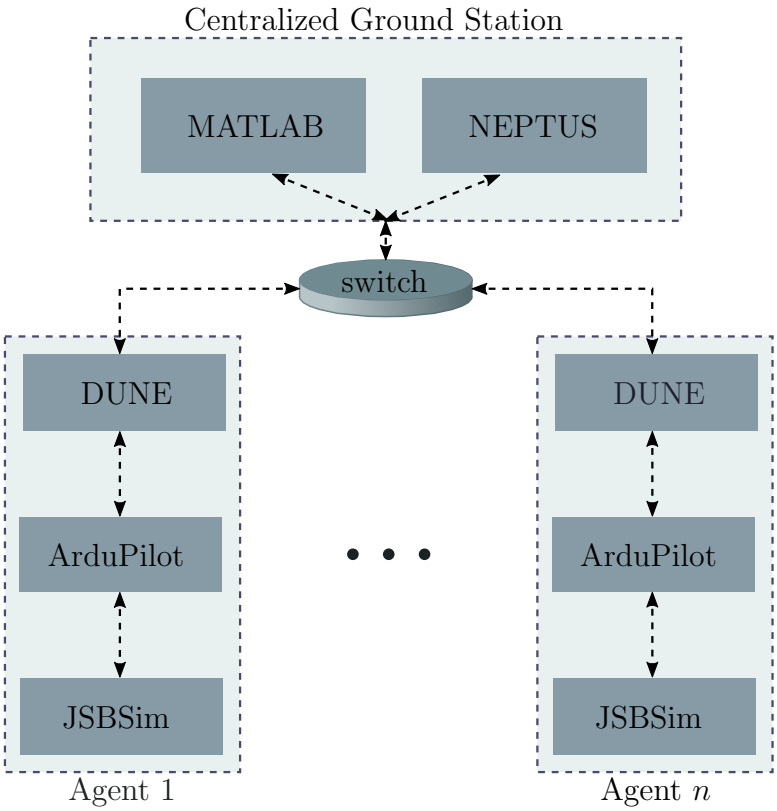
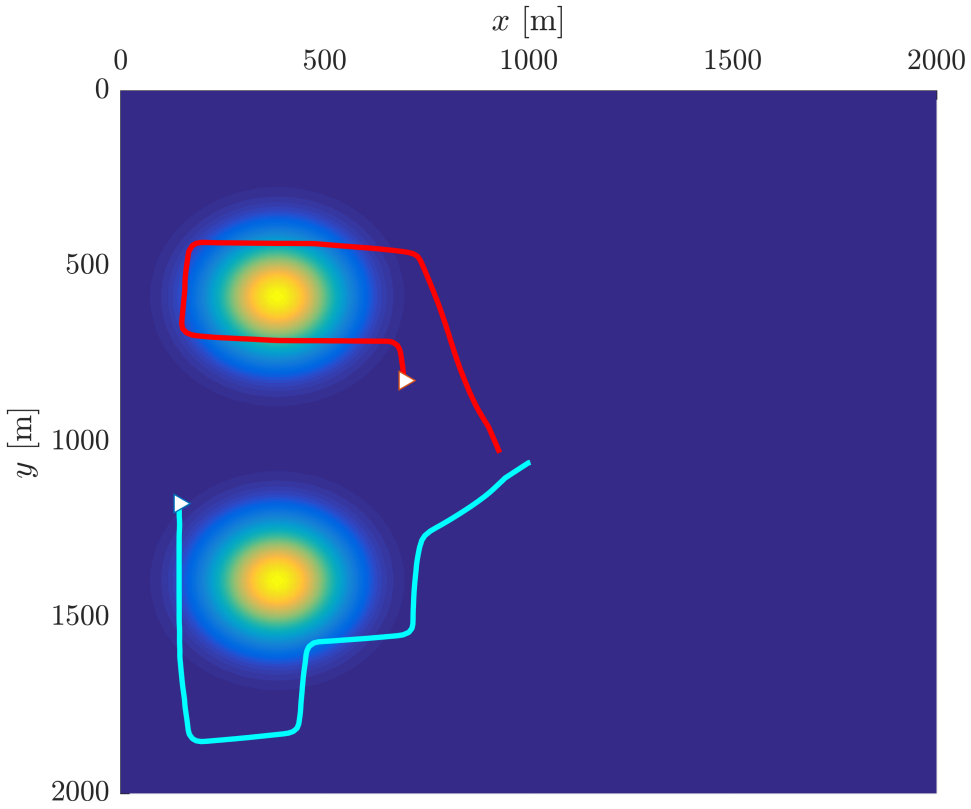
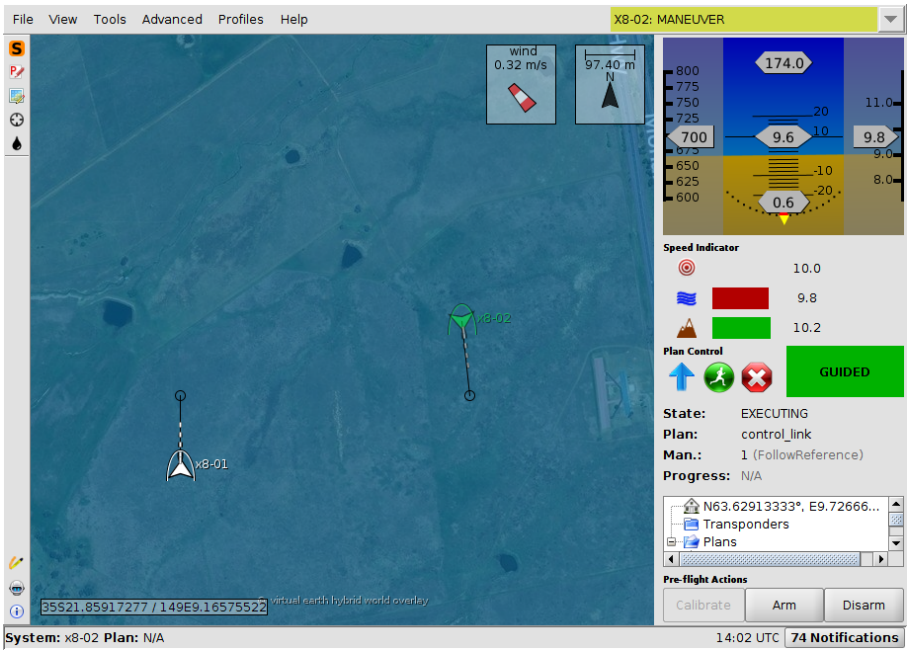


Figure 3.4: The architecture of the SITL test environment.

running on separate on-board computers using wireless communication links. Thus, the system implemented in this thesis is ready for field testing from a software perspective.



(a) Live search visualization by MATLAB.



(b) The Neptus command and control GUI.

Figure 3.5: Snapshot from SITL simulation.

Chapter 4

Simulations

In this chapter, the implemented system presented in Chapter 3 will be used to simulate various scenarios to empirically gain insight into the performance of the cooperative search algorithm outlined in Chapter 2. Section 4.1-4.5 will mainly investigate the performance and the scalability of the algorithm in terms of adjusting the number of agents participating in a search. In Section 4.6, a few of the design parameters of the system will be considered, to investigate the sensitivity to tuning and their impact on the system as a whole. To avoid biased results due to the specific configurations of a scenario, Monte Carlo simulations should ideally be employed with a high number of randomized scenario configurations to estimate the expected performance of the system. However, due to computationally heavy and time demanding simulations, simulating a high number of different configurations has not been feasible for all the tests conducted in this chapter. A few number of randomized configurations have however been simulated for each scenario in order to improve the accuracy of the results. For many of the simulated scenarios, plots are not included of all team sizes, and supplementary plots are shown in Appendix A.

As a note to the reader, the simulations in this chapter are performed without software in the loop, by only using MATLAB. This is due to the fact that all simulations running SITL are performed in real time, making each simulated scenario extremely time demanding. Furthermore, this chapter only intends to measure the behavior and choices made by the system. Thus SITL, which test hardware compatibility and real time requirements, are not important to the results, and will be disabled. Instead the UAV dynamics is simplified as steps with the assigned velocity in the direction of the next waypoint. Due to the large scale of the search regions in this chapter, it turns out the resulting trajectories have little difference from the ones obtained when running SITL for the same configurations.

4.1 No Prior Target Information

To test the path planning capabilities of the system, simulations have been carried out for a fixed number of stationary targets with no prior location information. The prior occupancy grid map $bel(m_0)$ is given the uniform belief 10^{-3} for all cells, leaving only the collision risk grid map to affect the path choices. The scenario parameters can be seen in Table 4.1. The search is executed by teams ranging from one to five agents, with a

Table 4.1: Simulation configurations for no prior target information.

Parameter	**Value**
Physical map size	10,000m × 10,000m
Number of grid cells	100 × 100
Number of nodes	25 × 25
MPS horizon length	6
Graph threshold	10^{-4}
Agent velocity	15m/s
Agent field of view (FOV)	440m × 440m
Number of targets	10
Target velocity	None

collision risk grid map created from the kernel

$$\mathcal{K} = \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{4.1}$$

As this kernel does not give any certain knowledge of the direction of target movement, the resulting collision risk grid map is exponentially decreasing from the offshore rig in all directions. To get a more accurate measure of the search durations, the simulation is carried out for three randomized target configurations consisting of 10 targets each. Figure 4.1 shows the resulting agent trajectory from a simulation with a team consisting

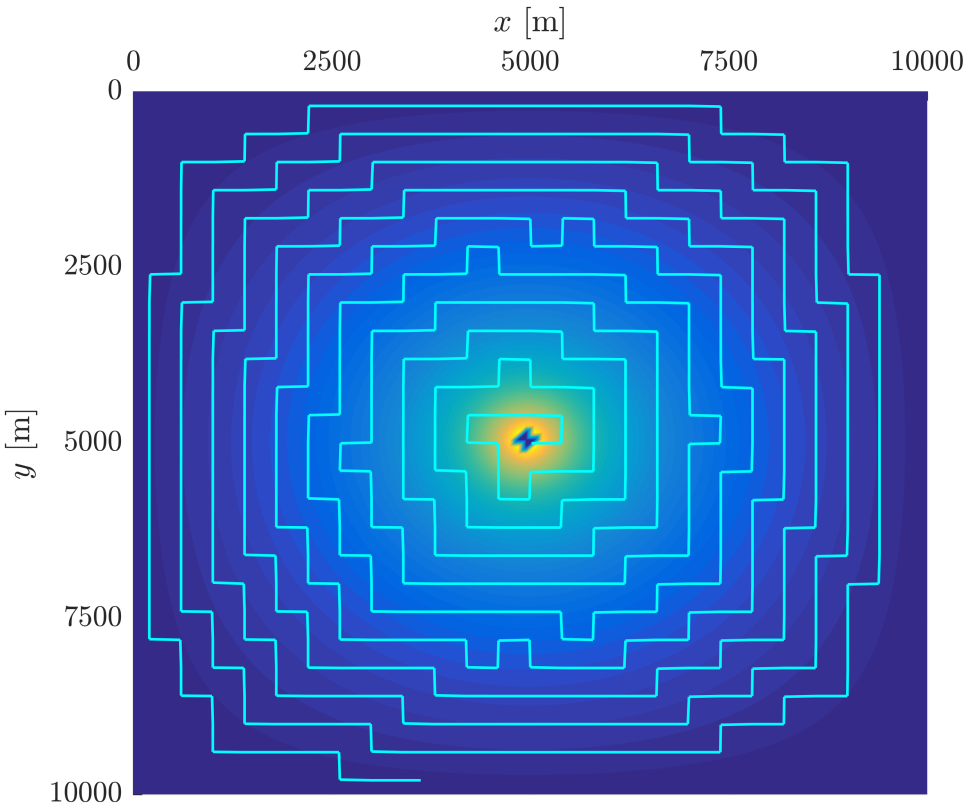


Figure 4.1: One agent searching for targets with no prior target information.

of one single agent. The colored map visualizes the search map s_t where yellow areas

indicates high cell values, whereas dark blue represents low cell values. With no prior target information, it can be seen that the resulting trajectory takes form as a growing spiral starting in the offshore rig. The rationality of this behavior is supported by the fact that in geometric exploration, the spiral search starting from a given point of interest is in fact the optimal search strategy for objects whose positions are unknown [5].

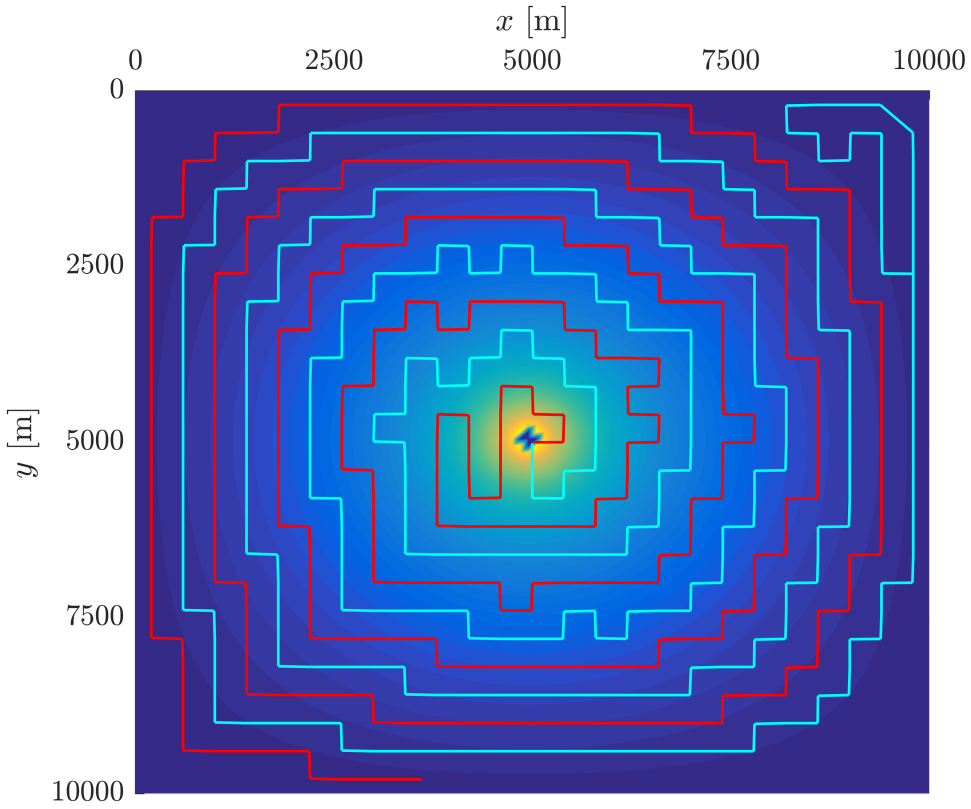


Figure 4.2: Two agents searching for targets with no prior target information.

Figure 4.2 shows the resulting search trajectories for a team consisting of two agents. The cyan trajectory represents agent one's search trajectory while the red represents agent two's trajectory. The search pattern is very similar to the case with a single agent, except the agents are now cooperating such that the resulting search spirals are growing approximately twice as fast. The increased efficiency caused by the cooperation can be seen from the search durations (Figure 4.3), where the mean search duration decreases from 236 to 117 minutes by increasing the team from one to two agents. This is an indication that deploying two agents opposed to just one, will double the efficiency of the search for situations where no prior target information is obtainable. Figure 4.3 also shows the curve constructed from the means of the search durations by least-squares fitting. The results show an exponential decrease in mission time by increasing the team size.

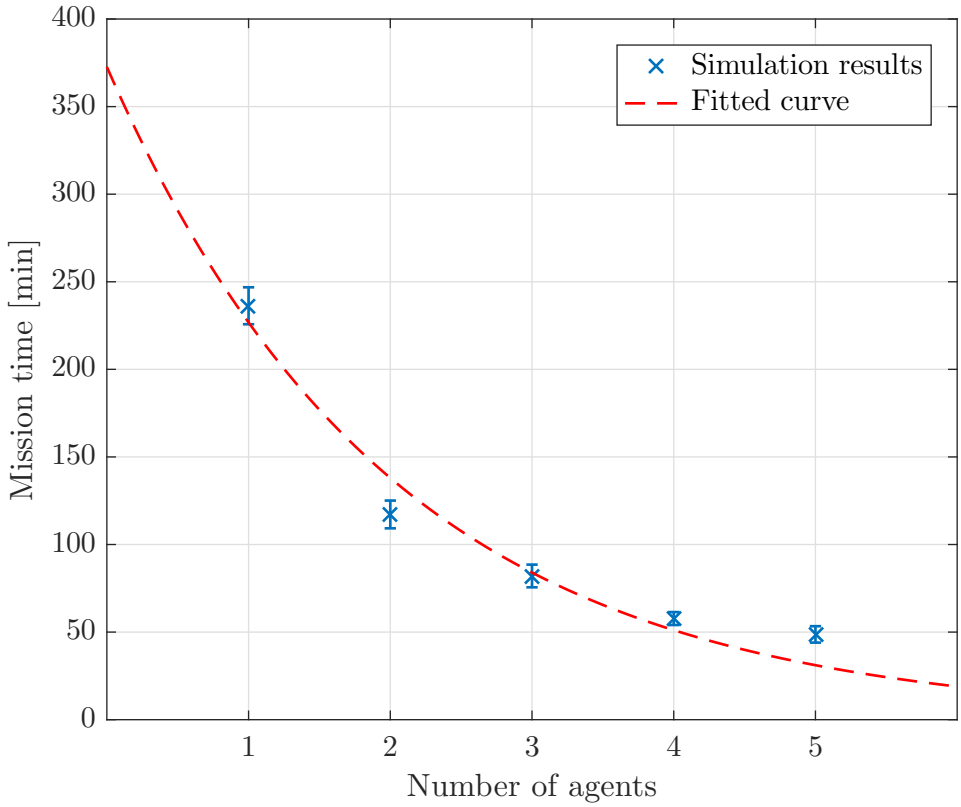
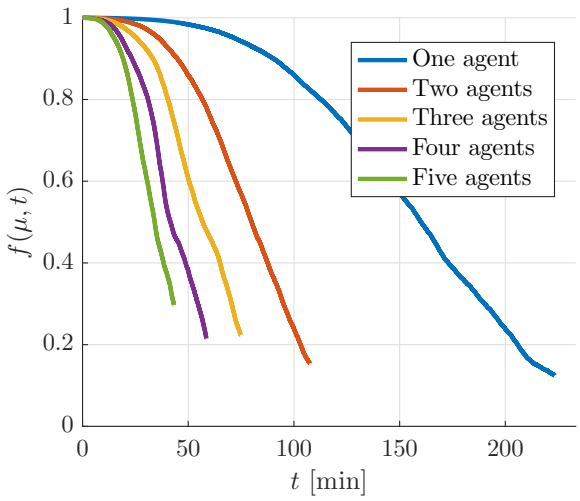
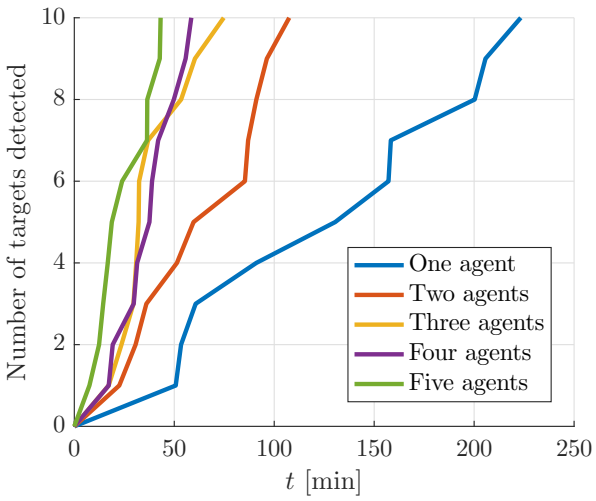


Figure 4.3: Error bar and curve fitting from search durations for teams ranging from one to five agents where no prior information is given.



(a) $f(\mu, t)$ over time for all search parties.



(b) Number of targets detected over time for all search parties.

Figure 4.4: Objective function value and number of targets detected over time where no prior information is given.

When considering performance and scalability is also of interest to compare the development of the objective function $f(\mu, t)$ between teams of different sizes (Figure 4.4a). An interesting note is how the general shape of $f(\mu, t)$ is similar for all team sizes, even though the search patterns vary with the size of the team (Figure A.1 shows trajectories

for teams of size three to five agents). The objective function value initially decreases slowly until the most critical regions have been covered. After this, $f(\mu, t)$ decreases in an almost linear fashion for all team sizes. Furthermore, from Figure 4.4a it seems like the time for each team to reach any given value in the objective function is also exponentially decreasing with the team size.

Figure 4.4b shows an example of the development of target detection for the various team sizes. The number of targets detected increases in a linear fashion, and the slope increases as the number of agents is increasing.

4.2 Prior Information on all Targets

Since the system presented in this thesis can use the prior information of the search area to its advantage, it is of interest to investigate the performance for cases where we to some degree have prior information of all targets as a search is initiated. As in Section 4.1, 10 stationary targets are distributed across the search area. All prior target distributions are modeled as Gaussian distributions with the same covariance matrix

$$\mathbf{C} = \begin{bmatrix} 250 & 0 \\ 0 & 250 \end{bmatrix} \tag{4.2}$$

and three randomly generated initial position configurations will be simulated to give a good measure of the search times. The configuration parameters are the same as in the previous section (Table 4.1), and the collision risk grid map is calculated from the kernel (4.1). Figure 4.5 shows the mean value and the variance of the search time as a function of the search party size. The means of the search durations does not have the same obvious exponential decrease as seen in section 4.1. This is most likely due to the fact that the target configurations have a large significance when this amount of prior information is given. Thus, three simulated configurations is likely not enough to get a good measure of the average search durations. However, the time to detect all targets is significantly lower than the resulting times when no prior information was given, which is an indication that the system is able to take advantage of the prior information of the target topology. The reason for the high variance of the mission durations in Figure 4.5 is also due to the fact that the trajectories of the agents are highly dependent on the target positions when we have a high degree of prior target information, as opposed to the spiral search pattern seen when no prior information is given.

Figure 4.6 shows the search trajectory of a single agent for one of the simulated scenarios. As seen from the plot, the agent goes straight for the mean when it approaches a target Gaussian. Furthermore, when multiple Gaussian target distributions are close together, the agent tends to cover all the means before systematically going back to cover the remains of the target distributions. After all, the search objective outlined in Section 2.3 considers the product of the contributions to a collision from each individual grid cell. This leads to prioritizing covering high reward areas as fast as possible, even if they are separated by stretches of low priority. This search strategy might result in an overall increased search duration, but a faster decrement in the objective function. Thus, this behavior seems like a rational search policy from a perspective of protecting an offshore installation. Figure 4.7 shows the same search scenario for a team of two agents. As expected, when increasing the number of agents the search party will fan out and focus on different targets. Plots showing the trajectories of teams ranging from three to five agent can be seen in Figure A.2

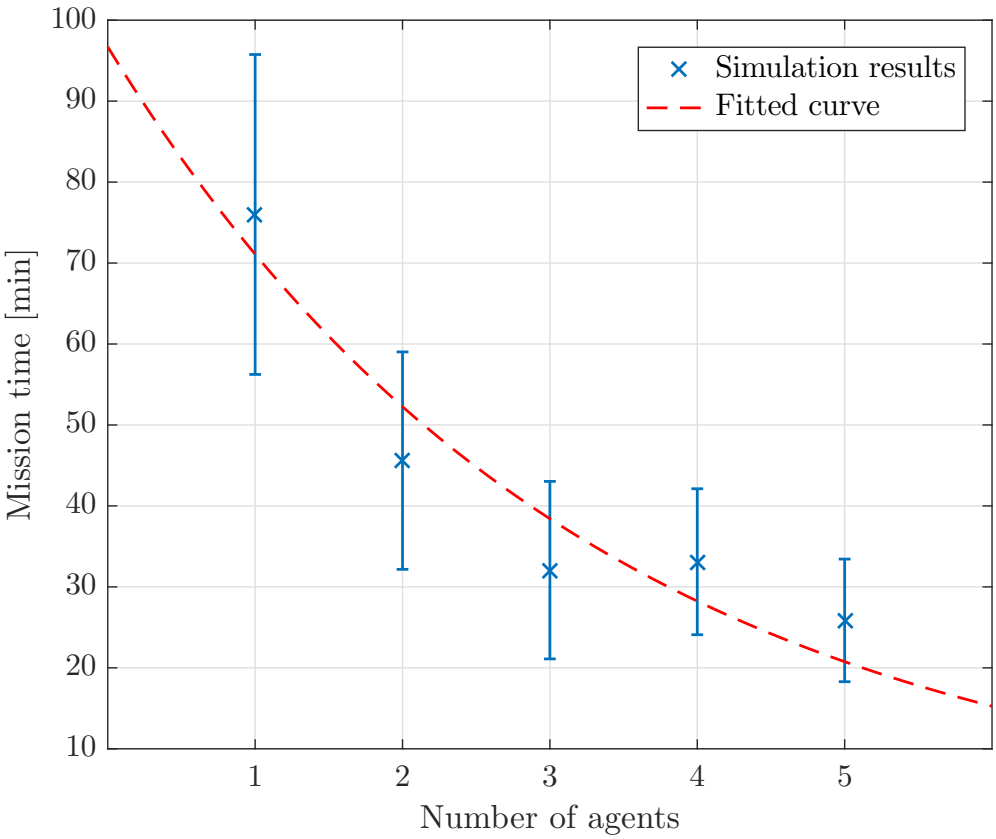


Figure 4.5: Resulting curvefitting for the mission times versus team sizes ranging from one to five agents when prior information about the targets are given.

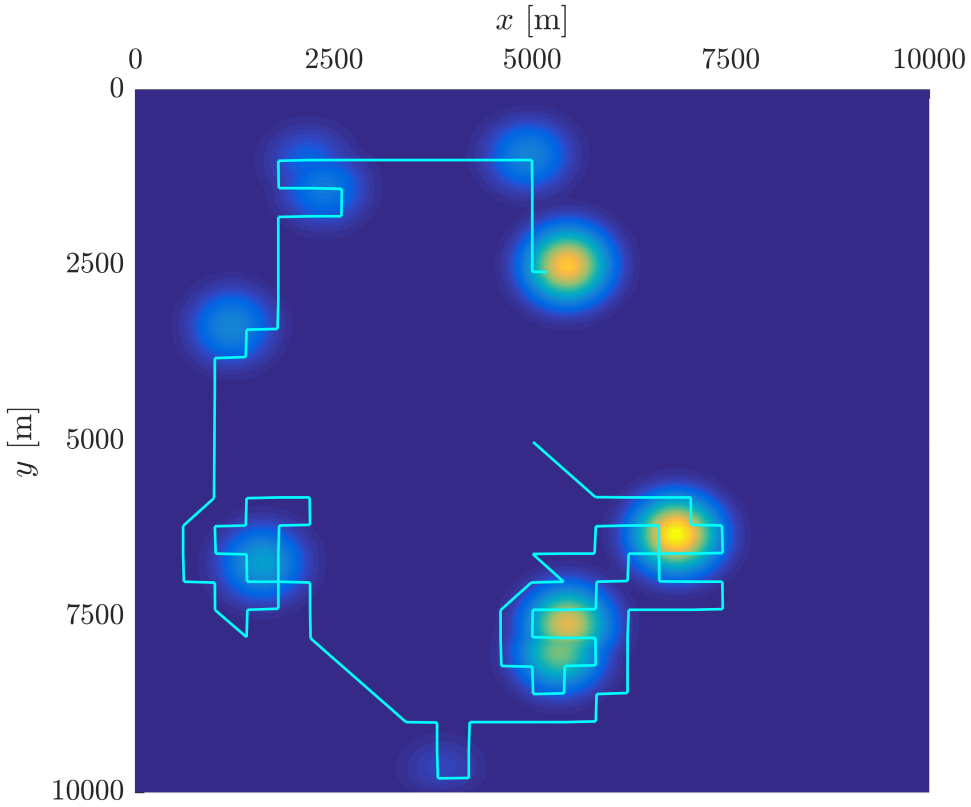


Figure 4.6: Search trajectory for one agent where prior information is given on all targets.

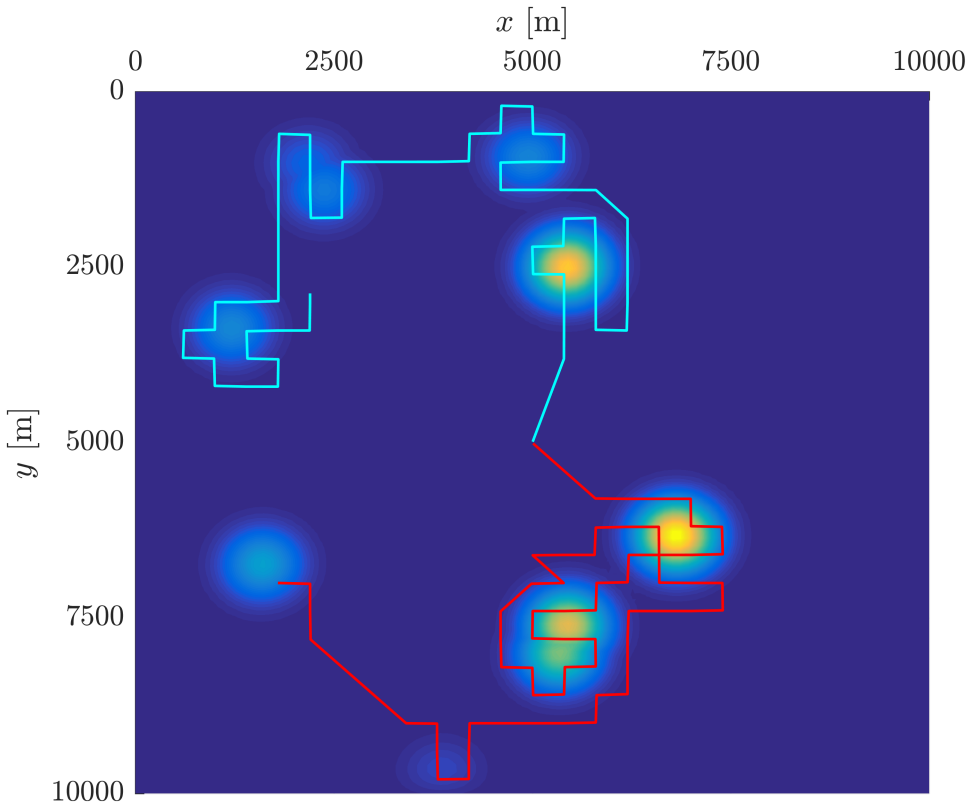
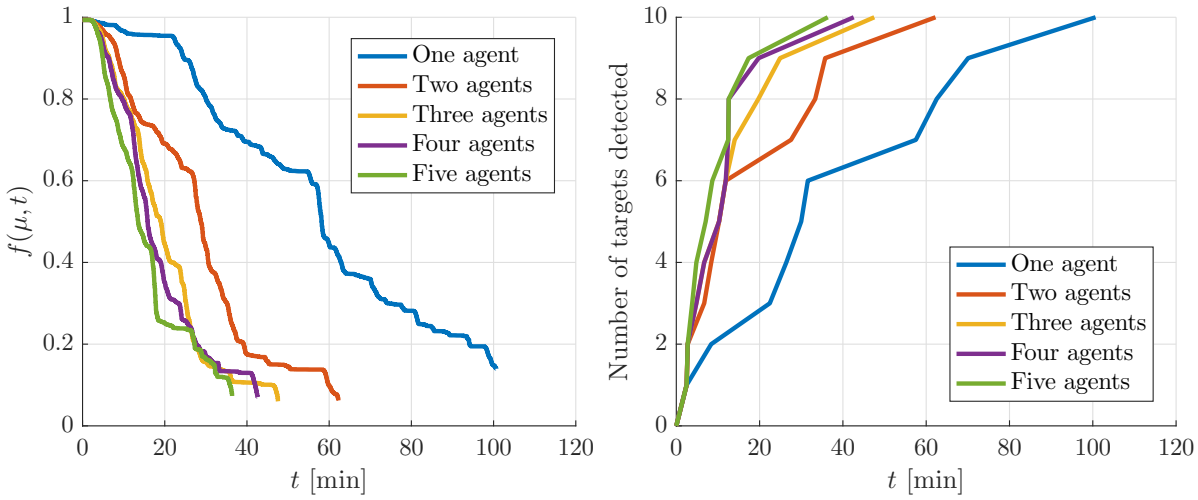


Figure 4.7: Two agents searching for targets where prior information is given on all targets.

When looking at the values of the objective function (Figure 4.8a) we can see it reduced in a more uneven fashion than than the case of no prior target information. This is because of the stretches of low priority that has to be traversed in order to reach the next high reward area. Furthermore, we see that for this particular scenario there is little to gained by deploying a team of more than three agents.



(a) $f(\mu, t)$ over time for all search parties.

(b) Number of targets detected over time for all search parties.

Figure 4.8: Objective function value and number of targets detected over time where prior target information is given.

4.3 Prior Information on a Subset of the Targets

A realistic ice management scenario might involve a search region where satellite images have provided some degree of prior target information. The information provided by a satellite might however be outdated or coarse, leading to the chances of unknown targets in the search region. Thus, it is of interest to investigate how the system will perform in situations where prior information is given on a subset of the targets. The same target configurations as in the previous sections will be tested, where five of the targets will be given a prior Gaussian distribution with the covariance matrix (4.2). No prior information is given on the five remaining targets, and the search proceeds until all targets are found. The configuration parameters of the simulations are listed in Table 4.1. The search durations for teams of various sizes can be seen in Figure 4.9. It makes sense that the means of the search times fall in between the case of a high rate of prior target information and the case of no prior information at all.

Figure 4.10 shows the search pattern for a single agent. Like the resulting trajectory displayed in Section 4.1, a spiraling path from the starting point can be observed. However, the agent makes detours once it comes within reach of known target distributions, leading to a faster discovery of the targets with prior information. Figure 4.11 displays the same scenario for two agents. The combination of prior target probability distributions and areas with an unknown target probability does not seem to cause trouble for the cooperative behavior of the agents. As seen from the trajectories there are no conflicting path choices between the two agents. This is also reflected in the objective function plotted for various team sizes (Figure 4.12a), where we see the rate of reduction of the objective value is significantly increased as the team size increases. The trajectories of teams ranging from three to five agents can be seen in Figure A.3.

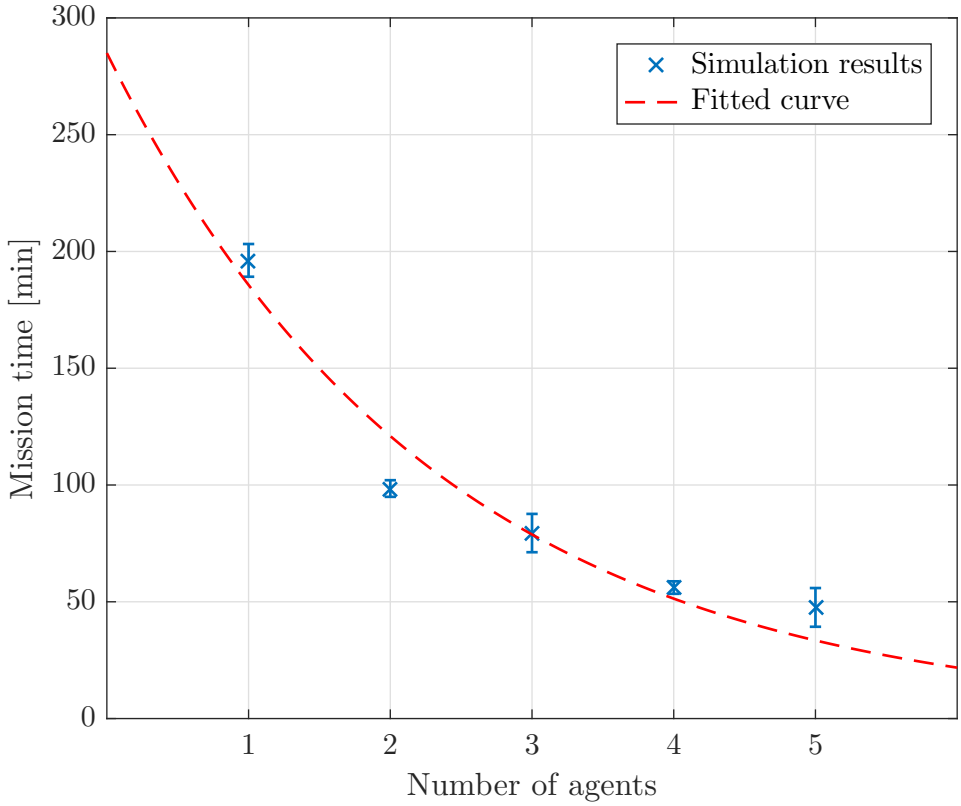


Figure 4.9: Resulting curvefitting for the mission times versus team sizes ranging from one to five agents when prior information is given about a subset of the targets.

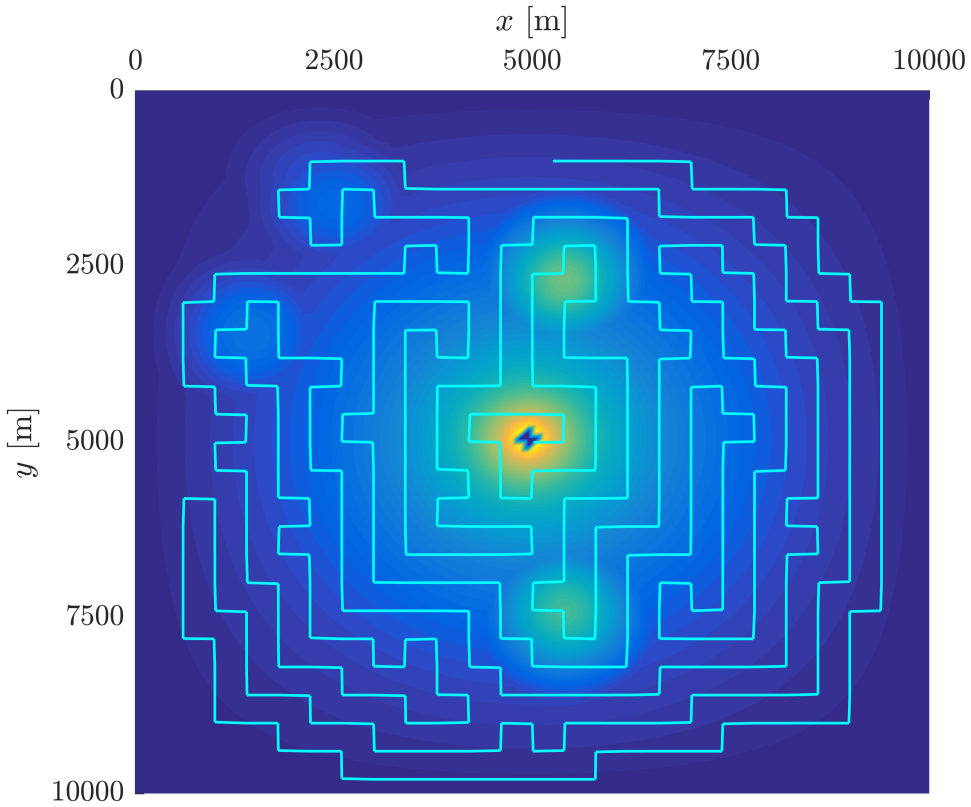


Figure 4.10: Search trajectory for one agent where prior information is given on a subset of the targets.

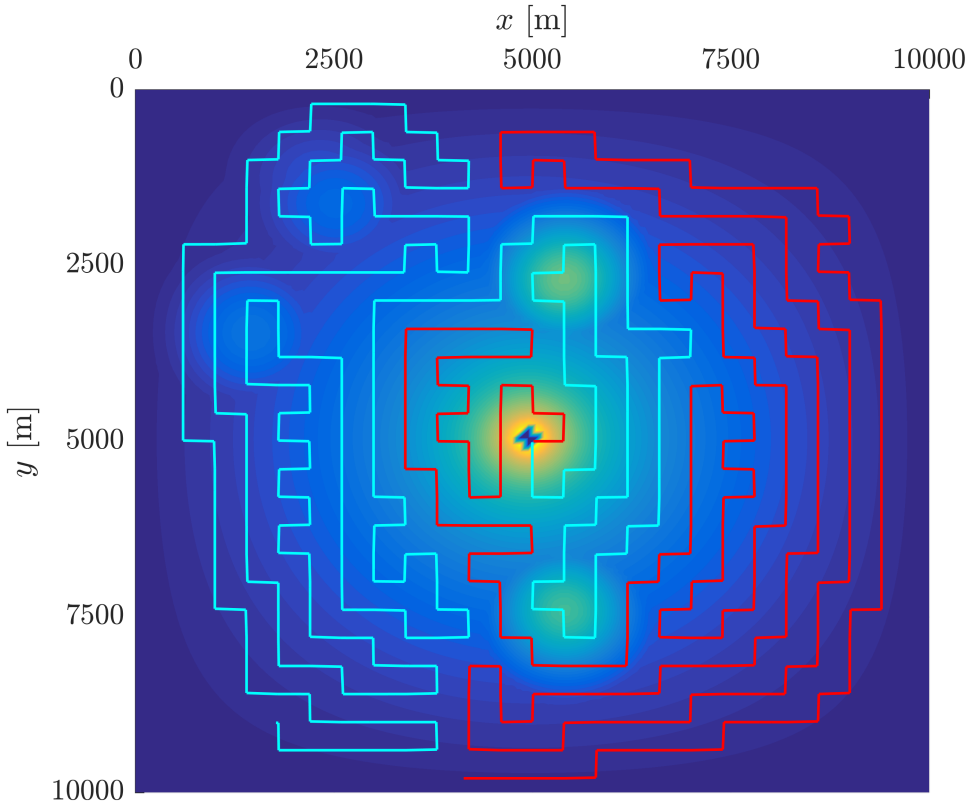
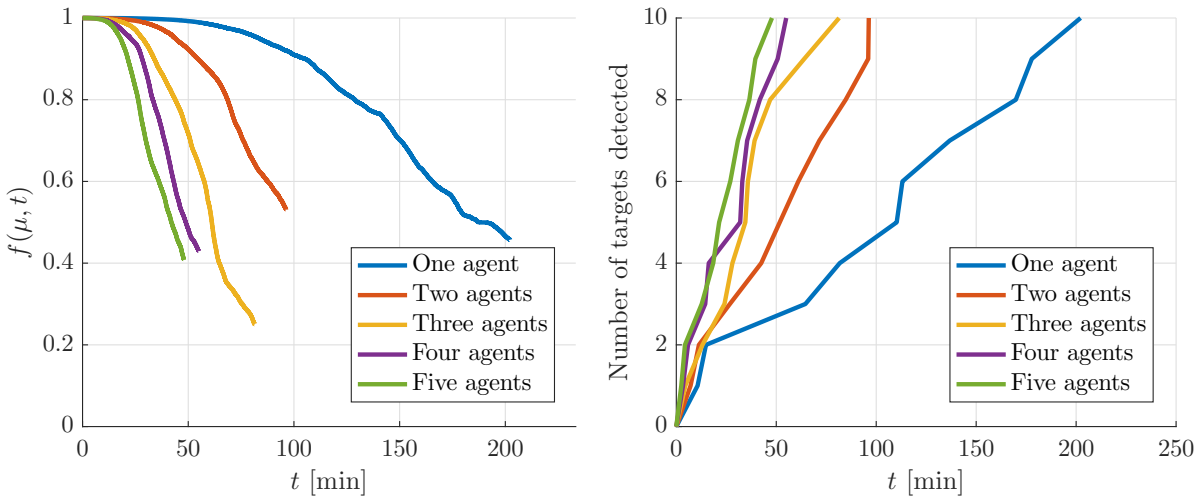


Figure 4.11: Search trajectory for two agents where prior information is given on a subset of the targets.



(a) $f(\mu, t)$ over time for all search parties.

(b) Number of targets detected over time for all search parties.

Figure 4.12: Objective function value and number of targets detected over time where prior target information is given on a subset of the targets.

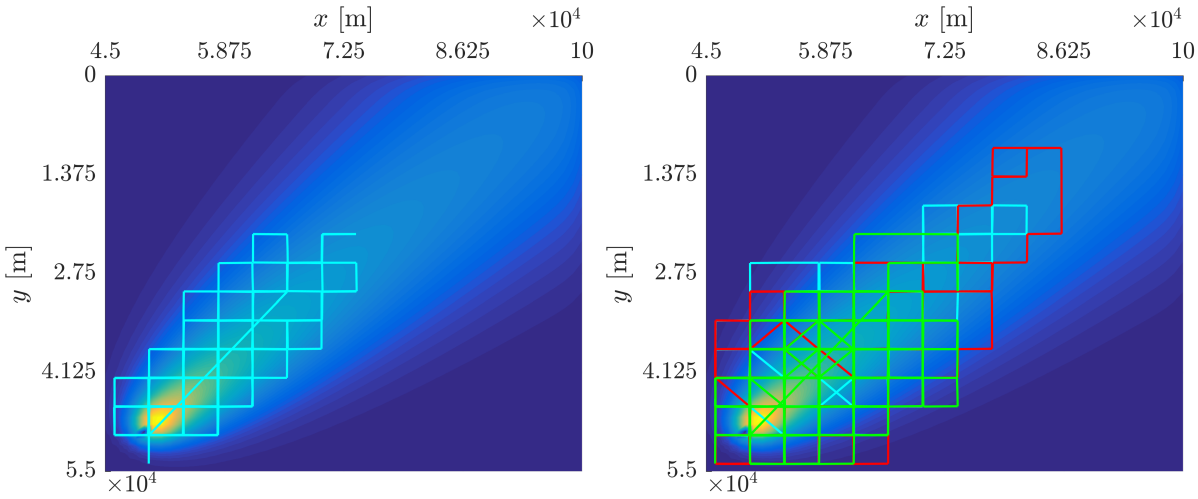
4.4 Team of Agents Compared to Satellite Imagery

A natural question arising from this thesis is whether the system developed would be a contribution to modern ice management. In order to get a measure of the contribution, we will compare the system with the use of satellite images for sea ice detection, which is a method currently used in ice management. We will consider a scenario with a timespan of 72 hours for a search area of 100km × 100km with the offshore platform positioned in the center of the search area. For this simplified scenario, targets in the search area are given a velocity with mean value 2m/s acting south-west. During the 72 hours of simulation time, 1000 targets are randomly sent into the search area. The parameter list can be seen in Table 4.2. Figure 4.13 shows the search trajectories for team sizes of one and three agents. The grid shaped pattern of the trajectories are a result of the map dynamics. There is a steady flow of unknown target probability drifting in the direction of the platform and the agents are not able to cover all the ground before having to venture back to reinvestigate the previously searched areas. The result is a patrolling behavior by the agents where they keep the regions deemed the most critical under constant surveillance, going back and forth across the lattice graph. A pleasing result is that extending the team size also extends the area the search party is able to patrol in order to protect the offshore platform from incoming threats. Agent trajectories for teams of two and four agents can be seen in Figure A.4.

Table 4.2: Configuration for the long term search.

Parameter	**Value**
Physical map size	100,000m × 100,000m
Number of grid cells	100 × 100
Number of nodes	25 × 25
MPS horizon length	6
Graph threshold	10^{-3}
Agent velocity	15m/s
Agent Field of view	4400m × 4400m
Mean target velocity	2m/s south-west

As a means of comparison we will look at the effects of satellite images without the aid of the UAS. In order to put this in the framework of this thesis, we incorporate a satellite image in the sense of the Bayesian target estimation from Section 2.2, as a large measurement z_t covering the entire search map. Figure 4.14 shows the value of the objective function for various time intervals of satellite images being taken. Prior information is given of the initial configuration of the targets. As seen from the Figure 4.14a, our initial information of the targets will drift towards the platform causing an abrupt increase in the objective function. As all initial information drifts away from the critical regions, the objective function will stabilize because of the steady influx of unknown target probability into the search map. The same behavior can be seen in Figure 4.14b when another satellite image is provided after two days. The belief of all grid cells will either spike or decrease significantly, causing the objective function to go to zero. What is interesting to note however, is that when increasing the frequency of satellite images provided, the average objective function value go up. This is due to the fact that a satellite image provides information of the entire search region, which in cases of high target density will cause a rise in the near future of the estimated risk of



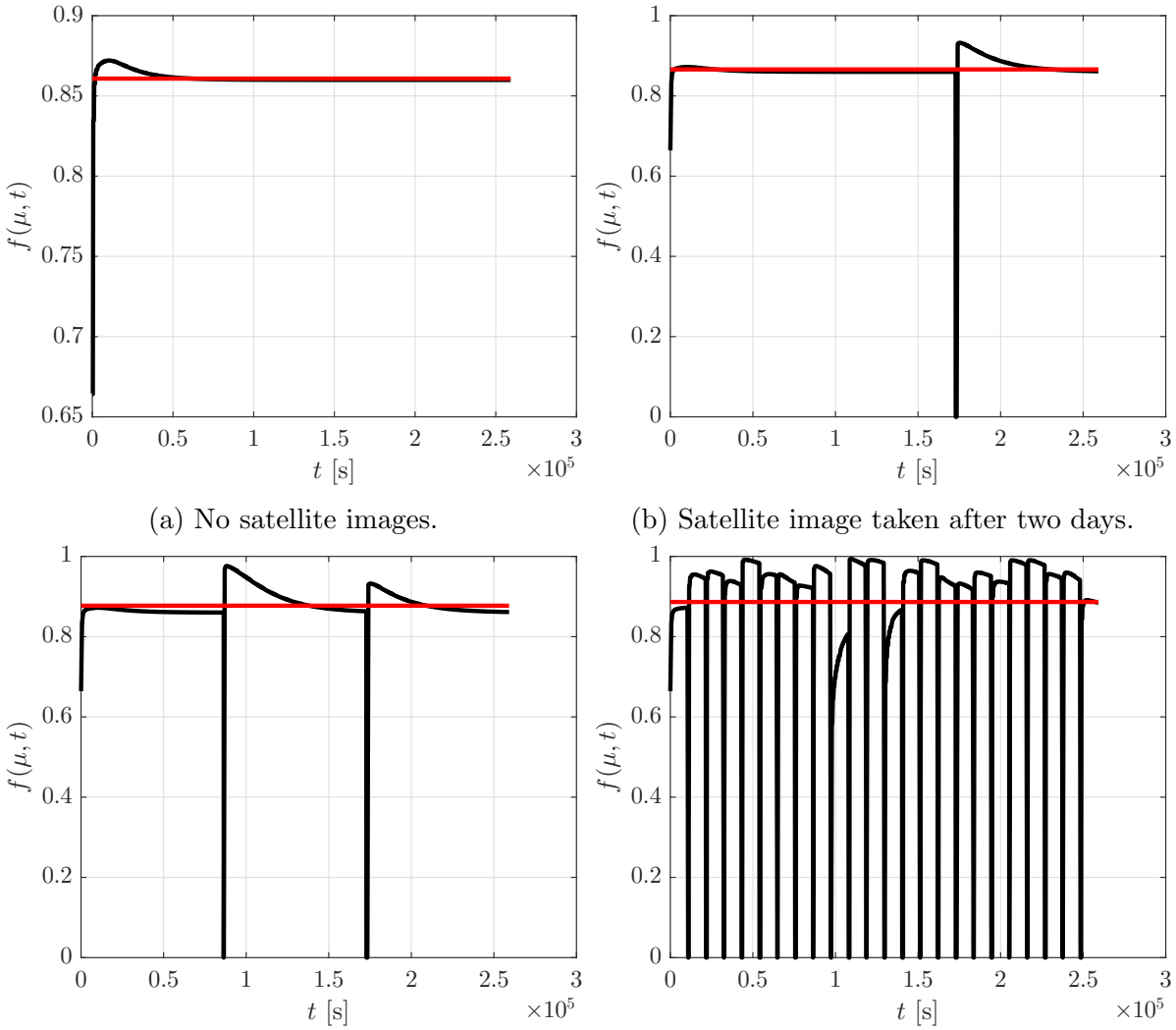
(a) Path history for one agent.

(b) Path history for three agents.

Figure 4.13: A section of the search area displaying path history for different team sizes. The underlying plot is the collision risk grid map.

collision. Figure 4.15 displays the objective function values for the exact same scenario for search parties ranging from 1-4 agents. Seen from the plots, the objective function $f(\mu, t)$ slightly increases when increasing the team size. The reason for this increase is the fact that a larger team will uncover more potential threats, causing an increase in $f(\mu, t)$ over time when the map dynamics are brought into effect. When increasing the team size to such an extent they have the majority of the critical areas under some degree of constant surveillance, simulations show the average value of $f(\mu, t)$ will again decrease.

By comparing Figure 4.14 and 4.15, we see that all team sizes manage to score a lower average objective function value than any of the simulated satellite image frequencies. We also see that all team sizes produce an average objective value that is lower than the case of only unknown target probabilities as seen in Figure 4.14a. While supplying satellite images gives a detailed update of the entire search region once in a while, a searching team of agents give continous updates only in critical regions. Of course, the optimal approach would be a combination of the two. It should be noted that although the use of UAS yield better overall results in these simulations, they are performed for teams of agents flying continuously for 72 hours. Even though agents could be replaced one by one as they were drained by battery life, this is a slightly unrealistic use case.



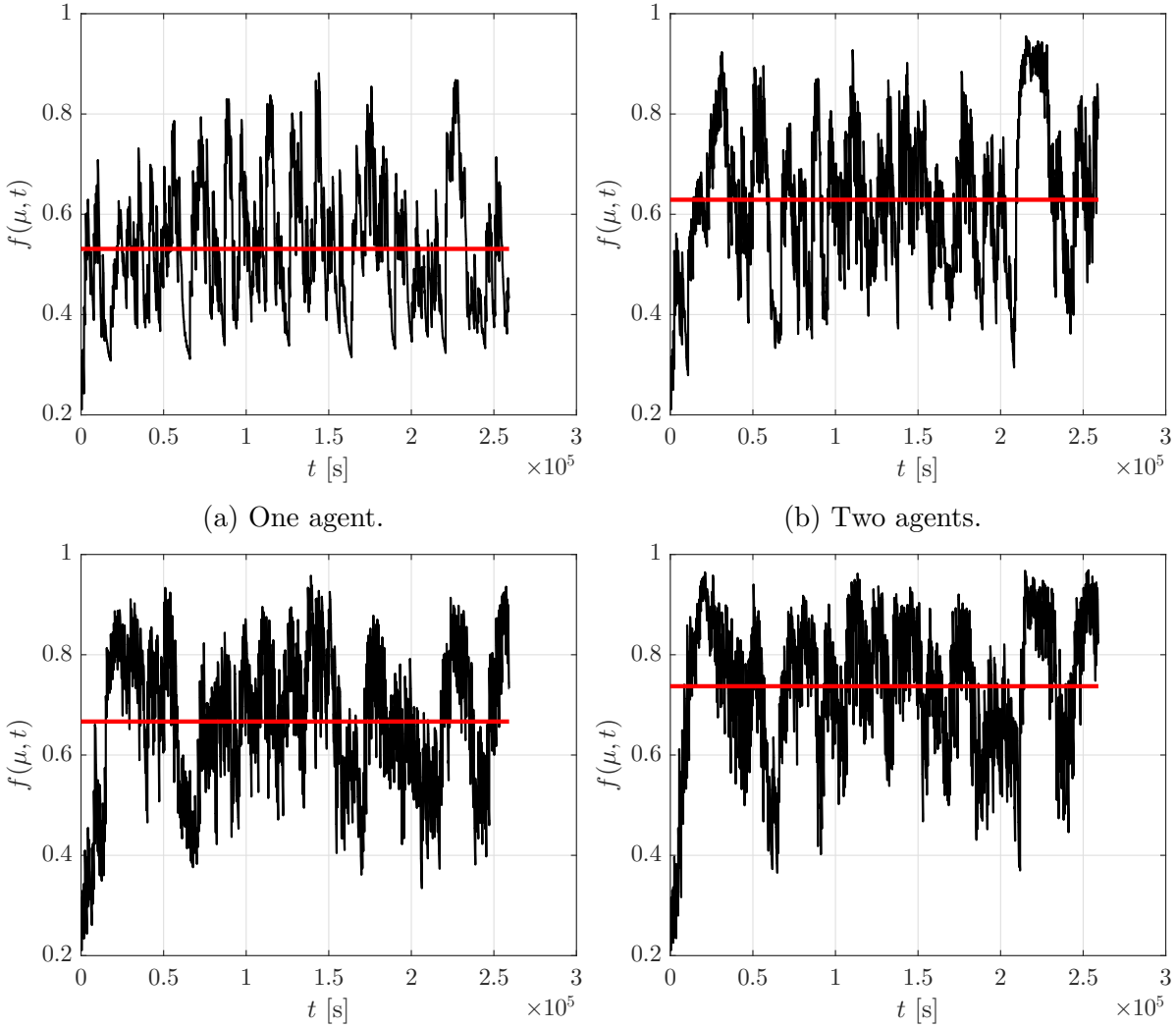
(a) No satellite images.

(b) Satellite image taken after two days.

(c) Satellite image taken once every day.

(d) Satellite image taken every two hours.

Figure 4.14: Objective function value plotted for various intervals of satellite images taken. The red line shows the average objective value.



(a) One agent.

(b) Two agents.

(c) Three agents.

(d) Four agents.

Figure 4.15: Objective function value plotted for team sizes. The red line shows the average objective value.

4.5 Tracking Performance

If we are fairly certain a cell in the occupancy grid map contain at least one target, the corresponding belief of the cell $bel(\mathbf{m}_{i,t})$ will have a value close to one. As a result of the map dynamics presented in Section 2.2.2, the predictive step may distribute the belief of this cell to the surrounding cells, resulting in a higher value of the priority function $\delta(bel(\mathbf{m}_{i,t}))$ for the surrounding cells. Eventually the value of the search map s_t will rise above the graph reward threshold τ, causing a node to appear in the search graph enabling agents to revisit a previously investigated area. Hence a notion of tracking is introduced to the behavior of the system. This section will isolate the tracking feature and compare the performance between team sizes ranging from one to three agents.

Table 4.3: Configurations for tracking scenario.

Parameter	**Value**
Physical map size	10,000m × 10,000m
Number of grid cells	100 × 100
Number of nodes	25 × 25
MPS horizon length	6
Graph threshold	10^{-3}
Agent velocity	15m/s
Agent Field of view	440m × 440m
Target velocities	2m/s east
Kernel	$\begin{bmatrix} 0 & 0 & 0.1 \\ 0 & 0 & 0.8 \\ 0 & 0 & 0.1 \end{bmatrix}$

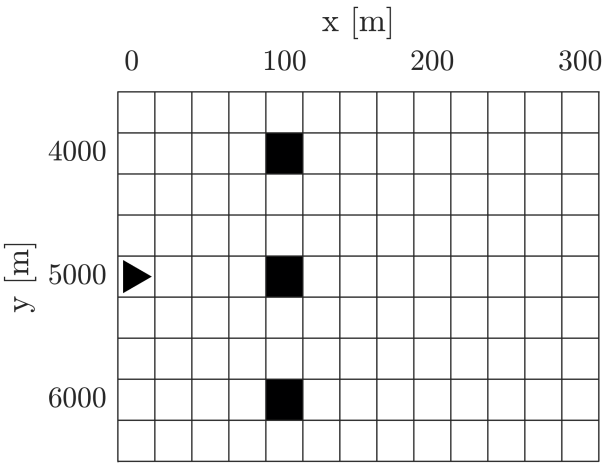
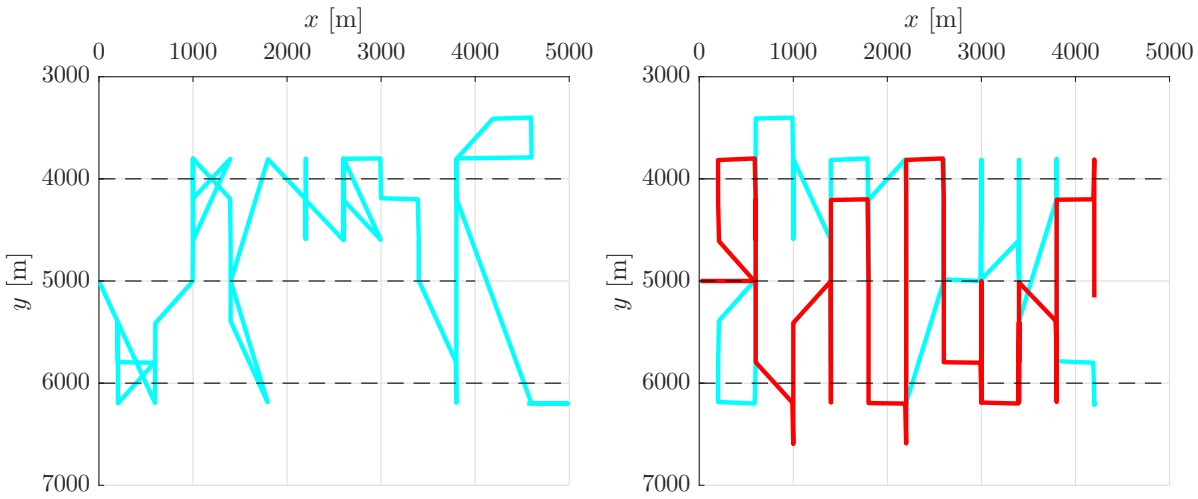


Figure 4.16: Configuration setup of tracking scenario. The black square represents the initial target positions and the triangle is the initial agent position.

The scenario is initiated with the configurations specified in Table 4.3. Three targets are initially placed at the coordinates $(100, 4000)$, $(100, 5000)$ and $(100, 6000)$, with agents initially placed at $(0, 5000)$ as shown in Figure 4.16. The positions of all targets are assumed to be fully known to the team of agents prior to the search, so $bel(\mathbf{m}_{i,0}) \in \{0, 1\}$.

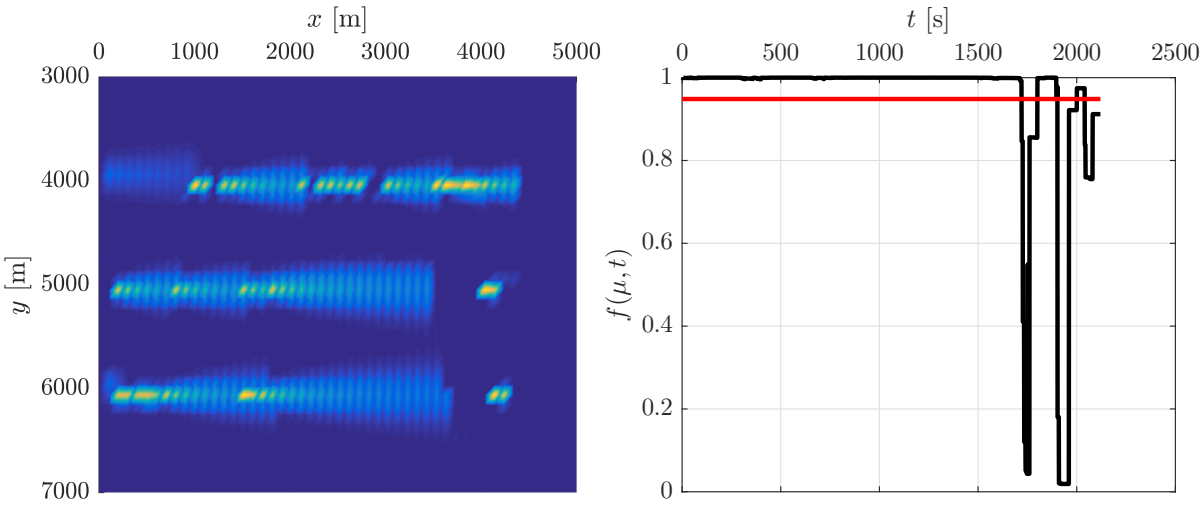


(a) Trajectory for team of one agent.

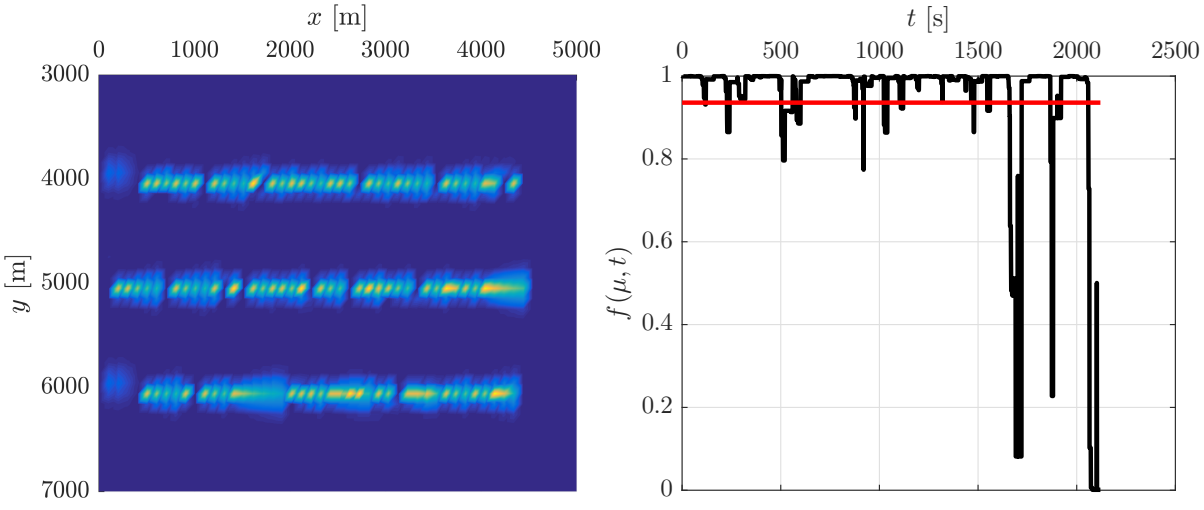
(b) Trajectory for team of two agents.

Figure 4.17: Trajectories of teams consisting of one and two agents tracking three targets moving towards the offshore rig.

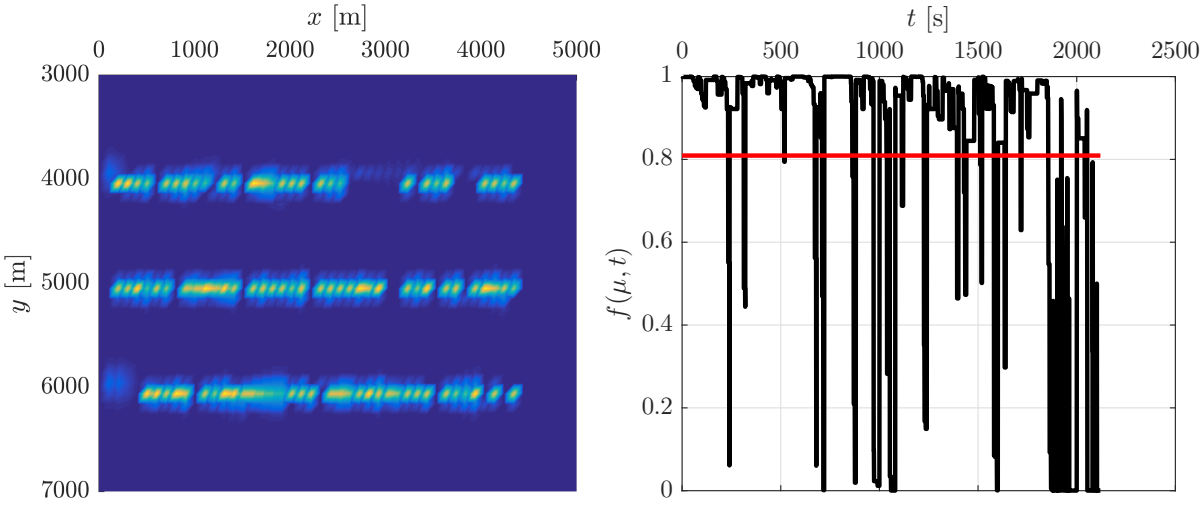
Figure 4.17 shows the resulting agent trajectories for teams consisting of one and two agents respectively. Trajectories for three agents are shown in Figure A.5. The main difference between a team of one and two agents is that two agents are able to cover more area and frequently update the position estimates of all the targets. Figure 4.17a shows that the single agent spends most of its time tracking the target following the line $y = 4000$, as opposed to regularly updating all target positions. This result is reflected in Figure 4.18a1, showing the development of the posterior estimate of the target positions for the single agent. As seen from the plot, the target following the line $y = 4000$ get more attention, and has a higher update rate. When the targets approach the offshore rig where the cells in the collision risk map contains high values, the last two targets are eventually revisited. Figure 4.18b1 and 4.18c1 show that the certainty in the posterior target estimations increases as the number of agents are increased for this tracking scenario. Figure 4.18a2, 4.18b2 and 4.18c2 show the objective function $f(\mu, t)$ over time for the three team sizes. A perfect tracking system would know the exact position of all targets at all times, rendering the objective function equal to zero at all times. Figure 4.18a2 shows that measurements from only one agent is not enough to reduce the objective function from the initial value, except when the targets get close to the offshore rig and the position information may be crucial for avoiding collisions. As the number of agents increases, the belief $bel(\mathbf{m}_{i,t})$ of the grid cells in general get a higher level of certainty, lowering the objective $f(\mu, t)$.



(a1) Team of one agent. (a2) Team of one agent.



(b1) Team of two agents. (b2) Team of two agents.



(c1) Team of three agents. (c2) Team of three agents.

Figure 4.18: The time development of the posterior estimates of target positions $bel(\mathbf{m}_{i,t})$ and objective function $f(\mu, t)$ over time for team sizes ranging from one to three agents. The red line is the average objective function value over time.

4.6 Effects of Parameter Tuning

The system developed in this thesis have several parameters specifiable by the user. It is of interest seeing how these parameters affects the behavior of the agents and the resulting search patterns. This section will isolate and adjust three of the most important parameters to examine their sensitivity to variations. The parameters to be examined are the relationship between the node spacing and size of the field of view (Section 4.6.1), the MPS horizon length (Section 4.6.2) and the cell transition probability kernel $\mathcal{K}_t$ (Section 4.6.3). In Section 4.6.3, the targets velocity vectors will be physically modeled by Gaussian distributions and Monte Carlo simulations are performed to estimate the expected result. However, Monte Carlo simulations are not performed in Section 4.6.1 and 4.6.2 as the statistical variations has limited influence on the validity of the results.

4.6.1 Relationship Between Node Spacing and Field of View

As described in Chapter 2.4.2, each node has an assigned field of region (FOR) containing the search map grid cells $\mathbf{s}_{i,t}$ in the neighborhood of the node. As an agent visit a node, the relationship between the agent's field of view (FOV) and the node's field of region will determine the amount of grid cells in the field of region the agent will examine. If the ratio between the field of view and the field of region is too large, the agent will examine cells belonging to the neighboring nodes' field of region, decreasing their search reward without even visiting them. By adjusting the node spacing, the size of the field of region is affected. A change in the ratio between the node spacing and the agents field of view will cause a corresponding change in the ratio between the field of region size and the field of view size. For a more intuitive metric, we define the FOV-to-FOR ratio r as

$$r := \frac{\text{FOV size}}{\text{FOR size}} \tag{4.3}$$

Table 4.4: Configuration of node spacing scenario.

Parameter	**Value**
Physical Size	5,000m × 5,000m
Number of grid cells	100 × 100
Number of nodes	25 × 25
MPS horizon length	6
Graph threshold	10^{-4}
Agent velocity	15m/s
Target velocities	None
FOV-to-FOR ratios	$[1, \ 1.5, \ 2]$

In this section, three scenarios with different values of r will be set up. The configurations of the search scenarios are listed in Table 4.4. Before the simulation results are presented, it is important to clarify the details of how the examination of grid cells is performed in the system implementation. When performing a measurement, every corner of a cell must lie inside the agents field of view for that cell to be registered as examined, such that a grid cell will not be updated by the Bayesian filter unless all the content of that cell is visible to an agent. Optimally, the FOV-to-FOR ratio r should be equal to one for an agent to only examine the cells inside a nodes FOR and ensure all the cells

in the particular FOR are examined. This implementation detail implies that the node placements should be in a manner such that the grid cells are positioned symmetrically around each node. Should this be the case, a FOV size equal to the FOR size would measure all cells belonging to the FOR of the visited node. This ideal outcome is however not always the case as both the number of cells and the node spacing can be set arbitrarily.

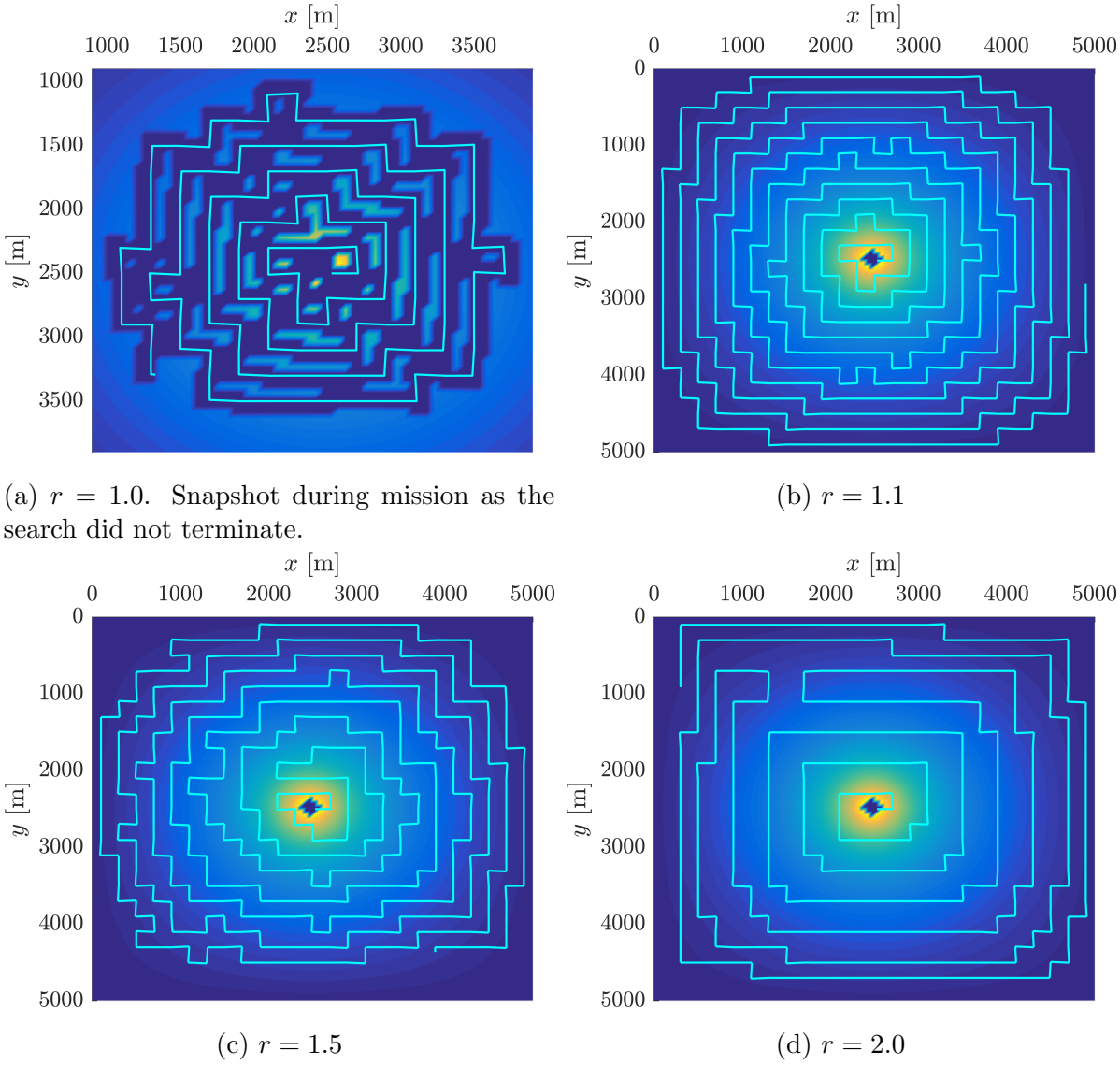
As we aim to isolate and compare the effects of adjusting FOV-to-FOR ratio, the size of the search party is irrelevant, thus all simulations in this section are performed for a single agent. The simulated scenarios are executed by letting the FOR size be the same for all scenarios, only adjusting the FOV size to reach the desired ratio r. This will make more visually intuitive results, as the waypoints are located at the same positions through all scenarios. However, it should be noted that for practical applications the field of view is dependent of the specifications of the camera used, and the flight altitude which depends on the weather conditions. In this setting, the FOR size should be designed according to the expected field of view to reach the desired FOV-to-FOR ratio. As this section only outlines the effects of ratio adjustments, the order of parameter determination is irrelevant.

Simulations results (Figure 4.19a) show the agent is unable to examine all cells belonging to the FOR of a node with $r = 1$. This is due to unsymmetrical positions of search map cells around the nodes and fluctuations in agent position, resulting in measurements that are wider to one side of the agent than the other. This phenomena leaves unsearched cells inside the FOR of a given node and the reward of a node might be above the threshold τ even after the node has been visited.

A solution to address the problem could be to increase the threshold τ such that the search reward of a node indeed drops below τ after visitation. However, increasing τ would also affect the overall search because other medium priority areas of the search region might drop below the threshold. To resolve the issue, a small margin is added to the agents field of view to compensate for unsymmetrical node placement in the the grid map. For this scenario configuration, the FOV-to-FOR ratio was increased from $r = 1$ to $r = 1.1$ (Figure 4.19b). The result show the agent is capable of covering all cells in the FOR of the visited nodes, without the FOR of unvisited nodes being significantly affected.

Figure 4.19c and 4.19d shows the resulting trajectories for $r = 1.5$ and $r = 2.0$ respectively. The search patterns are all variations of the growing search spiral observed in Section 4.1, but are performed in a less symmetrical manner. This lack of symmetry causes the search pattern to leave some areas unsearched. The corresponding nodes containing the unsearched area in its FOR may be assigned a reward below τ, making it disregarded for proceeding search. This may have fatal consequences as in the worst case there might be undiscovered targets inside such an unsearched region that are now unreachable from any of the remaining nodes.

Figure 4.20 shows the objective function $f(\mu, t)$ over time for the three different ratios r. As the node spacing is constant for all three scenarios, the field of view for each agent is increasing with the ratio. This means the area measured per time is proportional to the FOV-to-FOR ratio, and thus is makes sense that $f(\mu, t)$ decrease faster with higher values of r. It can be seen that when $r = 2$, $f(\mu, t)$ is decreasing the fastest. However, $r = 1.1$ actually has a higher rate of decrease than $r = 1.5$, even with a smaller FOV size for the agent. This is the result of the absence of remaining unsearchable areas due to the small margin added to $r = 1.1$, as the presence of these slow down the decrease of the objective function.



(a) $r = 1.0$. Snapshot during mission as the search did not terminate.

(b) $r = 1.1$

(c) $r = 1.5$

(d) $r = 2.0$

Figure 4.19: Resulting trajectories for four different FOV-to-FOR ratios.

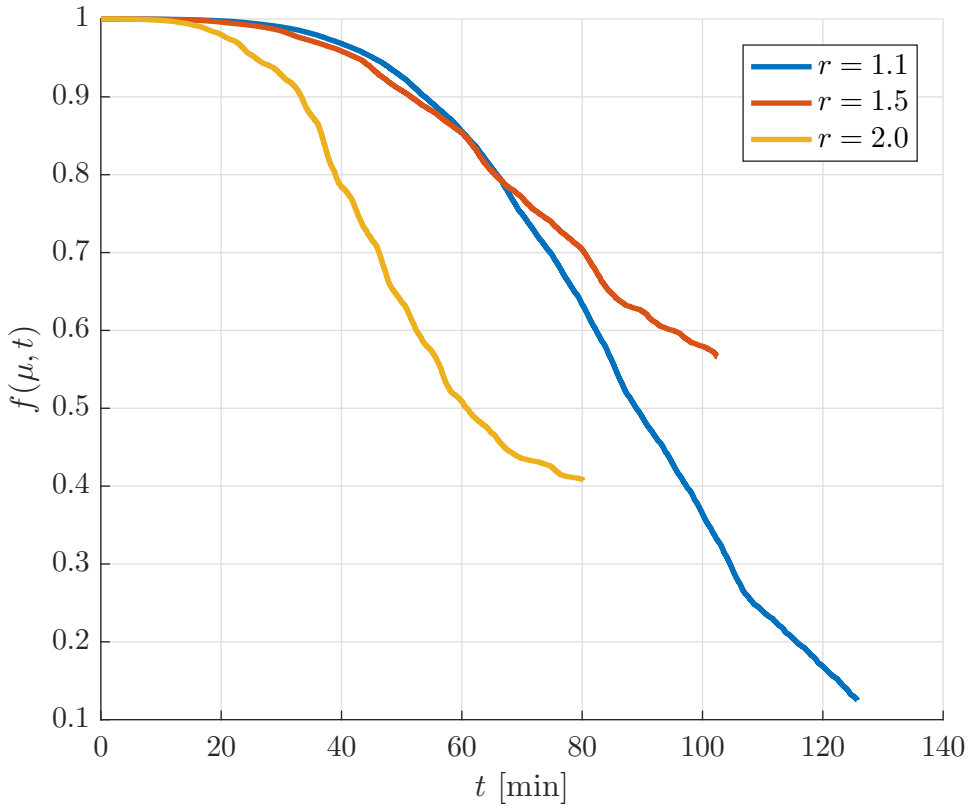


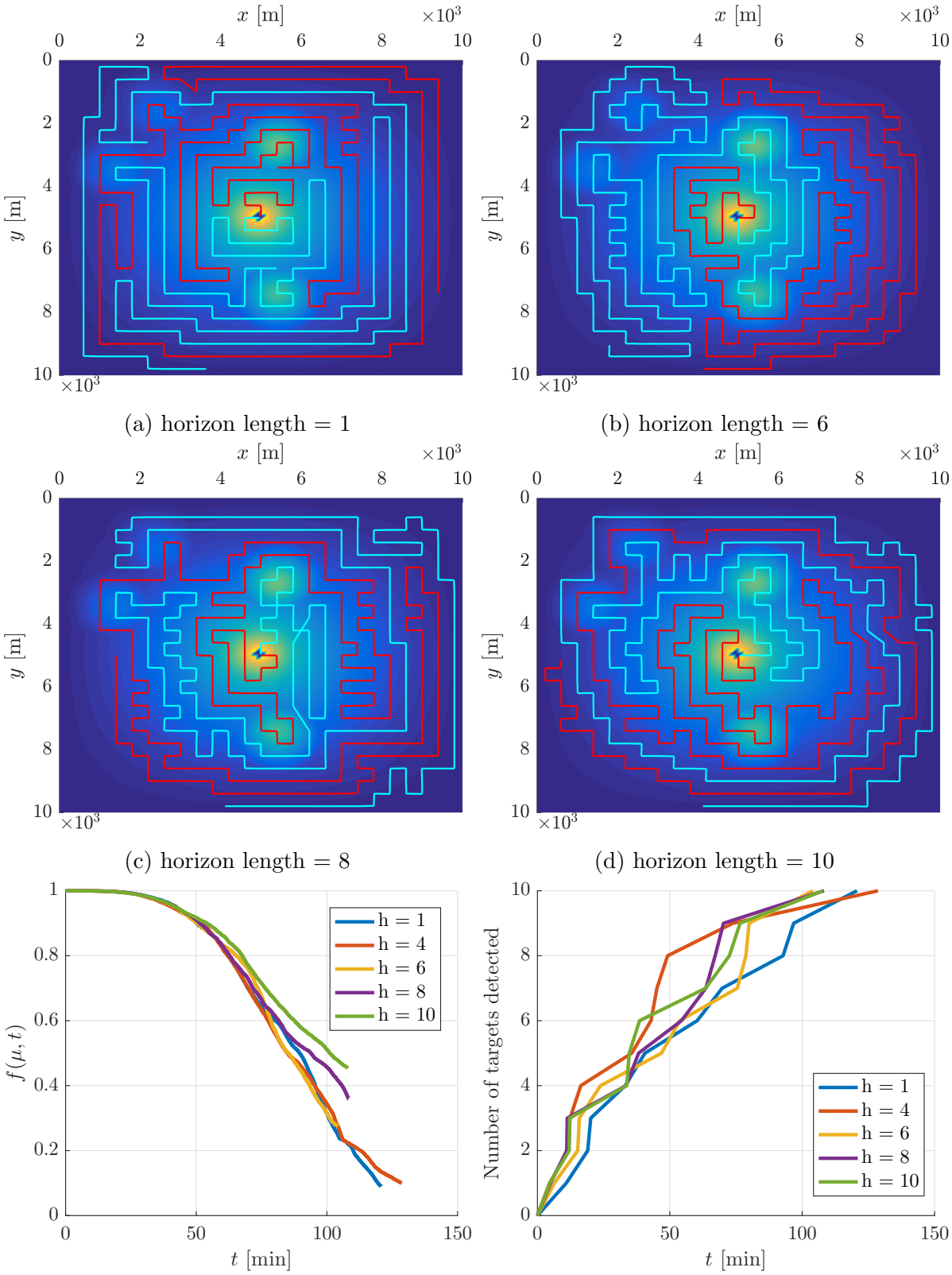
Figure 4.20: Objective function over time for different FOV-to-FOR ratios.

4.6.2 MPS Horizon Length

Another design parameter having an impact on the resulting search strategy is the horizon length used in the model predictive search algorithm. The horizon length represents how many waypoints into the future the agents are capable of considering when choosing the optimal path. By increasing the horizon length we can influence the search strategy from a greedy approach, to a more long term search strategy planning further into the future.

To examine the effects of adjusting the horizon length, a simulation scenario where prior information is given on a subset of the target positions is set up. The simulation configuration is shown in Table 4.5, and the simulation will terminate when the team has detected all ten targets. As we aim to isolate the effect of adjusting the horizon length, the number of agents deployed is irrelevant and all simulations in this section is performed for a team size of two agents. As stated in Section 3.1.1, the search for the optimal path is implemented as a depth first search, meaning the number of paths to examine is a rapidly growing function of the horizon length. Due to computational limitations, we are not able to test the case where the horizon length is equal to the total number of nodes in the search graph, but are limited to a maximum horizon length of approximately ten nodes.

Figure 4.21a - 4.21d shows the resulting trajectories for horizon lengths 1, 6, 8 and 10 respectively. Trajectories for horizon length 4 is shown in Figure A.6. A common behavior is that both agents are moving in an expanding manner from the offshore rig in all cases. As seen from the case of the short horizon length (Figure 4.21a), the agents are not immediately searching for the targets for which they have prior position information. Increasing the horizon length however, causes the agents to prioritize investigating the



(a) horizon length = 1

(b) horizon length = 6

(c) horizon length = 8

(d) horizon length = 10

(e) Objective function over time.

(f) Number of target detections over time.

Figure 4.21: Resulting agent trajectories for various horizon lengths in addition to objective function and target detection over time.

Table 4.5: Configuration for scenario with different MPS horizon lengths.

Parameter	**Value**
Physical Size	5,000m × 5,000m
Number of grid cells	100 × 100
Number of nodes	25 × 25
MPS horizon lengths	$\begin{bmatrix}1, & 4, & 6, & 8, & 10\end{bmatrix}$
Graph threshold	10^{-4}
Agent velocity	15m/s
Number of targets	10
Target velocities	None
FOV to FOR ratio	1.1

prior information areas at an earlier stage, as this will cause a faster decrease of the objective function.

Figure 4.21e and 4.21f shows the objective function over time and the number of detected targets versus time for all horizon lengths. The objective function is not affected much by the increased the horizon length. The rate of target detections is also similar for all horizon lengths. However, Figure 4.21f shows the horizon length equal to one marginally has the slowest rate of detection, while surprisingly the horizon length equal to four has to steepest detection rate. It is however important to note that these results are specific for this single scenario, thus no general conclusions should be drawn based on this data set.

The simulated mission time and the computer run time for completing the simulations are displayed in Figure 4.22. The simulated mission time is not much affected by the increased horizon length. However, passing a horizon length of eight greatly increase the computer runtime and at a horizon length of 10, the computer runtime is almost twice the length of the simulated mission time. This leaves the centralized computations useless for practical applications as they would require online computations.

Some improvements could be done to reduce the algorithm runtime, for instance implementing the optimization in a more seamless language with less overhead such as C or Fortran. Another suggestion is simply running the centralized software on a more powerful computer, to be able to increase the horizon length. However, the above results does not imply an extremely high horizon length necessarily results in a more efficient agent behavior.

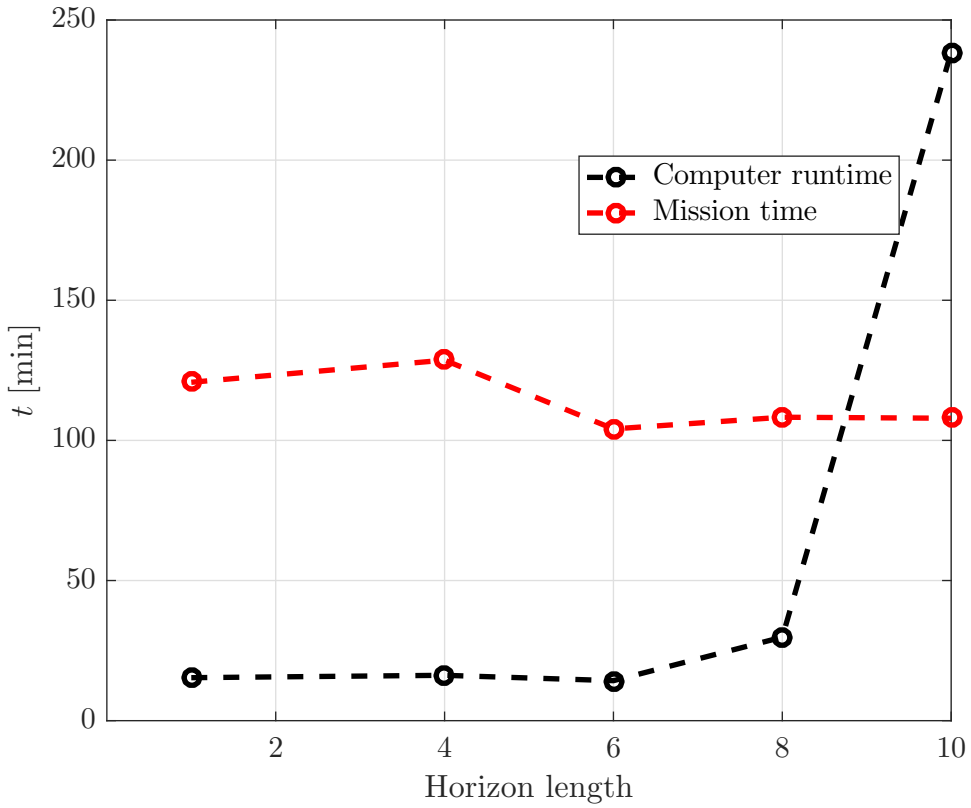


Figure 4.22: Simulated mission times, and actual computer runtime to complete a simulation. Simulations performed on an Intel® Core™ i7-4790 3.60GHz processor, 16.0 GB RAM.

4.6.3 Kernel Estimation Accuracy

Section 2.2.2 explains how the predictive step includes map dynamics as a convolution between the prior belief and the estimated cell transition probability kernel. As the nodes of the search graph only appear in areas with a satisfactory search reward, the accuracy of the estimated kernel is important to ensure targets are reachable from nodes in the search graph at all times.

In this section, we will assume the target velocities are random vectors drawn from a Gaussian distribution with *known* expected value μ and covariance matrix Σ. A kernel will be created by sampling the Gaussian distribution into grid cells with equal size as shown in Figure 4.23. It is of interest to see how accurate the sampling process of the kernel is due to rounding errors and how these may propagate through the prediction steps. For this test scenario, a target is initially placed at position $(100, 100)$ and will every iteration follow the random velocity vector V where

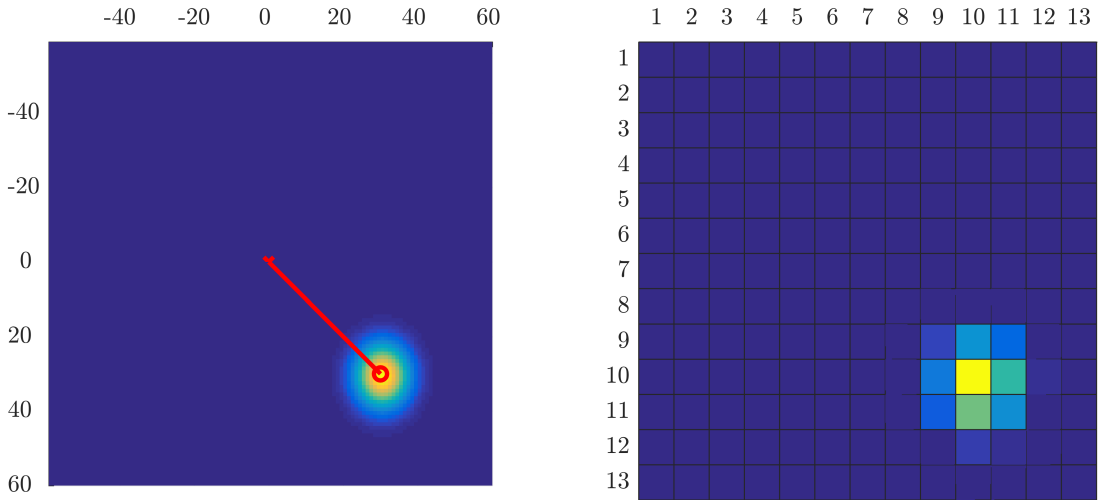
$$V \sim \mathcal{N}(\mu, \Sigma), \quad \mu = \begin{bmatrix} 30 \\ 30 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 22.5 & 0 \\ 0 & 31.5 \end{bmatrix} \tag{4.4}$$

The remaining search scenario parameters are shown in Table 4.6. In this scenario, the targets will move with a mean velocity of 2m/s, so a Gaussian distribution *could* be created with an expected value equal to this. However, as the map grid cells are of size 10×10 m^2, the resulting discretization would be too coarse due to the resolution of the kernel. Instead, the Gaussian is scaled such that we look at the position uncertainty after the targets have moved 30m in both x and y direction, which implies a predictive step is only to be employed every 15 seconds. Note that the distribution presented in (4.4) is *not* estimated from a realistic data set. The parameters are set arbitrary as we only wish to illustrate the challenges with the kernel estimation.

Table 4.6: Scenario Configuration for Kernel estimation.

Parameter	**Value**
Physical size	1,000m × 1,000m
Number of cells	100 × 100
Number of convolutions	20

The belief grid map will correspondingly be convolved by the kernel at every iteration, and the position of the target will be compared to the estimated belief map after 20 iterations to see the effects in terms of estimation error of the final target position due to kernel estimation errors. To get a satisfactory basis for comparison, Monte Carlo simulations are performed for 100 targets. Figure 4.24 shows the resulting Monte Carlo simulations at five different stages in the series of convolutions. Naturally, it can be seen that the deviations between the mean of the target position and target position belief are increasing with the number of convolutions, even when the kernel is sampled from the target velocity distribution. The source of the deviation error is the loss of accuracy through the down sampling (Figure 4.23). Higher resolution in terms of more map grid cells yields less loss in the process of down sampling. These errors becomes more significant as the numbers of convolutions increase and the errors are propagated. In addition, it should be clear from Figure 4.24 that a kernel sampled or estimated from an unknown or less accurate distribution, as is the case for a real life application, would result in additional prediction errors.



(a) Known target velocity distribution. (b) Sampled kernel from distribution in (a).

Figure 4.23: Kernel sampled from a Gaussian distribution.

Figure 4.25 illustrates how the belief of the expected target cell evolves through the predictive iterations. The mean target position after every iteration is calculated, and the belief in the corresponding grid cell is plotted against the iteration number. The expected target belief, which is a function of both uncertainty in the vector distribution and the estimation accuracy errors, is exponentially decreasing. This gives an indication that the predictive step decays the information about the target position estimation rapidly. However, as long as the target distribution is ensured to be inside an area of sufficient belief, the region containing the target will get a reward above the threshold τ should it venture into a critical region of the collision risk map.

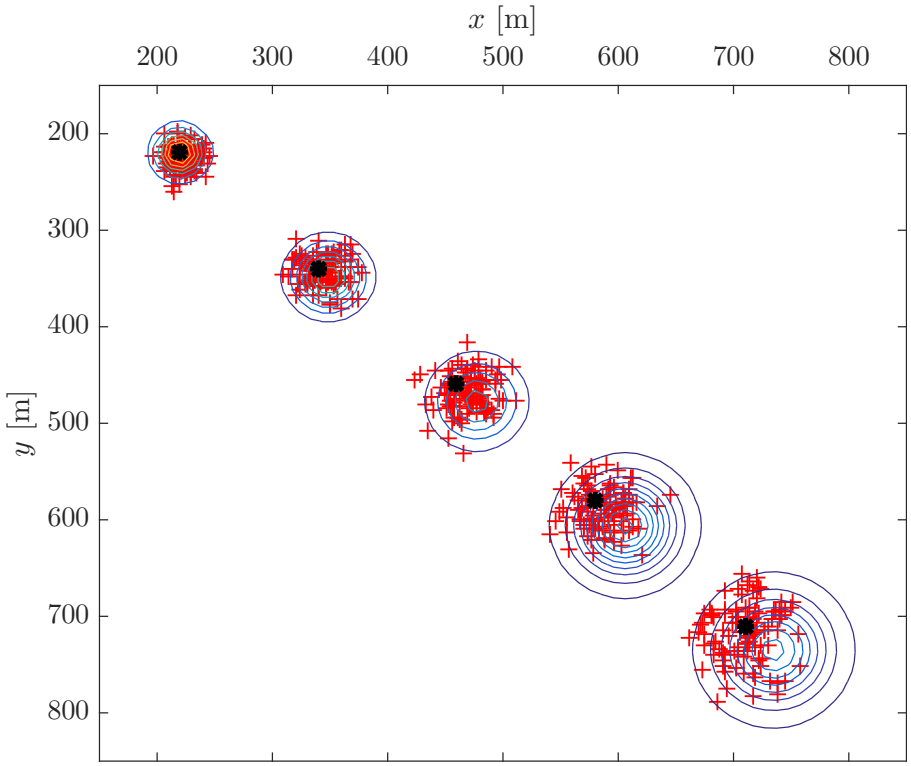


Figure 4.24: Resulting Monte Carlo simulation and belief predictions after 4, 8, 12, 16 and 20 convolutions. The red markers are the target positions and the blue lines are contour plots of the resulting predicted belief. The black markers are the mean positions of the targets.

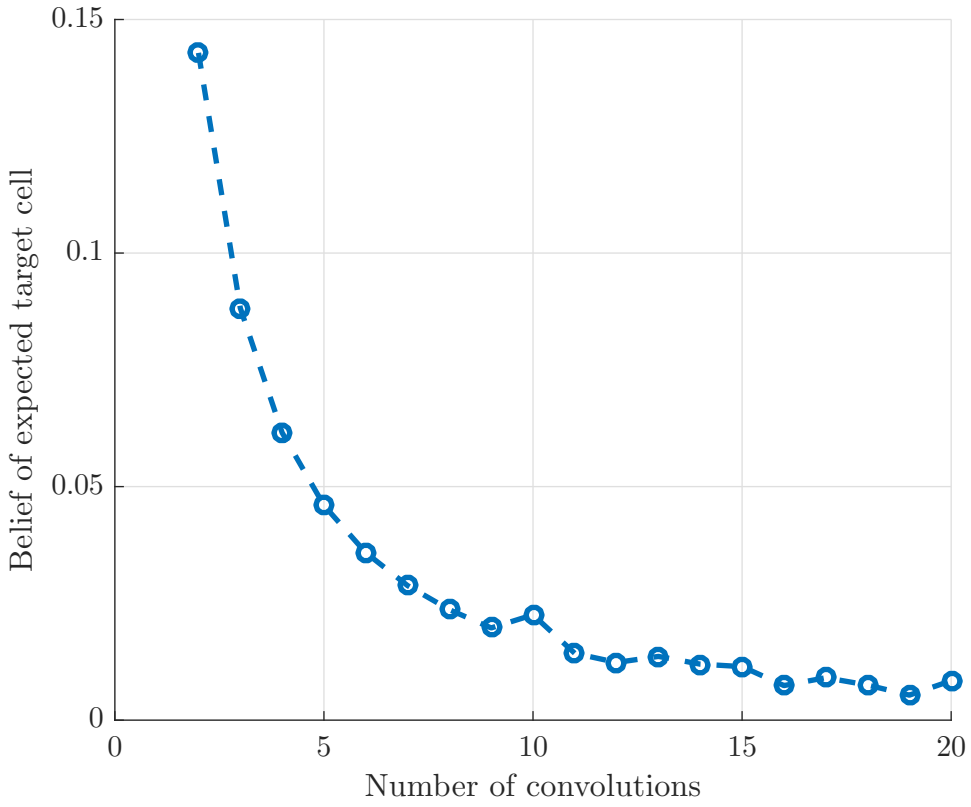


Figure 4.25: Kernel belief.

Chapter 5

Discussion

5.1 Performance, Optimality and Scalability

The system developed in this theses emphasize the focus of using multiple UAVs in ice management to increase the efficiency from using only one single UAV. In [9], the CGBMPS algorithm was employed with the intention of locating a single stationary target. The repurposing of the algorithm to account for an unknown number of targets was tested in Section 4.1, 4.2 and 4.3 for simulations of a stationary target topology with a varying degree of prior information. The results substantiates that the efficiency in terms of target detection rate and reduction in probability of collision indeed increase with the number of agents. Furthermore, efficiency in terms of search duration seems to be dependent on the amount of prior information supplied. When no prior target information is given, the mission durations have an exponential decrease. On the other hand, when prior information is given for all target positions, the same rate of decrease does not occur. The system's ability to utilize the a priori knowledge seems to be efficient as an increase in prior information leads to a reduction in mission duration.

Current ice management applications are often relying on satellite data to estimate the position of threatening sea ice. A comparison between the usage of the presented system and the usage of satellite imaging was performed in Section 4.4. The two systems are highly complementary as the satellite measurements have a low update frequency but a high level of data acquisition in terms of measured area, while the UAS has a relatively high update frequency but a low level of data acquisition compared to the satellite imaging. A comparison was made based on the objective function for the two systems. However, due the different modes of operation it is difficult to compare the two alternatives. Moreover, the search objective function is not an objective metric of the current danger posed by the surrounding sea ice, as it heavily depends on the gathered sensor measurements, thus making it hard to conclude which of the two alternatives are better suited for ice management. On the other hand, as the two systems are complementary, a combined system would be beneficial by combining the strengths of both alternatives. As the satellites have a high level of data acquisition, satellite imaging could be incorporated in the Bayes' estimation update of the UAS, even when the update rate is infrequent.

The introduction of a priority function δ dependent on the belief of a cell, introduces a notion of tracking to the the presented system. The results from Section 4.5 shows that the tracking ability is limited when the number of agents is small compared to the number of targets to track. However, increased tracking performance is achieved when the

number of agents is increased. A weakness of the tracking feature is the rapid oscillations in the objective function, as the estimated target positions quickly gain uncertainty. This weakness is a result of a sensitive priority function δ yielding a great increase from just a small decrease in the belief of a given cell. This ensures the agents are putting a high emphasis on tracking, which obviously reduce the time spent searching for new targets, raising the question whether the Shannon entropy is a good choice of δ after all. However, the advantage of the priority function is that it can be used as a tuning instrument for whether the user wants emphasis on searching or tracking, suggesting that δ should be tailored to each individual use case.

There are a few parameters in the implemented system architecture that require tuning for a satisfactory behavior. In Section 4.6.1 it is shown that given the size of the field of view of the agents, the node spacing is a sensitive parameter in terms of variations in the resulting search patterns. It is made clear that this parameter should be chosen such that a good coverage of a node's field of region is ensured when the node is visited. A proper node spacing will also affect the decrease in the objective function, and a poorly chosen node spacing may result in a search not terminating properly, as visited nodes may not be removed and the agents could get stuck in loops.

Section 4.6.2 presents the effects of changing the MPS horizon length, and it surprisingly turns out that increasing the horizon length does not necessarily result in a significant improvement of the search efficiency, as revealed by the objective function plots. However, the simulated horizon lengths ranging from 1-10 might be too short to see a clear difference, and the influence of this parameter could perhaps be seen for a horizon length approaching the total number of nodes in the search graph. The simulations performed in this thesis were limited by computational power and it is seen that for a horizon length of 10, the computer runtime is almost twice that of the actual mission time which would not be applicable for practical applications.

When calculating new agent paths in the model predictive search, each agent is assigned the path that will maximize the team's ability to reduce the chances of collision. However, new paths are only calculated every time an agent reachs its designated waypoint, and in the worst case scenario the entire team reach a waypoint at the same time. The resulting problem of finding the set of optimal paths for the entire team simultaneously has been simplified in this thesis to the sequential process of finding the optimal path for one agent at a time. Although this might result in a suboptimal solution for the team, it is much less computationally demanding.

To investigate the real time performance of the system when integrated in a fully operational framework, SITL simulations have been performed. The implementation of the software with the intention of SITL testing has made the implementation process significantly more time demanding and challenging, but it has ensured the system was implemented with an abstraction making it applicable to actual field testing. Thus, the SITL simulations can be viewed as a proof of concept of the software itself, with the end goal being an operational system. The fact that SITL tests have been successfully conducted, implies the system developed in this thesis is ready for field testing from a software perspective.

5.2 Dynamic Occupancy Grid Maps

In this thesis, the approach to a dynamic occupancy grid map has been to represent the dynamics as transition probabilities between individual grid cells. This representation is based on the discretization of the probability density function of the target velocity vector. In order to preserve the total probability after the predictive step in the Bayesian filter, the probability density function of the target velocity vector is assumed to incorporate the target dynamics for the entire search region. Although it is outside the scope of this thesis to use information like metaocean data, weather forecasts or historical target trajectories to analytically estimate this probability density, one can easily spot the weakness of generalizing one single distribution for the entire search region. This might be a good approach for small search regions with a low diversity in target dynamics, but once the search region grows large, the dynamics of a target might vary depending on the geographical position. Of course, this can be accounted for by increasing the uncertainty of the general target dynamics probability density function, but at some point the uncertainty might grow to a point where the predictive step will not add any useful information. Thus, using another approach calculating the dynamics depending on the spatial position in the search region, arguably would make more sense for many search scenarios. In the case of ice management however, the size and shape of the targets highly influence their trajectories which could make it an advantage having a generalized model for the entire search region. Additionally, by using an alternative approach to the kernel $\mathcal{K}_t$ in order to encapsulate the movement of the targets, the collision risk map algorithm proposed in Section 2.3.1 should also be reconsidered due to the fact that its creation relies heavily on the elements of $\mathcal{K}_t$.

Chapter 6

Conclusions and Future Work

This thesis presents the development of a cooperative search algorithm for detecting sea ice to aid ice management in offshore operations. By representing the search region as an occupancy grid map, the unmanned agents of the system use camera measurements together with their pose to iteratively update the estimate of target locations in the grid map. The qualitative ice management problem was formulated as a quantitative mathematical optimization problem where the agents minimize the probability of collision given the measurements taken. In order to account for dynamics in the search region, initial assumptions of independent grid cells were relaxed for the calculation of a predictive step in the iterative estimation process. An objective function to the optimization problem was proposed based on the estimates of target positions in combination with the estimated target dynamics used in the predictive step. This function was based on the objective of reducing the chances of collision from potential threats, and paths for the a team of agents are chosen accordingly by optimization. The proposed objective function removes the need for separating the two modes, searching and tracking. The optimal paths are chosen to maximize data acquisition according to the priority function δ, leading to a floating transition between searching and tracking. A Cooperative Graph Based Model Predictive Search framework using a waypoint based receding horizon allows for easy comparison between paths of different length by doing a coarser search in regions deemed unimportant. The system has been implemented for SITL (Software In The Loop) testing, to ensure the software performs well in real time when integrated in a fully operational setting. However, results from section 4.6.2 shows that the horizon length should be limited for the computation time to be feasible for online calculation.

An advantage of the presented system is that it can deal with an unknown number of targets with either some notion of prior target information or no prior information at all. Even though the system works well for one single agent, simulations show a significant increase in performance by increasing the size of the search party, both with regards to search time and threat detection. Furthermore, we see a great benefit from any prior target information supplied, indicating that combining the system with information gathered from satellite images is beneficial. However, simulations of the occupancy grid map dynamics reveal that the predictive step is sensitive to errors in the estimation of the target dynamics. This will prove a problem for large search regions and situations heavily relying on predictive updates.

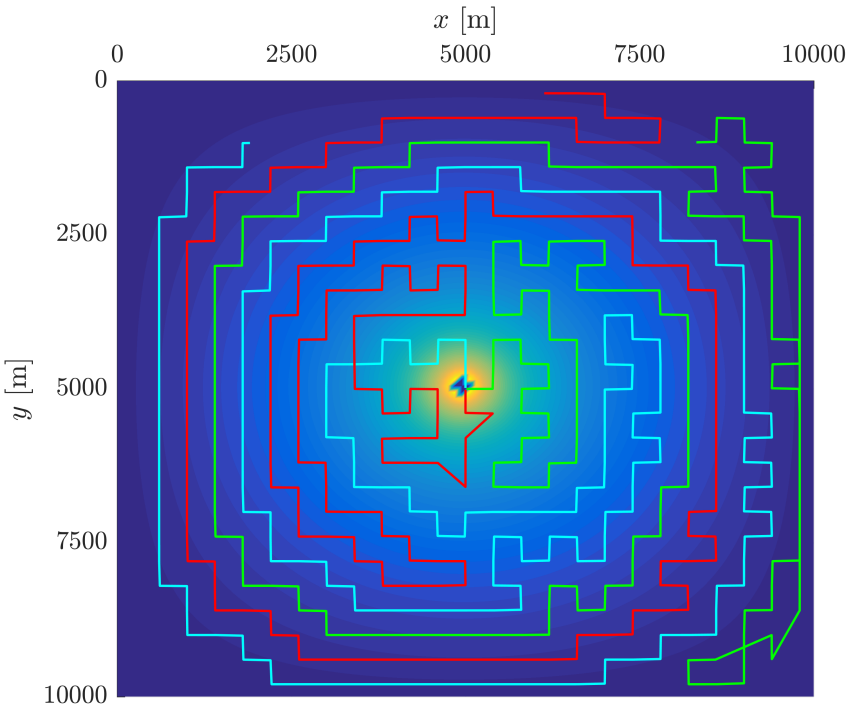
The focus of this thesis have not been on direct methods of employing metaocean data for calculating map dynamics. Future work would involve finding analytical methods to

estimate the cell transitional probabilities from these data, and even investigate other approaches to map dynamics altogether. In this thesis, a priority function calculating some notion of priority from the posterior belief of grid cells is proposed. Even though using the Shannon entropy of the belief yielded a desired effect, it proved slightly too sensitive to changes in the belief. The next step would involve a systematic approach to find a priority function perfectly suited for the use case. Another possibility is having an adaptive priority function, adjusting as a search unfolds and search motives change. Furthermore, a key assumption in this thesis is that the pose of the agents are perfectly known at all times, while in reality errors in the pose estimation will effect the mapping process. Future work should include implementing the system with agents capable of on-board image processing, investigating the performance of the system faced with errors caused by both pose estimation and camera calibration.

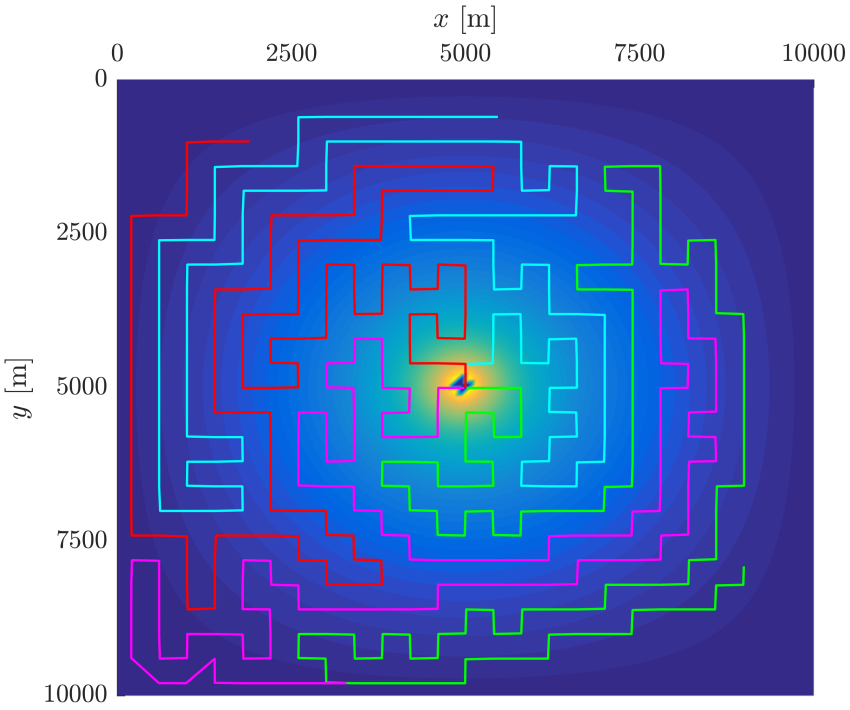
Appendix A

Additional Simulation Plots

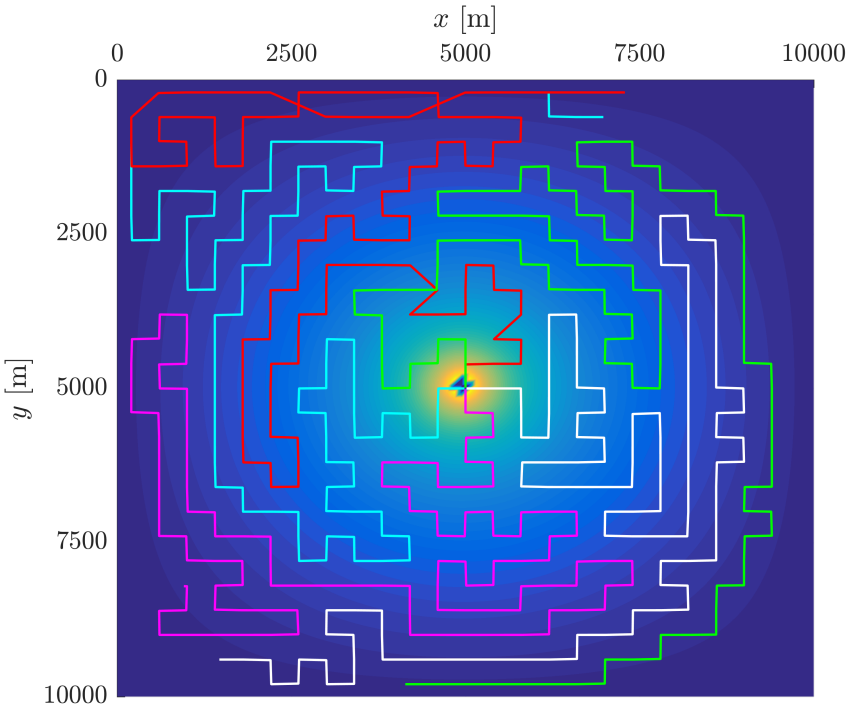
A.1 No Prior Target Information



(a) Number of agents = 3.



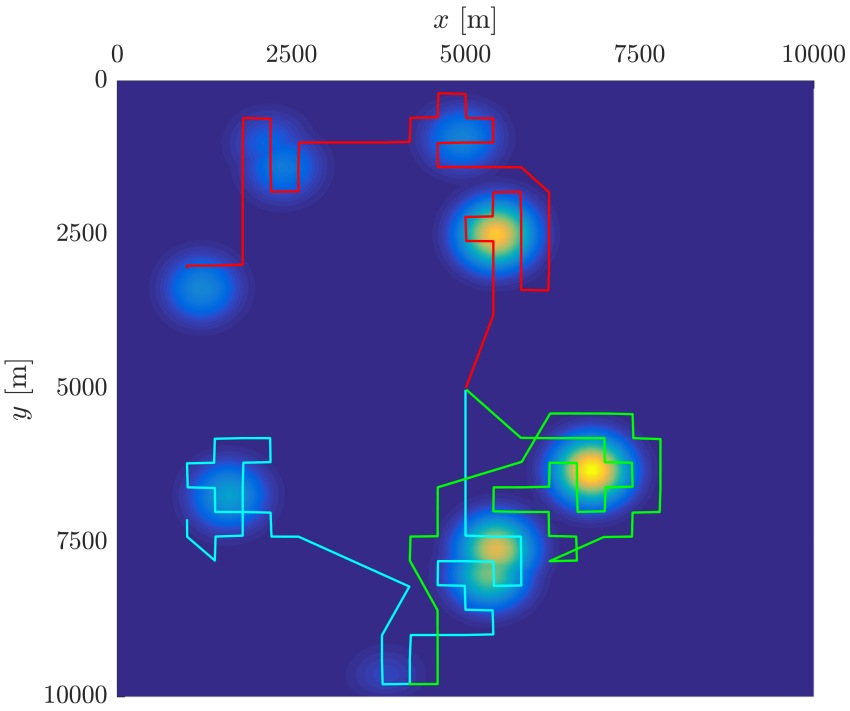
(b) Number of agents = 4.



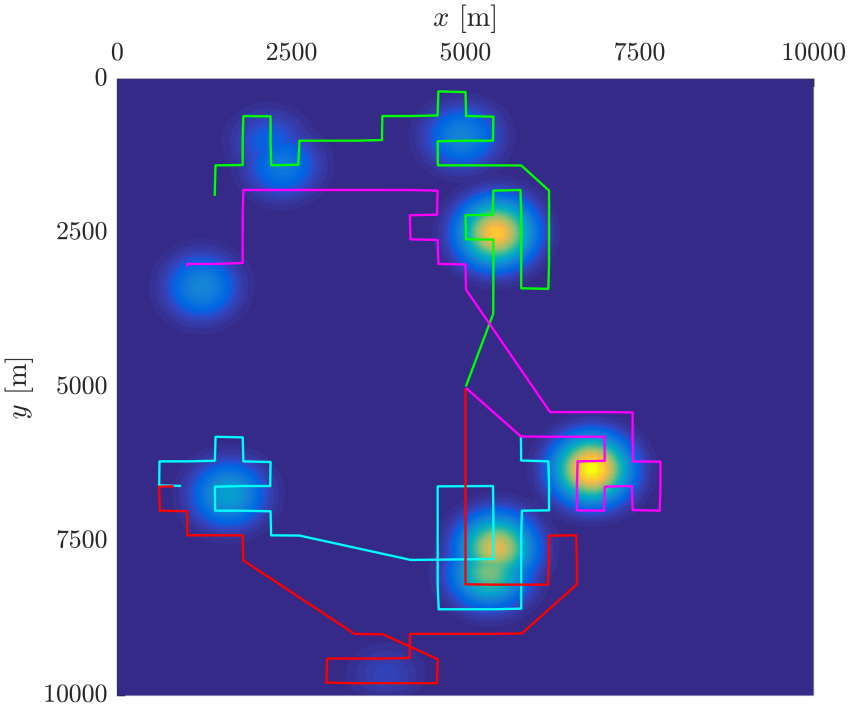
(c) Number of agents = 5.

Figure A.1: Agents searching for targets where no prior target information is given for any targets.

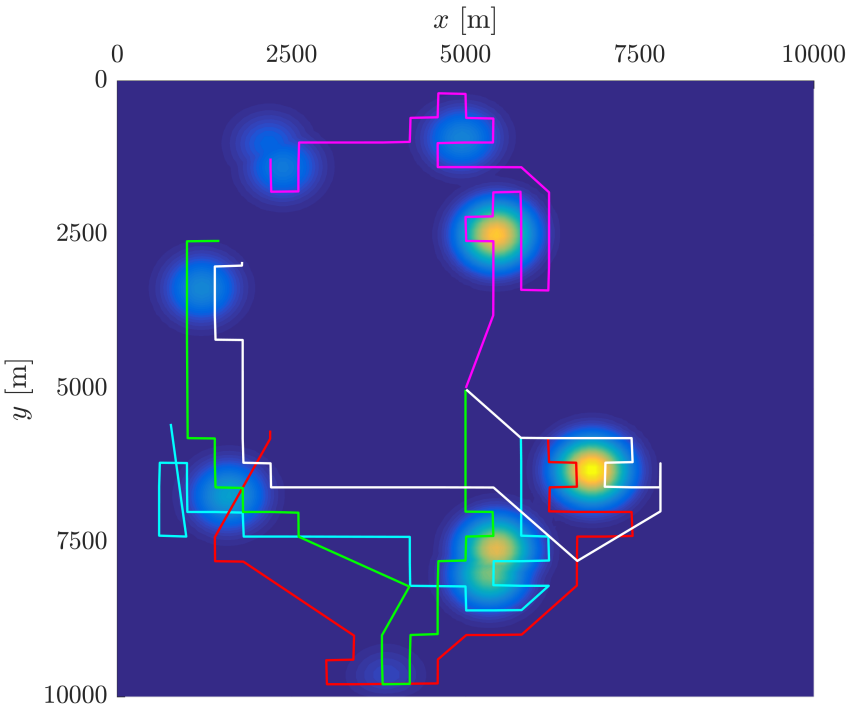
A.2 Prior Information on all Targets



(a) Number of agents = 3.



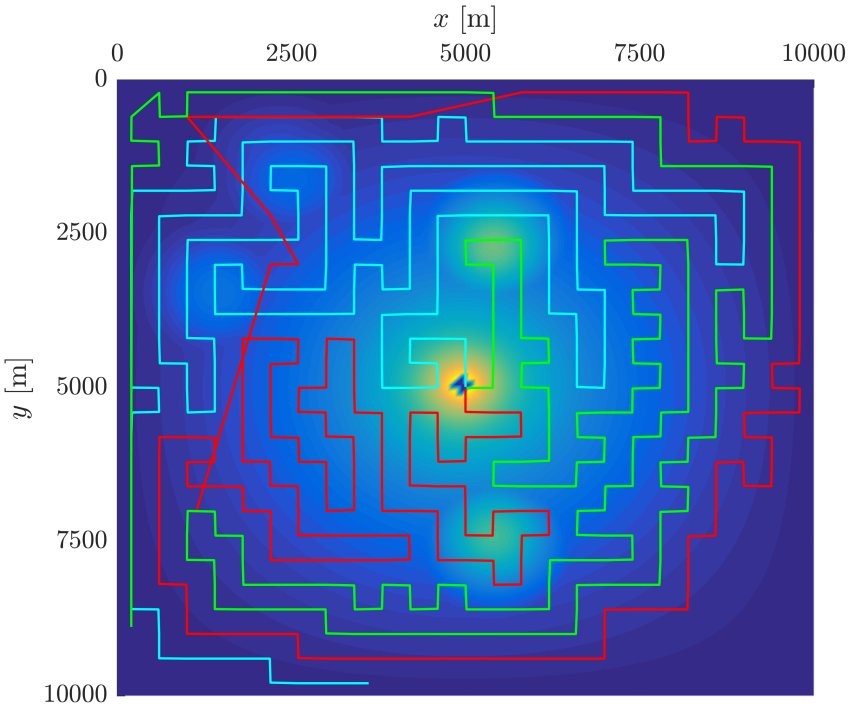
(b) Number of agents = 4.



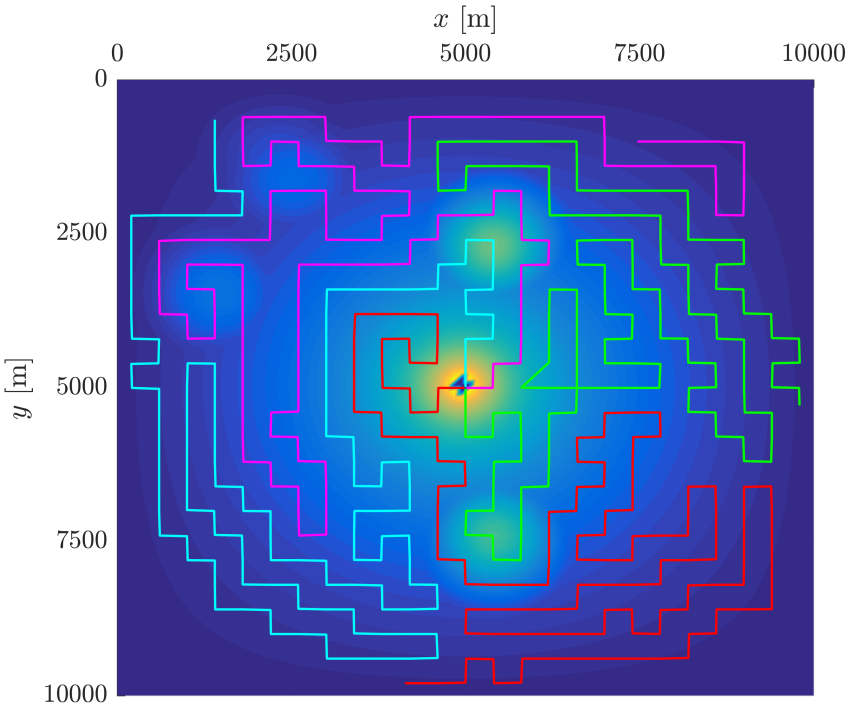
(c) Number of agents = 5.

Figure A.2: Agents searching for targets where prior information is given on all targets.

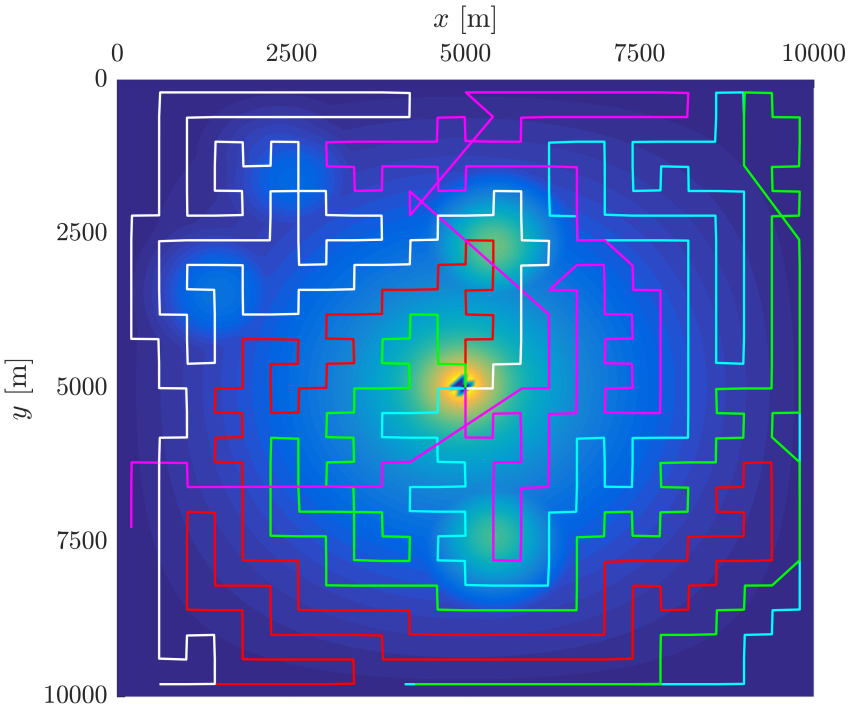
A.3 Prior Information on a Subset of the Targets



(a) Number of agents = 3.



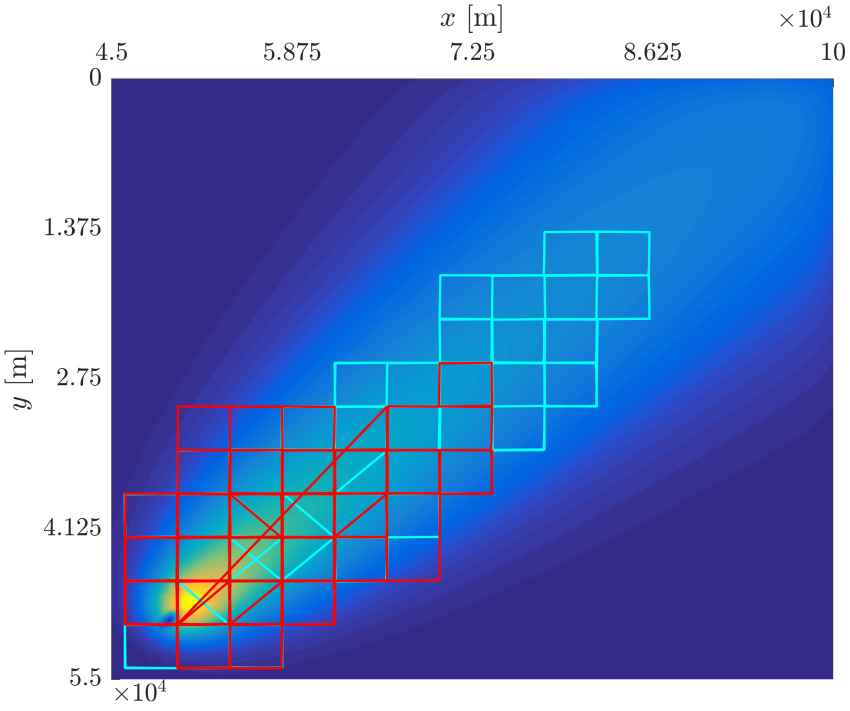
(b) Number of agents = 4.



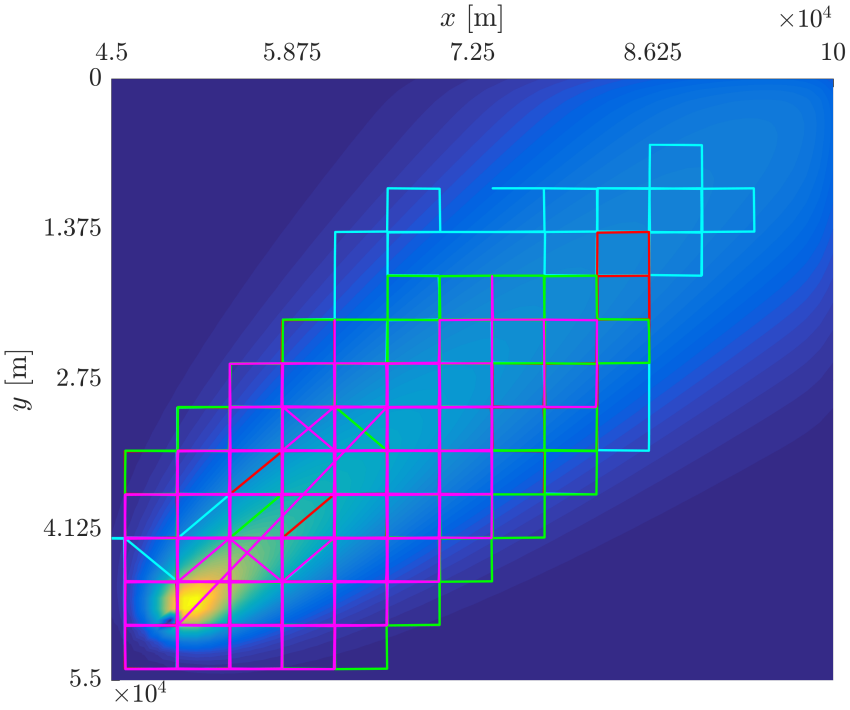
(c) Number of agents = 5.

Figure A.3: Agents searching for targets where prior target information is given for a subset of the targets.

A.4 Team of Agents Compared to Satellite Imagery



(a) Path logs for two agents.



(b) Path logs for four agents.

Figure A.4: Path logs for different team sizes when the agents are searching for 72 hours.

A.5 Tracking Performance

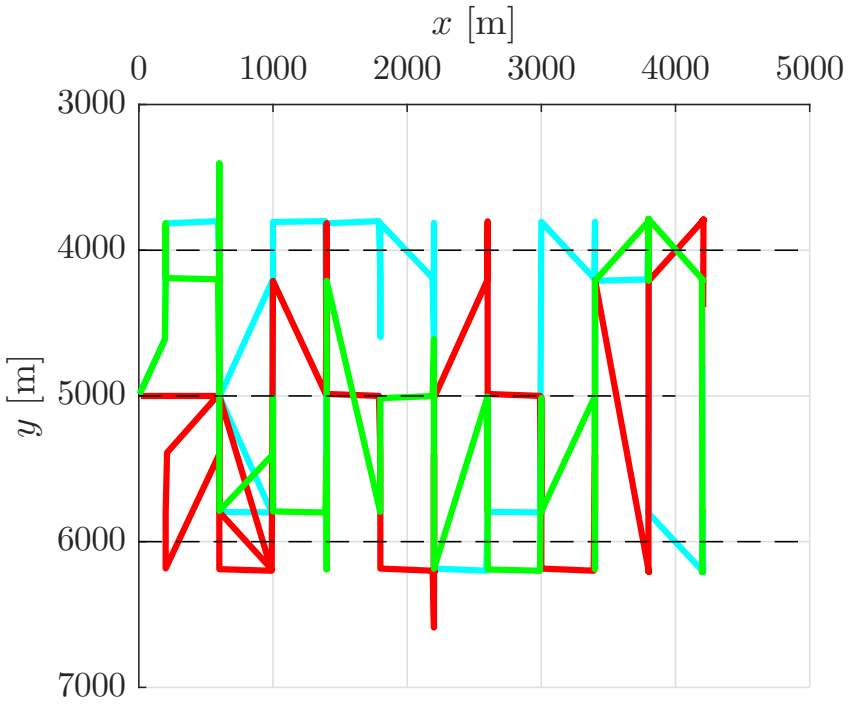


Figure A.5: Trajectories of team consisting of three agents tracking three targets moving towards the offshore rig.

A.6 MPS Horizon Length

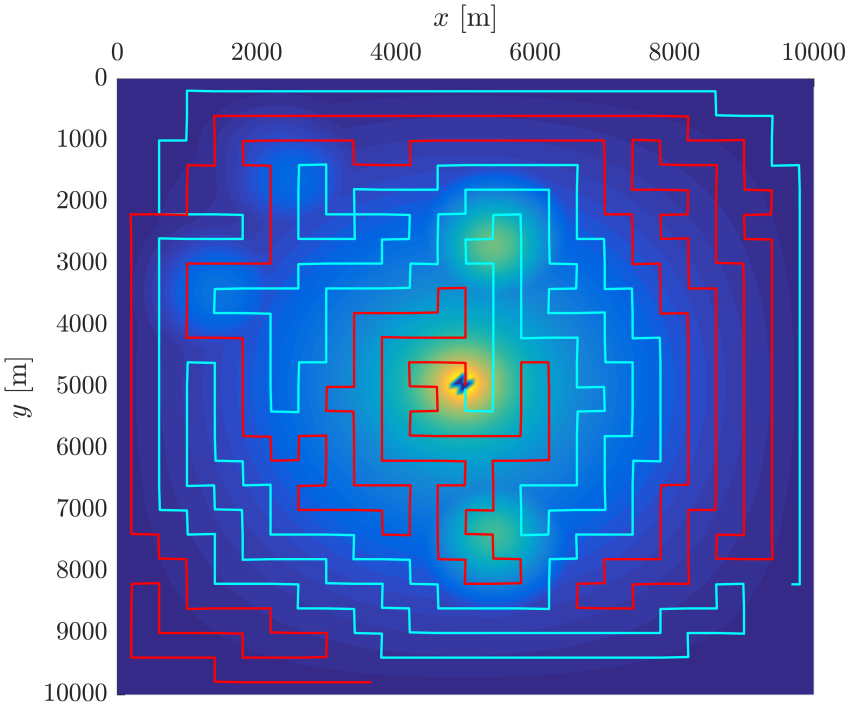


Figure A.6: Resulting agent trajectories with horizon length = 4.

Bibliography

[1] M. S. Arlampalam et al. *A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking*. 2002.

[2] J. S. Berndt. *JSBSim Reference Manual*. 2011. URL: `http://jsbsim.sourceforge.net/JSBSimReferenceManual.pdf`.

[3] P. Chandler and M. Pachter. *Hierarchical Control for Autonomous Teams*. 2001.

[4] B. DasGupta et al. "Honey-pot constrained searching with local sensory information." In: (2006).

[5] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics, 2nd edition*. 2010.

[6] K. Eik. "Review of Experiences within Ice and Iceberg Management." In: *Journal of Navigation 61.04* (2008).

[7] A. L. Flåten. "Experimental Monitoring of Sea Ice Using Unmanned Aerial Systems." In: (2015). URL: `https://brage.bibsys.no/xmlui/handle/11250/2352540`.

[8] R. C. Gonzales and R. E. Woods. *Digital Image Processing*. 2008.

[9] Gaemus E. Collins James R. Riehl and Joao P. Hespanha. "Cooperative Search by UAV Teams: A Model Predictive Approach Using Dynamic Graphs." In: (2011).

[10] B. O. Koopman. *Search and Screening. Operations Evaluations Group Report No. 56, Center for Naval Analyses, Alexandria, VA*. 1946.

[11] R. Martins et al. "IMC: A communication protocol for networked vehicles and sensors." In: *OCEANS 2009-EUROPE. IEEE* (2009).

[12] L. H. Nunn. "An Introduction to the Literature of Search Theory." In: (1981).

[13] M. Otto et al. "Uncertain 2D Vector Field Topology." In: *Eurographics* (2010).

[14] J. Pinto et al. "Implementation of a control architecture for networked vehicle systems." In: *Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles* (2012).

[15] J. Pinto et al. "Neptus: a framework to support a mission life cycle." In: *Proc. IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)* (2006).

[16] S. M. Pollock. *A Simple Model of Search for a Moving Target*. 1970.

[17] M. Skjønhaug and M. Hals. "Optimization of Coordinated Control Between Autonomous Vehicles in Ice Management." In: (2016).

[18] ArduPilot dev. team. *ArduPilot: Open Source Autopilot*. URL: http://ardupilot.org/.

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. 2002.