NTNU

Norwegian University of
Science and Technology

Unmanned Aerial Vehicle Mission Planning for Combined Iceberg Detection and Tracking Missions

Sindre Matthijs Langeveld

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

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

Unmanned Aerial Vehicles (UAVs) can be used for operations concerning ice management in Arctic environments. An example is oil drilling operations, where the oil rig or vessel has major interests when it comes to locating, and tracking ice that could be a potential threat to the operation.

Ice management is traditionally divided into two separate tasks: Search and track. The notion of tracking captures path planning based on already discovered icebergs, where the main challenge is to create accurate stochastic models of the dynamics of the ice, and to use this to decide the UAV's path. Search on the other hand, covers the process of discovering unknown ice, where the main issue is to decide where to search. For certain applications, it could be beneficial to combine these tasks into a single algorithm. In this master thesis, the possibilities of doing this will be studied, and a possible solution will be presented, implemented and simulated.

More specific, the following tasks are to be done:

1. Survey the literature and propose a system architecture where both iceberg search and tracking are encapsulated.
2. Demonstrate how the computation of the optimal path can be modelled as a Travelling Salesman Problem.
3. Create an iceberg drift model which can be used to simulate iceberg drift. Use real data from wind and ocean current to make the trajectories more realistic.
4. Develop a simulator to test the combined search and track algorithm. Include the iceberg drift model to generate test objects.
5. Simulate the system to test its performance. Analyse its robustness by simulating under different ice and weather conditions.
6. Investigate the possibilities of including radar and satellite data to the system.

Preface

This master thesis is my final work after five years at the MCs programme in Engineering Cybernetics and Robotics at NTNU.

I would like to thank my supervisor Tor Arne Johansen for his guidance and help, and for letting me work with this project. I would also like to thank my co-supervisor Frederik Stendahl Leira for giving me great advice both regarding research, and when it comes to writing a thesis. A special thanks goes to my co-supervisor Leif Erik Andersson, who has helped me understand some of the mysteries behind iceberg drift.

Finally, I would like to thank my fellow students at "Kontoret" for providing a great social and working environment.

Sindre Matthijs Langeveld
Trondheim, June 6, 2017

Abstract

In this master thesis, iceberg detection and tracking by Unmanned Aerial Vehicles has been studied. Currently, most systems for ice management separate these two tasks. However, combining these tasks into a single algorithm might only require one UAV, which could reduce the operating costs. In this thesis, we propose a combined search and track algorithm and investigate its performance and viability.

To test the algorithm, a simulator has been developed. A Kalman filter has been used to keep track of detected icebergs. Data from wind and ocean currents have been used to determine critical areas that should be searched. Optimal paths for combined search and track missions for an Unmanned Aerial Vehicle (UAV), have been computed by solving a Travelling Salesman Problem. To create a more realistic simulation environment, an iceberg drift model has been developed. By collecting real wind and ocean current data from the test area, the model generates dynamic icebergs, which can be used to test the algorithm. A series of simulations were done to test the algorithm's performance and robustness.

The results indicated that the combined algorithm had the desired behaviour, in the sense that both search and tracking were prioritized in the UAV missions. Further, we found that an iceberg that is being tracked, should be rediscovered as often as possible, in order to optimize the tracking performance. Other tests indicated that the total search radius was decreased when the ice density was increased. The algorithm showed acceptable performance up to an ice density of 50 icebergs. Considering the decreasing search radius, it was concluded that ice densities above this quantity, could lead to an inadequate detection of icebergs. The algorithm's robustness was also tested in a more rapid changing weather condition. This resulted in a degraded total performance, but it was concluded that the likelihood of weather changing this fast and random, was low. The system showed good flexibility when including radar and satellite data. For the radar case, a safer system was obtained. For the satellite case, results indicated that the performance was improved as long as the iceberg classification accuracy was above 75%.

The results of this study support the viability of a combined search and track algorithm for UAV search and track missions. However, some issues have still to be solved, for instance when it comes to performance in rapidly changing weather. Future work should therefore focus on making the algorithm more robust, before performing experiments with a UAV for proof of concept.

Sammendrag

I denne masteroppgaven har det blitt sett nærmere på søk og sporing av isfjell, ved hjelp av ubemannede fly (UAV). De fleste av dagens systemer for ishåndtering, deler opp de to operasjonene i ulike systemer. Ved å kombinere disse operasjonene i en og samme algoritme, er det gode muligheter for å klare seg med én enkelt UAV, noe som kan redusere de totale operasjonkostnadene. I denne oppgaven foreslår vi en kombinert søk- og sporingsalgoritme, og undersøker dens ytelse og levedyktighet.

Det er blitt utviklet en simulator for å teste algoritmen. Et Kalman filter ble brukt til å spore oppdagede isfjell. Data fra vind og havstrømninger ble benyttet for å bestemme kritiske områder det bør søkes i. Optimale rutevalg for kombinerte søk- og sporingsoppdrag for en UAV, har blitt bestemt ved å løse et Travelling Salesman Problem. For å skape et mer realistisk simuleringsmiljø utviklet vi en en modell som tar for seg isfjellbevegelse. Ved å samle inn reele data i form av vind og havstrømninger fra testområdet, kan modellen brukes til å generere bevegelige isfjell, som kan brukes til å teste algoritmen. En rekke simuleringer har blitt gjort for å teste algoritmens ytelse og robusthet.

Resultatene indikerte at algoritmen har en ønskelig oppførselen, i den forstand at både søk og sporing ble prioritert i UAV-oppdragene. Videre fant vi at isfjell som blir sporet, bør bli gjennoppdaget så ofte som mulig, for å optimalisere sporingsytelsen. Gjennom videre testing fant vi at søkeradien minket, når istettheten økte. Algoritmen viste akseptabel oppførsel opp til en istetthet på 50 isfjell. Med tanke på den minkende søkeradien, ble det konkludert med at en istetthet over dette antallet, vil kunne føre til at isfjell ikke blir oppdaget før det er for sent. Algoritmens robusthet ble også testet i et miljø hvor været endret seg mye raskere enn før. Resultatene indikerte at den totale ytelsen ble redusert, men det ble konkludert med at sannsynligheten for at været endrer seg så ofte og tilfeldig, er liten. Systemet viste god fleksibiliet når radar- og satellittdata ble inkludert. For tilfellet med radar fikk vi et tryggere system. Når vi inkluderte satellittdataen, viste resultatene at ytelsen ble forbedret, så lenge man kunne garantere en klassifiseringsnøyaktighet av isfjell på over 75%.

Resultatene fra denne oppgaven, støtter bruken av en kombinert søk- og sporingsalgoritme for operasjoner hvor ubemannede fly blir benyttet til å søke og spore isfjell. Likevel er det fortsatt noen problemer som må løses, for eksempel når det kommer til algoritmens ytelse i et område med raskt skiftende vær. Fremtidig arbeid bør derfor fokusere på å gjøre algoritmen mer robust, før man tester ut algoritmen i praksis på en UAV.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and motivation

The search for oil and gas in the Arctic region has seen an increased interest the recent years. The US Geological Survey estimates that 25 % of the remaining hydrocarbon resources in the world are located in the Arctic [1], so a major increase in oil drilling operations in Arctic areas must be expected. One critical aspect when it comes to Arctic offshore activities, is how to handle the ice, which in later years has been referred to as Ice Management. This includes [1]:

1. Detection, tracking and forecasting of sea ice, ice ridges and icebergs.
2. Threat evaluation.
3. Physical ice management such as ice breaking and iceberg towing.
4. Procedures for disconnection of offshore structures applied in search for or production of hydrocarbons.

How, and to what extent these tasks are performed, depends on the application. For oil drilling companies it is important to minimize downtime during operations, in order to maximize profit. Therefore, much research has been put on tasks 1-3 to reduce the disconnection time implied by task 4. By detecting and removing threatening ice, the drilling procedure can continue without interruptions.

Physical ice management is normally done by boats, which relies on information provided from other applications. These applications cover detection, tracking and forecasting of sea ice, in combination with threat evaluation when it comes to deciding which icebergs that should be towed away. How detection, tracking and forecasting are practised, is crucial to succeed with ice management. The main focus of this thesis will therefore be related to this. More information about the procedures and challenges regarding point 2-4 in the listing above, can be found in [2] and [1].

Due to the risks which are present when operating in Arctic environments such as wind, waves, darkness and low visibility, the use of Unmanned Aerial Vehicles (UAVs) for information gathering will be an important tool when investigating these areas. The main task for the UAV in ice management, is to fly a predetermined path and, while flying, collect data that can be used for tracking and classification of unknown objects. The detection of objects are based on on-board sensors such as camera, LIDAR (Laser Illuminated Detection and Ranging) and RADAR. As an example [3], uses an infra-red camera, together with a real-time camera-vision module installed in a two-axis gimbal system, as data collector and processor with the goal to compute optimal paths for the UAV using Model Predictive Control (MPC).

The topic of detection, tracking and forecasting is complicated, but can be split into two main tasks: Discovering unknown ice, and tracking of already discovered ice. For search after unknown ice, the main challenge is related to the fact that the surveillance area is too big to be covered by a single UAV in one efficient mission. To get a hot start and focus on the most important areas, one therefore has to take advantage of a priori data, such as satellite imagery and/or data about the current, wind and waves. Ice tracking applications employ previous observations to estimate ice dynamics. In order for this to be accurate, UAVs can be used to provide updated measurements about the current states.

[4] presents a target detection strategy for multiple agents. Here, the UAVs gather and share information about the likely locations of a target, with the objective of finding the target in minimum time. This is a promising approach, which can reduce the search time compared to single UAV missions. The downside is that it requires several UAVs in operation, which increases the risks and expenses. For this matter, only single UAV operations will be considered in this thesis.

When it comes to tracking, much work has been put into researching how to build up a framework for object tracking, based on a priori observations. [5] investigates the possibilities of estimating the dynamics of surface object by using a Kalman filter. The estimates are based on observations made by the UAV and the path is then computed using nonlinear programming (NLP), with the goal to minimize the uncertainty in the model. Similarly, [6] proposes a Mixed Integer Linear Programming (MILP) framework for monitoring of moving targets with a single UAV. The goal is to solve a Targets Visitation Problem (TVP), which is a special case of the Travelling Salesman Problem (TSP) where the possibility of prioritizing targets is included. The path is recomputed in an MPC manner, i.e., every time the first iceberg in the path is reached. A tuning mechanism is also introduced, where a parameter μ is used to weight if the optimization problem should be solved with respect to minimizing the estimates uncertainty or the to-

tal flying path distance.

In this thesis, detection and tracking of icebergs will be studied. Ideas from [5] and [6] will be employed, where the surface objects mentioned will be icebergs moving on open water. Common for [5] and [6] is that they both focus on tracking discovered objects, assuming they already have estimates about their location. In fact, most systems for ice management separate the two tasks of tracking already discovered ice, and discovering unknown ice, into two separate problems. However, combining these tasks into a single algorithm might only require one UAV, which could reduce the operating costs. In this thesis, we propose a combined search and track algorithm and investigate its performance and viability.

1.2 Outline

The thesis is divided into six chapters:

- Chapter 1 serves as an introduction to the problem, in addition to abbreviations and notation used.
- The data used throughout this thesis, how it has been collected, and how it will be used is outlined in Chapter 2.
- In Chapter 3, the mathematical models and methods used in this thesis, are presented.
- Chapter 4 covers the simulator development process, including necessary assumptions.
- In Chapter 5, simulation results are presented and discussed.
- Concluding remarks, in addition to further work are presented in Chapter 6.

1.3 Contributions

The main focus of this thesis will be on combining the tasks of object search and tracking into one algorithm. A similar framework as in [5] will be used, where the objects will be tracked using a Kalman filter. Instead of using NLP, the path will be computed using Dynamic Programming (DP). In addition, real weather data from wind and ocean current will be used to analyse the threat level around a vessel in operation. This will be used by the search model to determine critical areas that should be inspected by a UAV. The goal is to combine the Kalman filter and search model in such a way that both search and track can be performed in one operation, with the aid of a single UAV. As in [6], a tuning mechanism will be introduced, in order to be able to prioritize which of the two subtasks that should be weighted more when solving for the optimal path. To test the combined algorithm, a simulator will be developed. For the simulator environment to be as realistic as possible, a great deal of effort has been put into modelling of realistic iceberg drift.

1.4 Notation

The notation list is intended as a reference for the reader. The list includes symbols used throughout the thesis, and a corresponding explanation for what it represents. Bold symbols, e.g. $\mathbf{x}$ and $\mathbf{A}$, denotes a vector or a matrix.

Symbol	Explanation
k_l	Keel depth of an iceberg.
m	The physical mass of an iceberg.
s_l	Sail height of an iceberg.
$\mathbf{x}$	A vector containing the iceberg states.
$\mathbf{p}$	Iceberg position state. Part of $\mathbf{x}$.
$\mathbf{v}$	Iceberg velocity state. Part of $\mathbf{x}$.
$\mathbf{u}$	A vector containing the environmental driving forces for an iceberg.
m_{add}	Added mass of an iceberg.
m_t	Total mass of an iceberg.
$\mathbf{v}_w$	Wind velocity.
$\mathbf{v}_c$	Current velocity.
$\mathbf{F}_c$	Current force acting on the iceberg.
$\mathbf{F}_w$	Wind force acting on the iceberg.
$\mathbf{F}_{cor}$	Coriolis force acting on the iceberg.
C_w	Water drag coefficient of an iceberg.
C_a	Air drag coefficient of an iceberg.
ω	Angular velocity of the earth.
ϕ	Latitude position of an iceberg.

Symbol	Explanation
k_{tr}	Known threat probability.
u_{tr}	Unknown threat probability.
$\mathbf{w}_{xy}$	Each grid cells weather velocity component.
$\mathbf{v}_{ice}$	Approximated iceberg speed by the 2%-rule.
P_{xy}	Grid cell $(x, y)s$ position.
$d_{xy,vessel}$	Distance from grid cell (x, y) to the vessel.
$\bar{r}_{xy}$	Direction threat probability.
$\bar{d}_{xy}$	Distance threat probability.
$\hat{\mathbf{x}}_k$	Estimated state by the Kalman filter.
$\mathbf{z}_k$	Measurement by the UAV to update the Kalman filter states.
$\mathbf{P}_k$	Error covariance matrix corresponding to $\hat{\mathbf{x}}_k$.
$\mathbf{K}_k$	Kalman filter gain.
$\sum_{\mathbf{pp}}$	Position covariance.
$\sum_{\mathbf{vv}}$	Velocity covariance.
$\hat{\mathbf{p}}$	Iceberg position estimate.
$\hat{\mathbf{v}}$	Iceberg velocity estimate.
$\mathbf{v}_z$	Measured iceberg velocity by the UAV.
$\mathbf{x}_{xy}$	Grid cell $(x, y)s$ state vector.
$\mathbf{X}$	Occupancy grid map matrix containing $\mathbf{x}_{xy}$.
y_{xy}	Grid cell $(x, y)s$ weighted threat probability.
$\mathbf{Y}$	Threat probability matrix.
a	Tuning variable to prioritize iceberg tracking.
b	Tuning variable to prioritize iceberg search.
$\mathbf{J}$	Satellite image matrix.
$\mathbf{S}_{ice}$	Image of the true states of the icebergs.
$\mathbf{W}$	Satellite data uncertainty matrix.
V	The set of grid cells included in the UAV path.
d_{ij}	Cost of flying from grid cell $(x, y)_i$ to $(x, y)_j$.
δ	Upper limit for the number of grid cells that can be included in V.
$\mathbf{D}$	Cost matrix.
n_i	The number of icebergs generated in a simulation.
$\mathbf{Q}$	Process noise matrix in Kalman filter.
$\mathbf{R}$	Measurement noise matrix in Kalman filter.

1.5 Abbreviations

The list of abbreviations is intended as a reference to the reader.

Abbreviation	Explanation
UAV	Unmanned Aerial Vehicle.
TSP	Travelling Salesman Problem.
MPC	Model Predictive Control.
NLP	Nonlinear Programming.
MILP	Mixed Integer Linear Programming.
TVP	Travelling Visitations Problem.
NP	Nondeterministic Polynomial.
DP	Dynamic Programming.
NED	North-East-Down.

1.6 Note on previous work

This master thesis is a continuation of the author's project thesis. One may therefore see some of the parts being reused in this thesis. This mainly applies for the introduction and some of the mathematical models.

Chapter 2

Data Collection

An important foundation for the mathematical models developed in this thesis, is weather data from wind and ocean current. Since we are to search and track icebergs, real data from the Arctic has been used. The data is provided from Copernicus The European Earth Observation Programme [7], which is the world's largest observation programme, directed by the European Commission and the European Space Agency. By the use of satellites, maps, ground based weather stations, ocean buoys and air quality monitoring they provide data which can be used in services regarding:

- Land Monitoring
- Emergency Management
- Marine Monitoring
- Atmosphere Monitoring
- Security
- Climate Change

In this thesis, data from the Marine Environment Monitoring Service has been used. By using previous data about wind and ocean currents, one can model the iceberg dynamics as realistic as possible in an area of interest.

The area that will be studied lies within the coordinates $78° - 79°$ latitude, and $4° - 9°$ longitude. This far north, $1°$ latitude ≈ 100 km and $1°$ longitude ≈ 20 km. This corresponds to an area of approximately 100×100 km^2 west of Svalbard, as shown in Figure 2.1 and 2.2. The data downloaded consist of data about wind- and ocean current velocities. Both has one eastward and one northward velocity component. The wind data is updated every 6 hours, while the ocean current data is updated every 24 hours. The wind data has measured values for every

0.25° longitude and latitude, while the ocean current data has measured values for every 0.5° longitude and latitude. Current data is provided in 23 different depth layers, starting at the ocean surface, descending to 110 meters.

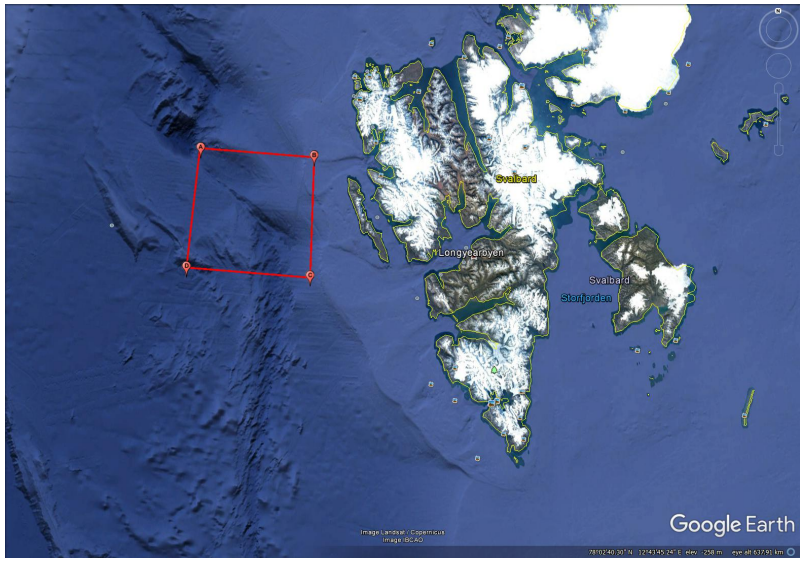

Figure 2.1: Operation area close-up [8].

Figure 2.2: Operation area zoomed out [9].

Interpolation has been used to increase the number of data points within the operation area. In this way it is possible to get approximate data values at every coordinate within the area, at every depth, at every instance of time. Through this thesis, data from October 1st 2016 to November 1st 2016 will be used. This means that simulations could potentially last for up to one month.

The data will be used through this thesis for three main tasks:

1. To model the iceberg dynamics. In order to test the search and track algorithm, icebergs and their dynamics will be modelled and used as the true states of the icebergs we are trying to detect and track. To get an iceberg drift that is as realistic as possible for the area of interest, wind and current data will be used.

2. To determine which areas that are of bigger threat than others. For the search part of the algorithm, we are interested in searching the most threatening areas. Wind and current directions can give us information about where potential threats are heading.

3. To track icebergs. This will be done by using a Kalman filter. For this filter to be able to estimate the velocities and positions of the tracked icebergs, wind and current data will be used.

A more detailed explanation of how the data will be used, is outlined in Chapter 3.

Chapter 3

Mathematical Modelling and Methods

The main goal in this thesis is to combine the usually decoupled tasks iceberg search and iceberg tracking in one algorithm. To do so, mathematical models are required. This chapter serves as an explanation to the theory behind these models, in addition to some modifications made to adapt the models to our system.

3.1 Iceberg drift modelling

In order to test the search and track algorithm, a realistic model of the reality has to be built. Much effort has therefore been put into modelling of both the iceberg's geometry and dynamics, which together determines its drift.

The geometry of an iceberg is decided based on five parameters: Length, width, sail height, sail shape, keel depth and keel shape. The sail height is defined as the vertical length of the iceberg above ocean surface. Similarly, the keel depth is defined as the vertical length of the iceberg below ocean surface. The icebergs come in many different shapes. To account for this, three different shapes will be considered in this thesis. Inspired by [10], a rectangular, a semi-elliptic, and a triangular keel shape will be used. Since only approximately 10% of an iceberg is above ocean surface, the sail shape will not influence the dynamics much. Therefore, each iceberg will be modelled with the same sail shape, formed as an elliptic cylinder. Figure 3.1 illustrates a cross section of the three iceberg shapes, in addition to their relation to sail height and keel depth. Figure 3.2 shows the projection of the elliptic cylinder onto the surface plane. The iceberg width is defined as the semi-minor axis of this ellipse, while the length is defined as the semi-major axis.

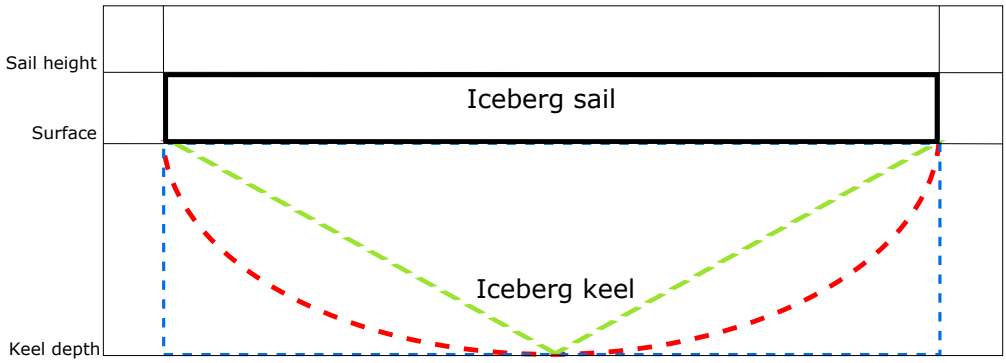

Figure 3.1: Iceberg seen from the side. Illustrates the three shapes. The rectangular sail shape is meant to illustrate a cylinder seen from the side. Figure inspired by [10].

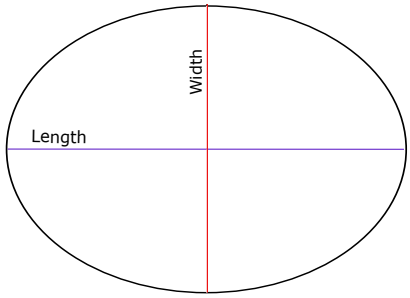

Figure 3.2: Iceberg seen from above. Shows length and width parameters.

Each iceberg is generated at random. This means that the iceberg is placed on a random location in the operational area, while its length and width are drawn from a uniform distribution in the range 10-200 meters. The length is then sat to be the greatest of these two values. Various studies have been carried out to study how the iceberg's length is related to its keel depth, sail height and mass. One of these studies, performed by A. Barker et al. [11] derives the following empirical relationship between length (l) and keel depth (k_l), using curve fitting and data from 14 observations:

$$k_l = 2.91 \times l^{0.71}$$

By using regression analysis, a linear approximation is derived:

$$k_l = 0.7 \times l \tag{3.1}$$

In the same study, similar analyses were carried out to find the length-mass relationship, resulting in the following equation:

$$m = 0.5 \times \rho_{ice} \times l^3 \tag{3.2}$$

Here, m is the physical mass of the iceberg and $\rho_{ice} = 900$ is the density of ice.

Equation (3.1) and (3.2) have been used to compute the iceberg's draft and mass based on its length. To compute the sail height (s_l), the results presented in [12] has been used. Without going into detail, the following relationship was derived by studying 230 pairs of observations:

$$s_l = 0.402 \times l^{0.89}$$

Every iceberg is assigned either a semi-elliptic, rectangular or triangular keel shape, as mentioned above. These are assigned at random. The keel is then split into 23 different parts, one for each current layer in the data set. In this way every current component act on its own part of the keel, for which the geometry is decided by the shape assigned at initialization.

Each iceberg has a unique water and air drag coefficient, respectively C_w and C_a, which accounts for the uncertainties in iceberg surface roughness, keel depth and mass. In [14], these coefficients were through experiments and data analyses determined to lie in the range $0.1 - 2.4$. Each iceberg has therefore been assigned these coefficients as random values in this range.

The iceberg geometry and weight will determine how the iceberg will be affected by external forces, and hence how the iceberg will drift. Based on this, the iceberg dynamics can be modelled as a set of ordinary differential equations:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}_0 = \mathbf{x}(t_0) \tag{3.3}$$

Here, $\mathbf{x}$ is a vector containing the iceberg states (position and velocity), with

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} x_{pos} \\ y_{pos} \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{3.4}$$

and $\boldsymbol{u}$ are the environmental driving forces acting on the iceberg. Further, $\mathbf{x}_0$ is the initial state vector.

The equation of motion for each iceberg can be derived by using Newtons second law of motion

$$\sum \mathbf{F} = m_t \mathbf{a} \tag{3.5}$$

where $\mathbf{a}$ is the iceberg acceleration and $\sum \mathbf{F}$ is the sum of the forces acting on the iceberg. m_t is the total iceberg mass, i.e $m_t = m + m_{add}$. As recommended in [17], m_{add} has been sat to $\frac{1}{2}m$, which gives $m_t = \frac{3}{2}m$.

Using that $\mathbf{a}$ is the time derivative of $\mathbf{v}$ we get

$$m_t \frac{d\mathbf{v}}{dt} = \sum \mathbf{F} \tag{3.6}$$

In this thesis, it is assumed that the following forces act on the icebergs:

$$\mathbf{u} = \begin{bmatrix} \mathbf{F}_{cor} \\ \mathbf{F}_c \\ \mathbf{F}_w \end{bmatrix}$$

which gives

$$m_t \frac{d\mathbf{v}}{dt} = \mathbf{F}_{cor} + \mathbf{F}_c + \mathbf{F}_w \tag{3.7}$$

Here, $\mathbf{F}_w$ and $\mathbf{F}_c$ are wind and current forces, respectively, and $\mathbf{F}_{cor}$ is the Coriolis force. These forces, and how they act on an iceberg are illustrated in Figure 3.3. For simplicity the iceberg has been illustrated as a cylinder.

.

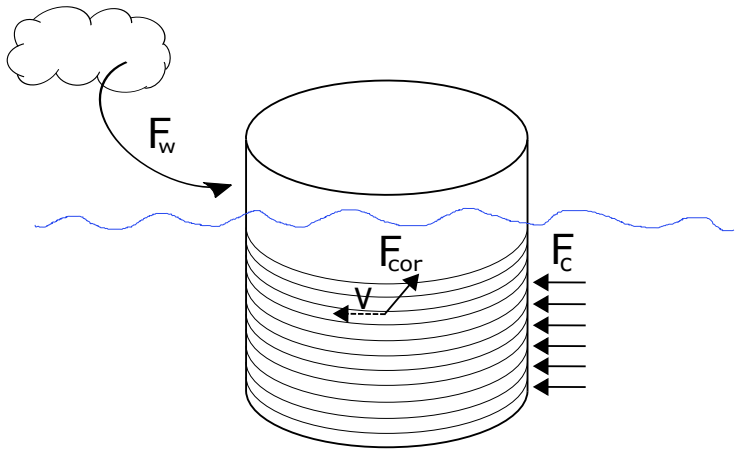

Figure 3.3: Forces acting on an iceberg. Figure inspired by [10].

The current force is caused by the current acting on the k layers of the iceberg keel and is calculated by

$$\mathbf{F}_c = \frac{1}{2}\rho_w C_w \sum_{k=1}^{n} A_c(k)|\mathbf{v}_c(k) - \mathbf{v}|(\mathbf{v}_c(k) - \mathbf{v}) \tag{3.8}$$

where $\rho_w = 1027\ \mathrm{kg/m^3}$ is the water density, and C_w is the water drag coefficient. Since each of the current forces, acts on the keel in different directions, one force component is computed for all n current layers. The current velocity component corresponding to this is $\mathbf{v}_c(k)$, which acts on the kth keel cross section, $A_c(k)$.

These components are then summed up to give the total current force component, $\mathbf{F}_c$.

Similarly, the wind force is caused by the wind acting on the iceberg sail and is calculated by

$$\mathbf{F}_w = \frac{1}{2}\rho_a C_a A_w |\mathbf{v}_w - \mathbf{v}|(\mathbf{v}_w - \mathbf{v}) \tag{3.9}$$

Here, $\rho_a = 1.225$ kg/m^3 is the air density, and C_a is the air drag coefficient. A_w is the sail cross section and is the part of the iceberg above water, that is vertical to the incoming wind. An approximation of the cross section of the elliptical cylinder can be computed as:

$$A_w = 2rs_l \tag{3.10}$$

where r is the average radius of the major and minor axis of the ellipse, and s_l is the sail height. This is a simplification, but should be sufficient since divergence from the actual model is accounted for through the drag coefficient, C_w. The incoming wind acting on this cross section is $\mathbf{v}_w$. As explained in Chapter 2, the wind and current velocity components used to compute $\mathbf{F}_w$ and $\mathbf{F}_c$ are real data, gathered at the exact position the iceberg is located.

The third force acting on the iceberg, is the Coriolis force, $\mathbf{F}_{cor}$. The Coriolis force is caused by the rotation of the earth and can be computed as:

$$\mathbf{F}_{cor} = -m2\omega \sin(\phi)\mathbf{k} \times \mathbf{v} \tag{3.11}$$

where m is the physical mass of the iceberg, $\omega = 7 \times 10^{-5}$ is the angular velocity of the earth, ϕ is the latitude of the iceberg's position, $\mathbf{k}$ is the unit vector directed upwards parallel to the z-axis and $\mathbf{v}$ is the iceberg velocity. The Coriolis force will deflect the iceberg in either a clockwise or counter clockwise direction, depending on its location. Since the operation takes place on the northern hemisphere, this effect will deflect the iceberg clockwise [15].

With the mathematics behind the environmental forces established, we return to the equations of motion. Since the iceberg dynamics are slow, and position and velocity can be considered constant for small periods of time, a discrete model will be used. We rewrite (3.6) to:

$$\frac{m_t(\mathbf{v}_{k+1} - \mathbf{v}_k)}{\Delta t} = \mathbf{F}_{cor} + \mathbf{F}_c + \mathbf{F}_w$$

$$\Rightarrow \mathbf{v}_{k+1} = \mathbf{v}_k + (\frac{\mathbf{F}_{cor} + \mathbf{F}_c + \mathbf{F}_w}{m_t})\Delta t \tag{3.12}$$

We can now write (3.3) as

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{x}_0 = \mathbf{x}_{k=0} \tag{3.13}$$

where the states can be updated according to

$$\begin{aligned} \mathbf{p}_{k+1} &= \mathbf{p}_k + \mathbf{v}_k \Delta t \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + (\frac{\mathbf{F}_{cor} + \mathbf{F}_c + \mathbf{F}_w}{m_t})\Delta t \end{aligned} \tag{3.14}$$

The discrete time ODE can now be written as

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k \tag{3.15}$$

with

$$\begin{gathered} \mathbf{x} = \begin{bmatrix} x_{pos} \\ y_{pos} \\ u \\ v \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{F}_{corr} \\ \mathbf{F}_c \\ \mathbf{F}_w \end{bmatrix}, \\ \mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \frac{1}{m_t} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Delta t & \Delta t & \Delta t \\ \Delta t & \Delta t & \Delta t \end{bmatrix} \end{gathered} \tag{3.16}$$

where Δt is the time step between the states $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$.

This completes the iceberg drift model. Equation (3.15) can now be used, together with the weather data, to generate iceberg trajectories. These will be used as the true state of the icebergs, during the simulations, and will be unknown to the UAV and the search and track algorithm.

3.2 Threat modelling, prediction and forecasting

We can now start building a model for the search and track algorithm. This involves modelling of the search area, a model for tracking, a model for search, and most importantly, a way to combine these models. We start by defining a framework we can build this in.

3.2.1 Occupancy grid mapping

Autonomous vehicles must be able to operate in unknown environments without prior knowledge about the map. For this to be possible, one must heavily rely on non-deterministic sensor data, which may be affected by noise and uncertainty in measurements. Occupancy grid mapping addresses these requirements and allows several different robot applications to be performed directly on the grid environment. The basic idea of occupancy grid mapping is to represent a map as a discrete grid, where each grid cell holds a probabilistic estimate of its state.

The state variable is defined as a discrete random variable with two states, empty or occupied.

In this thesis, an occupancy grid mapping inspired framework has been used. The search and track algorithm should make the UAV both track and search icebergs. To do so, the notion of empty or occupied is modified to a more general term: Threat or no threat, which will be quantified by the two variables:

$$k_{tr} = \text{"kown threat"} \quad \text{and} \quad u_{tr} = \text{"unknown threat"} \tag{3.17}$$

Here, k_{tr} quantifies the probability for a grid cell to contain an iceberg, i.e. the probability that a known iceberg that we are tracking is occupying this grid cell. Similarly, u_{tr} tells us something about the current threat a specific grid cell poses to the vessel in operation, when none of the icebergs' positions are known. These variables can now be used to give priority to both search and detection of unknown threats, and tracking of known threats, during a UAV mission.

The search area, which is modelled as a square with the vessel in its centre, consists of $n \times n$ grid cells which all have corresponding values for u_{tr} and k_{tr}. These values will determine if the particular cell should be considered a threat to the vessel in operation or not. u_{tr} and k_{tr} will be computed based on two different models:

1. An unknown threat model that uses weather data to predict in which direction, potential objects in each grid cell are moving.
2. A known threat model based on a Kalman filter, keeping track of each detected iceberg.

The models are outlined below.

3.2.2 Unknown threat model

The unknown threat model quantifies how threatening potential objects in a given grid cell are to the vessel in operation. This is stored in the variable u_{tr}. Since we do not know whether a grid cell holds an iceberg or not, we always assume that this is the case, and analyse how threatening this iceberg is to the vessel. This will be determined by two physical factors: Weather data, and distance from the vessel.

Weather data
Prior to every mission, a weather map will be constructed. This weather map will be the same size as the occupancy grid map, where each grid cell (x, y) holds a weather velocity component, $\mathbf{w}_{xy}$. To compute these velocity components, data from ocean currents and wind will be used. This is real data, collected from the

area the search and track operation will be located. By setting the simulation start time to one of the days we have data from, this weather data can be viewed as real-time data which will be updated according to what was outlined in Chapter 2.

When computing these velocity components, we do not know anything about size, shape and mass of the potential icebergs we are calculating the velocity of. Therefore, we cannot use the kinematic model derived in section 3.1. A different approach, based on the so-called 2%-rule, has therefore been taken. The 2%-rule states that the icebergs move with approximately 2 % of the wind velocity plus the mean of the ocean current [16]. The velocity of potential icebergs in a grid cell can therefore be approximated to a simple kinematic model:

$$\mathbf{v}_{ice} \approx \bar{\mathbf{v}}_{current} + 0.02\mathbf{v}_{wind} \tag{3.18}$$

This model is based on observations and has some limitations, but as long as the icebergs does not get too large, this is a good rule of thumb when the iceberg dimensions are unknown.

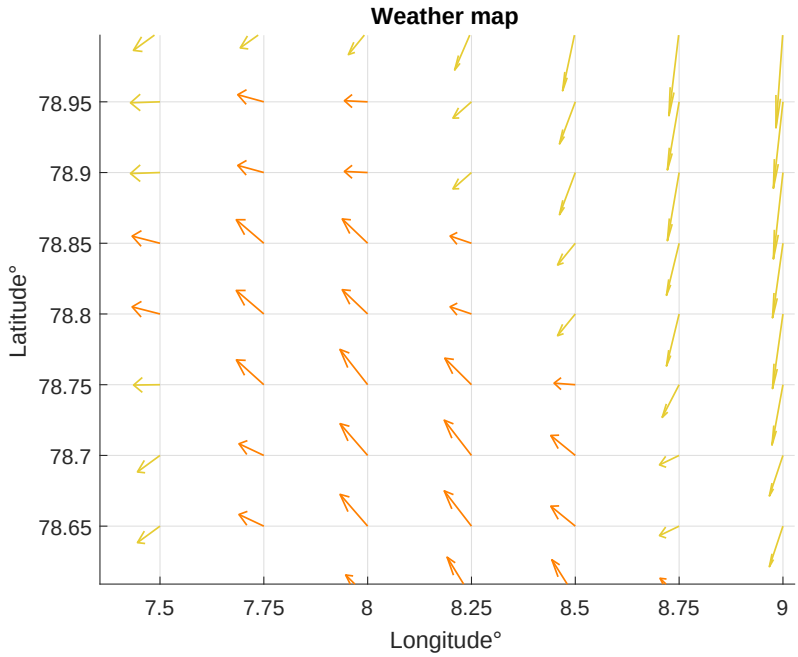

Figure 3.4: An example of how the computed 2%-rule weather map could look. The arrows show the computed velocity for each grid cell in a small area.

By computing this for every grid cell in the occupancy grid map, we get a weather map saying something about the velocity an iceberg would have, within a given grid cell. An example of how such a weather map could look during a simulation, is illustrated in Figure 3.4. Here, the arrows point in the direction of movement, while the arrow length illustrates the speed.

Direction

We define the direction weight for each grid cell, r_{xy}, to be the shortest distance from the vessel in operation to the line between P_{xy} and the point

$$Q_{xy} = P_{xy} + \frac{\vec{w}_{xy}}{\|\vec{w}_{xy}\|} \, d_{xy,vessel} \tag{3.19}$$

P_{xy} is the grid cells position and $d_{xy,vessel}$ is the euclidean distance from P_{xy} to the vessels position, defined in (3.22). This means that if $\mathbf{w}_{xy}$ is pointing directly towards the vessel, r_{xy} is zero. If $\mathbf{w}_{xy}$ is pointing away from the vessel, the shortest distance is

$$r_{xy} = d_{xy,vessel} \tag{3.20}$$

The graphical interpretation of r_{xy} is illustrated in Figure 3.5.

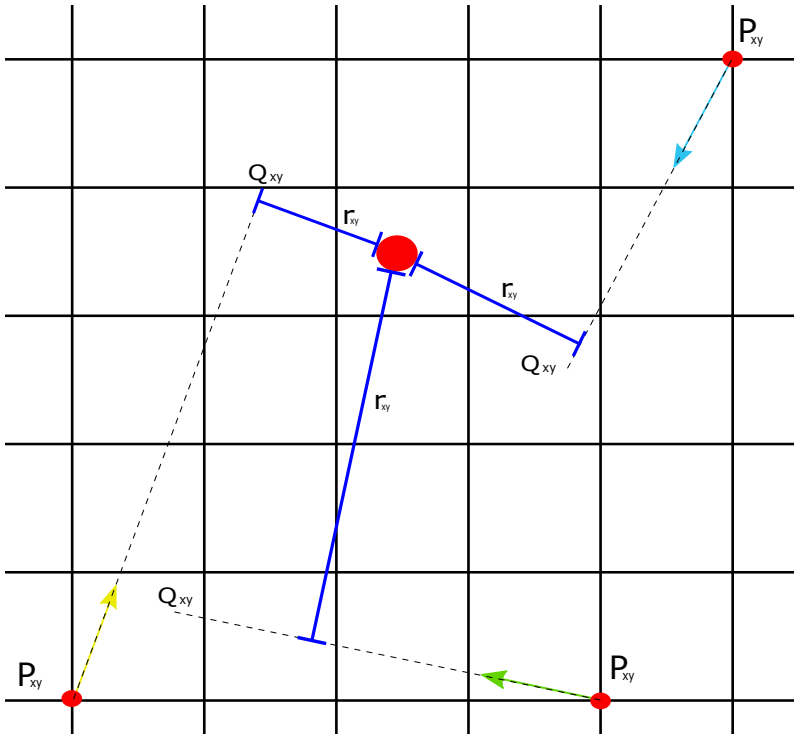

Figure 3.5: The direction weight r_{xy} illustrated for three different $\mathbf{w}_{xy}$.

Here, r_{xy} is computed for three different grid cells. The arrows illustrate $\mathbf{w}_{xy}$, while the dotted line shows the line from P_{xy} to Q_{xy}. We see that the blue lines, representing the distance r_{xy}, have different lengths: The shorter the length of the lines, the bigger threat objects inside the grid cell poses to the vessel.

We want this direction weight to be expressed as a probability for being a threat. This can be done by normalizing and subtracting this from 1. This gives

$$\bar{r}_{xy} = 1 - \frac{r_{xy}}{d_{xy,vessel}} \tag{3.21}$$

From this we see that potential objects that will pass by close to the vessel is assigned a high threat probability, because $d_{xy,vessel} >> r_{xy}$, and objects that pass by far away is assigned a low value, since $d_{xy,vessel} \approx r_{xy}$.

We now have a way to compute grid cell (x, y)'s probability for being a threat to the vessel, when only direction of movement is considered. We now add distance as a threat factor.

Distance
We have accounted for the fact that the grid cell could be moving towards the vessel in operation. What also has to be considered when computing u_{tr}, is the euclidean distance each grid cell has from the vessel. Icebergs that are close is obviously a bigger threat than icebergs that are far away, and this has to be accounted for in the model. The euclidean distance, $d_{xy,vessel}$, is calculated as

$$d_{xy,vessel} = \sqrt{(P^x_{xy} - P^x_{vessel})^2 + (P^y_{xy} - P^y_{vessel})^2} \tag{3.22}$$

We want this number to express grid cell (x,y)'s probability of being a threat to the vessel, when only distance is considered. We obtain this by normalizing and subtracting from 1 (as above):

$$\bar{d}_{xy} = 1 - \frac{d_{xy,vessel}}{d_{max}} \tag{3.23}$$

where d_{max} is the maximum distance from the vessel position to a point in the grid space. Since we assume that the vessel is in the centre of the map, d_{max} will be the distance to one of the corner points.

Model
With the models behind $\bar{r}_{xy}$ and $\bar{d}_{xy,vessel}$ established, the total "unknown threat" each grid cell poses to the vessel in operation can be computed by multiplying the two probabilities. This gives

$$u_{tr} = \bar{r}_{xy} \times \bar{d}_{xy,vessel} \tag{3.24}$$

where

$$u_{tr} \in [0, 1]$$

3.2.3 Known threat model

We now have a model for u_{tr}, saying something about the probability for a grid cell being a threat to the vessel in operation, when only direction of movement and distance are considered. In this section, a model for the known threats, k_{tr}, will be derived.

To model and keep track of the known threats (icebergs), a Kalman filter will be used. The Kalman filter is a recursive filter that can be used to estimate states from measurements that can be affected by noise. It can be used on both continuous and discrete systems and has the nice property that it can be used to make an educated "guess" about the future states.

Discrete Kalman filter
Since we are dealing with a discrete map, the discrete Kalman filter will be used in our model. The process can be modelled as

$$\mathbf{x_{k+1}} = \mathbf{\Phi_k x_k} + \mathbf{\Delta u_k} + \mathbf{w_k} \tag{3.25}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3.26}$$

where $\mathbf{x}_k$ is the process state vector at time t_k, $\mathbf{\Phi}_k$ is a known matrix representing the relationship between $\mathbf{x}_{k+1}$ and $\mathbf{x}_k$, $\mathbf{\Delta}$ is the control matrix, $\mathbf{w}_k$ is the process noise vector, and $\mathbf{u}_k$ is a vector containing external influences which are not related to the state itself. Similarly $\mathbf{z}_k$ is the measurement vector at time t_k, $\mathbf{H}_k$ is a known matrix representing the relationship between $\mathbf{z}_k$ and $\mathbf{x}_k$, and $\mathbf{v}_k$ is the measurement noise vector. Both $\mathbf{w}_k$ and $\mathbf{v}_k$ are modelled as a white sequence with zero mean and covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively. Further, we assume no cross-correlation between $\mathbf{w}_k$ and $\mathbf{v}_k$, i.e.

$$E[\mathbf{w}_k \mathbf{v}_i^T] = \mathbf{0}, \quad \forall\ i, j \tag{3.27}$$

Essential to the Kalman filter algorithm is the *a priori* state estimate $\hat{\mathbf{x}}_k^-$. We define the estimation error and the corresponding error covariance matrix to be

$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^- \tag{3.28}$$

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T] \tag{3.29}$$

This *a priory* estimate is based on knowledge about the process prior to t_k and can be computed from the equation

$$\hat{\mathbf{x}}_{k+1}^- = \mathbf{\Phi}_k \hat{\mathbf{x}}_k + \mathbf{\Delta}_k \mathbf{u}_k \tag{3.30}$$

Here, $\hat{\mathbf{x}}_k$ is the updated estimate which is a linear blending of the prior estimation and the measurement, $\mathbf{z}_k$ provided from sensors. For every iteration, the updated estimate is computed from the equation:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^-) \tag{3.31}$$

The matrix $\mathbf{K}_k$ is called the Kalman gain and is one of the most important parts of the Kalman filter. From (3.31) we observe that $\mathbf{K}_k$ determines how much the error between previous estimates and new measurement should be weighted. A small $\mathbf{K}_k$ will result in small changes in the estimated state, even if the measured states are way off. How we pick $\mathbf{K}_k$ is therefore crucial to the algorithm.

The error covariance matrix corresponding to the updated estimate, $\hat{\mathbf{x}}_k$ can be computes as in (3.29):

$$\mathbf{P}_k = E[\mathbf{e}_k\mathbf{e}_k^T] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T] \tag{3.32}$$

Using (3.31) and (3.27) gives:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^{-1} + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{3.33}$$

The Kalman gain $\mathbf{K}_k$ can now be found by solving (3.33) for the $\mathbf{K}_k$ that minimizes the estimation error variance. Since the estimation error variance appears on the diagonal of $\mathbf{P}_k$, we use the trace derivative and solve for the $\mathbf{K}_k$ that minimizes the expression. This gives

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{3.34}$$

We now have an equation for how to update the state estimate for each iteration of the algorithm. The final part of each iteration is to compute $\mathbf{P}_{k+1}^-$ which will be used as $\mathbf{P}_k^-$ in the next iteration. This can be computed from the equation

$$\mathbf{P}_{k+1}^- = \mathbf{\Phi}_k\mathbf{P}_k\mathbf{\Phi}_k^T + \mathbf{Q}_k \tag{3.35}$$

where $\mathbf{P}_k$ is given by (3.33). This leaves us with a recursive algorithm, in the sense that only present input measurements and the last calculated state and uncertainty matrix is required, to compute the next step. The algorithm can be summarized with the illustration shown in Figure 3.6. Further information about the Kalman Filter and its applications can be found in [18].

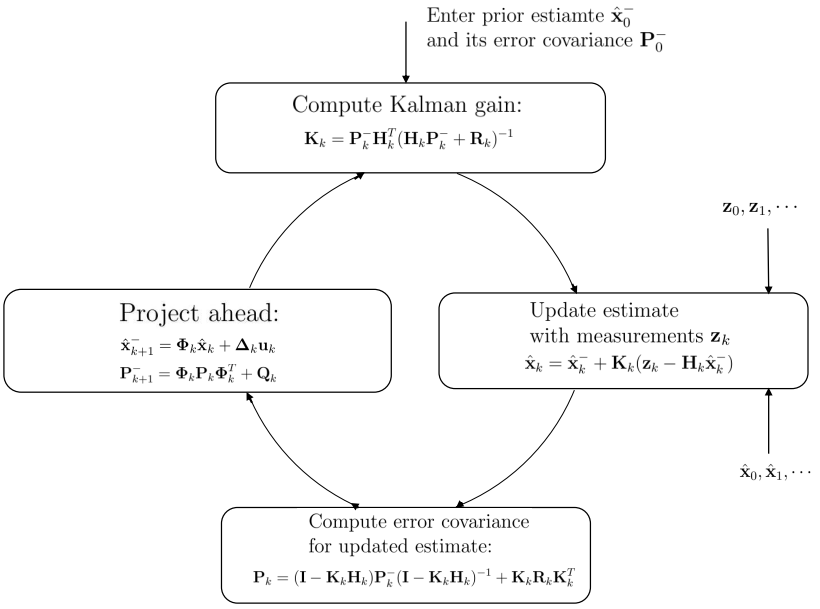

Figure 3.6: Kalman filter loop. Inspired by Figure 4.1, p. 147 in [18].

Modifications
We aim to use the Kalman filter to estimate the icebergs' position and velocity. This gives the state estimate vector

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{p}} \\ \hat{\mathbf{v}} \end{bmatrix} \tag{3.36}$$

and the corresponding covariance matrix:

$$\mathbf{P} = \begin{bmatrix} \Sigma_{\mathbf{pp}} & 0 \\ 0 & \Sigma_{\mathbf{vv}} \end{bmatrix} \tag{3.37}$$

where $\Sigma_{\mathbf{xy}}$ denotes the covariance matrix between vectors $\mathbf{x}$ and $\mathbf{y}$, and zero correlation between position and velocity is assumed. The *a priory* position estimate can be calculated similar to the model used in section 3.1, assuming a linear relationship between position and velocity:

$$\hat{\mathbf{p}}_{k+1}^- = \hat{\mathbf{p}}_k + \hat{\mathbf{v}}_k \Delta t \tag{3.38}$$

where $\Delta t = t_k - t_{k-1}$ is the time since the last update (not to be confused with $\mathbf{\Delta}$).

The velocity will be estimated in two different ways:

1. When an iceberg is discovered (or rediscovered), the velocity estimate will be set equal to the velocity measured by the UAV. Since the iceberg dynamics are slow we assume that the iceberg will keep this velocity for the first two hours after the UAV made the measurement.

2. If no new measurement has been made of that same iceberg after two hours, the velocity will be updated according to the kinematic model (3.18), presented in section 3.2.2.

Thogether, this gives the *a priory* velocity estimate:

$$\hat{\mathbf{v}}_{k+1}^{-} = \begin{cases} \mathbf{v}_z & \forall t_z \leq 2 \text{ hours} \\ \bar{\mathbf{v}}_c + 0.02\mathbf{v}_w & \forall t_z > 2 \text{ hours} \end{cases} \tag{3.39}$$

where $\mathbf{v}_z$ is the velocity measurement made by the UAV, $\bar{\mathbf{v}}_c$ is the mean of the current velocity component at the estimated position, $\mathbf{v}_w$ is the wind velocity component at the estimated position and t_z is the time since the last measurement was made. If the iceberg is rediscovered on a later occasion, t_z is set equal to zero.

We now have a model on the same form as (3.30) with:

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\mathbf{p}}_k \\ \hat{\mathbf{v}}_k \end{bmatrix}, \quad \mathbf{u}_k = \begin{bmatrix} \mathbf{v}_z \\ \bar{\mathbf{v}}_c \\ \mathbf{v}_w \end{bmatrix}, \quad \mathbf{\Phi} = \begin{bmatrix} 1 & \Delta t \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{\Delta} = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} & \forall t_z \leq 2 \text{ hours} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0.02 \end{bmatrix} & \forall t_z > 2 \text{ hours} \end{cases} \tag{3.40}$$

From the literature we know that iceberg drift is hard to forecast and that factors like shape, size, mass wind and ocean currents, affect its dynamics [10, 16, 17]. Because of this, and the fact that most of these factors are unknown, one should therefore not put too much trust in the estimate, compared to what is measured by the UAV. To account for this in the model, $\mathbf{Q}$ has been chosen such that $\mathbf{Q} >> \mathbf{R}$. Doing so will make $\mathbf{K} \to \mathbf{I}$, which in

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^{-}) \tag{3.41}$$

will push $\hat{\mathbf{x}}_k$ towards $\mathbf{z}_k$.

With a big $\mathbf{Q}$, we see from (3.35) that $\mathbf{P}_{k+1}^{-}$, and hence $\mathbf{P}_k$, will increase dramatically. In our model, an estimate's covariance matrix, $\mathbf{P}_k$, is reduced when the estimate is updated with a measurement. In this way, the lack of measurements will increase the estimate uncertainty fast, which is good, considering the low trustworthiness mentioned above.

Mapping from dynamic to static

By using this Kalman filter model, we can assign a probability, k_{tr}, to each grid cell. Since k_{tr} is static and fixed to the grid cell, while the tracked icebergs in the Kalman filter are dynamic, we need a way to map the dynamic real-world environment onto the static grid map.

Each iceberg that is being tracked has an estimate of its current position, $\hat{\mathbf{p}}$, and a corresponding covariance matrix $\Sigma_{\mathbf{pp}}$. Since the Kalman filter assumes that the variables are random and Gaussian distributed, each tracked iceberg form a 2-dimensional Gaussian, as illustrated in Figure 3.7. Figure 3.8, shows the contour plot of the same Gaussian.

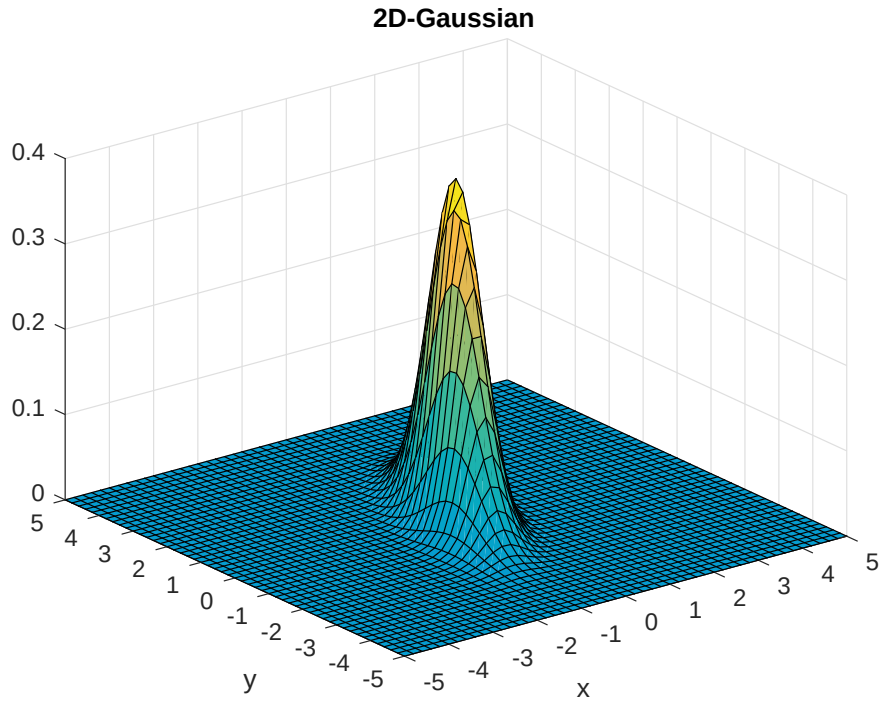

Figure 3.7: Illustration of a 2D-Gaussian.

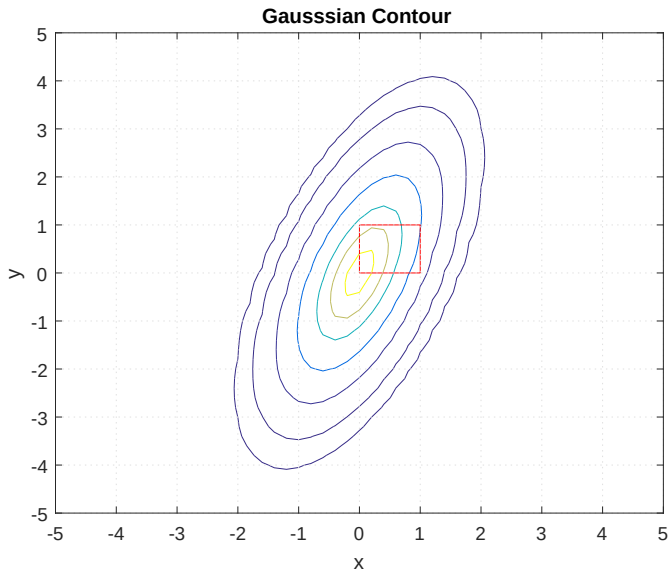

Figure 3.8: Contour plot of the Gaussian from Figure 3.7.

Let us assume that this Gaussian belongs to tracked iceberg i, and that the red rectangle in Figure 3.8 covers grid cell (x, y). By calculating the cumulative probability of the Gaussian pdf over this rectangle, we obtain grid cell $(x, y)'s$ probability for being occupied by iceberg i. If we repeat this for each of the tracked icebergs and sum the probabilities, we get this grid cell's probability for being occupied by a known threat. This is what we call k_{tr}. Considering grid cell (x, y), $k_{tr}(x, y)$ can be expressed mathematically as:

$$k_{tr}(x, y) = \sum_{i}^{N_i} F(i, x, y) \tag{3.42}$$

where $F(i, x, y)$ is a cumulative density function that computes the probability that iceberg i is located in grid cell (x, y), and N_i is the number of tracked icebergs.

Since we are summing up the probabilities, $k_{tr}(x, y)$ is not a probability in the true sense. However, since the chances are small that more than one iceberg will be occupying the same grid cell, we will treat this sum as a threat probability. To avoid that $k_{tr}(x, y)$ gets higher than one, we set:

$$k_{tr}(x, y) = \min(k_{tr}(x, y), 1) \tag{3.43}$$

Since grid cells with a threat probability of one will be prioritized anyway, we do not lose much generality by rounding k_{tr} down.

3.3 Combining the models

We can now move on and combine k_{tr} and u_{tr} to get a complete mathematical model for the system. To account for the fact that u_{tr} depends directly on the distance $\bar{d}_{xy,vessel}$, k_{tr} is adjusted slightly to:

$$k_{tr} = k_{tr} \times \bar{d}_{xy,vessel} \tag{3.44}$$

In this way objects that are being tracked that are close to the vessel in operation will be assigned higher threat probability, compared to the ones that are far away.

For every grid cell in the grid map there is a corresponding state vector,

$$\mathbf{x}_{xy}(k) = \begin{bmatrix} k_{tr} \\ u_{tr} \end{bmatrix} \tag{3.45}$$

The occupancy grid map can now be mathematically defined as a matrix containing these state vectors, i.e.:

$$\mathbf{X}(k) = \begin{bmatrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \cdots & \mathbf{x}_{1,n} \\ \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{n,1} & \dots & \dots & \mathbf{x}_{n,n} \end{bmatrix} \tag{3.46}$$

The goal is to determine, based on the state $\mathbf{X}(k)$, which grid cells that should be included in the UAV's flight path. In [5] and [6], which were mentioned in Chapter 1, this was done with the goal to track already discovered icebergs. In this project, u_{tr} will be included to also give weight to areas that are of a specific threat due to weather data. In this way, both iceberg search and tracking are encapsulated.

We want to combine k_{tr} and u_{tr} such that we end up with a weighted probability, quantifying the need to include this grid cell in our path. We define the output matrix $\mathbf{Y}$ as

$$\mathbf{Y}(k) = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ y_{n,1} & \dots & \dots & y_{n,n} \end{bmatrix} \tag{3.47}$$

where

$$y_{xy}(k) = \mathbf{c}^T \mathbf{x}_{xy}(k), \quad \mathbf{c} = \begin{bmatrix} a \\ b \end{bmatrix} \tag{3.48}$$

$$\Rightarrow y_{xy}(k) = a k_{tr} + b u_{tr}, \quad a + b = 1 \tag{3.49}$$

where a and b are tunable prioritizing weights. Choosing $a > b$ will prioritize tracking of icebergs with high k_{tr}. On the other hand, choosing $a < b$ will give more weight to iceberg search in grid cells with high u_{tr}. Note that a and b always has to sum up to 1, which gives

$$y_{xy} \in [0, 1]$$

The special case $a = b = 0.5$ represents the case when both tasks are prioritized just as much.

We now have our mathematical models and tuning mechanism, all encapsulated in $\mathbf{Y}$. The entries of $\mathbf{Y}$ can now be used to compute the optimal path.

3.4 Satellite data

As an additional part of this thesis, we want to investigate the possibilities of integrating satellite data into the system. Different companies and agencies, like Copernicus and Polar View, provide free satellite imagery for the use in the Arctic region. By the use of different types of satellite imagery, they deliver detailed and up to date pictures of sea ice distribution which can be used by ships and other maritime applications. A satellite image of the operational area can be seen in Figure 3.9, where the operational area lies within the blue square. We see that there are no icebergs here at the moment, which could be because most of the ocean to the north and west is covered by sea ice. The light blue part in the upper part of the image is mainland Svalbard.

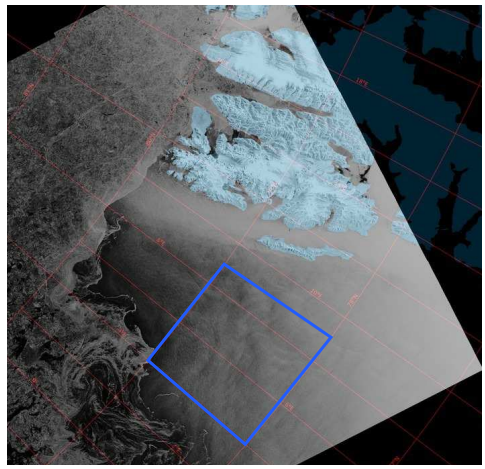

Figure 3.9: Satellite image of the operational area (blue square) [19].

We want to test if the use of satellite imagery can improve the combined algorithm's performance. Since the algorithm is to be tested through simulations, we need a way to model an image of the true states of the icebergs in the simulation.

The true states of the icebergs are stored in the matrix $\mathbf{S}_{ice}$. When a real satellite image is used, classification tools have to be used to classify objects on the image as icebergs. To account for the uncertainty in these tools, we will use the uncertainty matrix $\mathbf{W}$. The sum of $\mathbf{S}_{ice}$ and $\mathbf{W}$ gives a matrix containing the result of the classification. This gives the following model:

$$\mathbf{J} = \mathbf{S}_{ice} + \mathbf{W} \tag{3.50}$$

where $\mathbf{J}$ is called the image matrix. This image will be fed to the Kalman filter, where each of the classified icebergs in $\mathbf{J}$ will be added as a detection of a new iceberg. In this way, we do not need to make any changes to the other models derived above.

3.5 Path planning

The information provided from the models derived above, can now be used to compute the UAV's flight path. The goal is to visit the most threatening grid cells, and either detect or rediscover icebergs. In order for the iceberg tracking to be as accurate as possible, it is important to get updated measurements relatively often. It is therefore crucial to minimize the flying time for the UAV. To do so, a minimum cost path has to be computed. As in [6], this will be done by solving a Travelling Salesman problem.

3.5.1 Travelling Salesman Problem

In the Travelling Salesman Problem (TSP), a salesman wishes to make a *tour*, including n vertices, visiting each vertex exactly once and finish at the vertex he started from. For every pair of vertices (i,j) there is a corresponding edge with an integer cost $c(i, j)$ associated to it. The salesman wants to make the tour in such a way that the total cost along the edges is minimized. The analogy to the problem of the salesman can be understood by viewing these vertices as cities, and the cost along the edges as travel time from one city to another. In this way the problem of the travelling salesman becomes to minimize his travel time, under the condition that he is visiting every city exactly once, ending up where he started. An example of this is illustrated in Figure 3.10, where the optimal path is $\{a, c, b, d, e, a\}$.

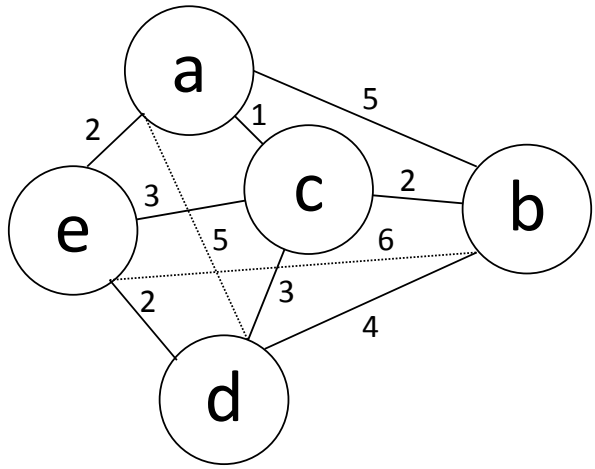

Figure 3.10: An illustration of the Travelling Salesman Problem with $V = \{a, b, c, d, e\}$.

The Traveling Salesman Problem is NP-complete [20]. NP is a complexity class used to describe certain types of decision problems. In general, we classify problems either as solvable in polynomial time, or not. If they are, we refer to them as tractable or easy problems, while the problems that are not solvable in polynomial time are referred to as intractable, or hard. NP-complete problems are special because they do not belong to any of these classes. In fact its status is still unknown. No one has been able to come up with a polynomial time algorithm that solves an NP-complete problem. On the other hand, no one has yet been able to prove that no polynomial time algorithm exist for the problems. Examples of problems known to be NP-complete are "Clique", "Subset Sum", "Vertex Cover", "Hamilton Cycle" and the "Travelling Salesman Problem". A further description of these problems, NP-completeness and approximate solutions can be found in [20].

Since a polynomial time algorithm yet has to be discovered for NP-complete problems, the run time for the TSP increases exponentially with n and has a worst case run time of $O(n^2 2^n)$. Because of this, the number of vertices we are optimizing over has to be kept at a minimum. Several approximating algorithms exist, which can solve problems of much larger dimension, but for the current application and simulation a run time of $O(n^2 2^n)$ will be sufficient.

3.5.2 Computing the cost

In this work, the set V of vertices will be the grid cells that are to be included in the path. The cost d_{ij} of flying from grid cell i to j will be computed as the euclidean distance between the grid cells centre. In this way we minimize the

total flying time for the UAV.

Since the TSP has high time complexity, we can only select some of the elements in $\mathbf{Y}$ to make up the set V of vertices we are optimizing over. To do this we introduce the variable δ.

δ will be used as the upper limit for the number of vertices V can include. This means that if the number of n vertices in V is greater than the number of vertices the TSP solver can handle, the vertices with the lowest y_{xy} will be removed.

This gives us a set V of the δ grid cells in $\mathbf{Y}$ that are considered most important to visit. Corresponding to V, we define the cost matrix $\mathbf{D}$, which contains the computed cost between the grid cells in V.

3.5.3 TSP algorithm

The Travelling Salesman Problem has been solved using the Held-Karp algorithm, a dynamic programming algorithm proposed in 1962 by Richard E. Bellman [22]. Dynamic programming solves problems by combining solutions to already solved subproblems [21]. In contrast to similar approaches like divide and conquer algorithms, dynamic programming is applicable when the subproblems are dependent, that is when subproblems share subsubproblems. The complete algorithm is formulated in Bellman's paper from 1962 [22]. In short it consist of the following steps:

1. Define $f(i; j_1, j_2, \cdots, j_k)$ = length of path of minimum length from i to a fixed vertex 0, which passes once and only once through each of the remaining k unvisited cities $j_1, j_2, \cdots, j_k$. Thus, if we obtain $f(0; j_1, j_2, \cdots, j_n)$, and a path which has this length, the problem can be solved.

2. Let us also define d_{ij} to be the distance between the ith and jth cities. Based on what is stated in 1., this gives an iterative procedure:

$$f(i; j_1, j_2, \cdots, j_k) = \min_{1 \leq m \leq k} \{d_{ij_m} + f(i; j_1, j_2, \cdots, j_{m-1}, j_{m+1}, \cdots, j_k)\} \tag{3.51}$$

   which can be initiated through the use of the known function

$$f(i; j) = d_{ij} + d_{j0}$$

   from which we obtain $f(i; j_1, j_2)$, which in turn, through (3.51) yields $f(i; j_1, j_2)$, and so until $f(0; j_1, j_2, \cdots, j_n)$ is obtained.

3. By storing the sequence of values of m which minimizes the expression on the right hand side of (3.51), this gives the desired minimal path.

3.5.4 Complete algorithm

What is presented in this chapter, can now be combined to a complete algorithm for search and track using a single UAV. In pseudo-code it can be stated in the following way:

while *mission is still active* **do**
  get updated weather data from the area;
  compute k_{tr} from (3.44);
  compute u_{tr} from (3.24);
  based on $\mathbf{X}$, determine $\mathbf{Y}$ by using (3.49);
  find V from $\mathbf{Y}$ by using δ;
  compute the cost matrix $\mathbf{D}$ for the vertices in V;
  solve for the optimal path using the Held-Karp algorithm with $\mathbf{D}$;
  send the waypoints in the optimal path, to the UAV;
  wait for the UAV to visit all the waypoints;
  update the Kalman filter with iceberg measurements from the UAV;
end

A graphical representation of the algorithm can be seen in Figure 3.11. Updated weather data and UAV measurements are the inputs, while the output is mission waypoints for the UAV.

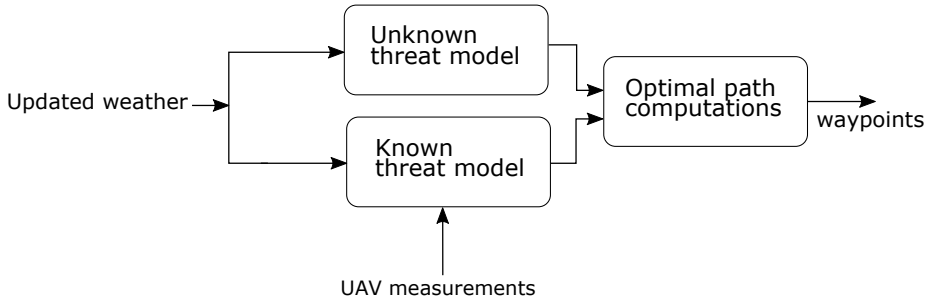

Figure 3.11: Graphical representation of the search and track algorithm and its information flow.

Chapter 4

Simulator Development

In order to test the combined search and track algorithm, a simulator has been developed. This has been done in MATLAB. This chapter serves as an explanation to how the simulator is built, and how the models presented in Chapter 3 has been embedded into the system. Some preparatory simulations are also included.

4.1 Assumptions and simplifications

To be able to simulate the algorithm without having to focus too much on models and dynamics that are partly irrelevant, a few assumptions and simplifications has been made. These are as follows:

- The UAV will be modelled as a point in space, assuming we already have an implemented controller taking care of the dynamics.
- We assume we have a sensor, e.g. a camera, that is able to classify an object to be an iceberg with 100% accuracy. However, there will still be some uncertainty on the measurements that are made.
- The UAV's sight covers at least half a grid cell (2.5 km). This can be achieved by adjusting the altitude and sensor position, or by flying zigzag inside the grid cell.
- The UAV is able to measure an iceberg's position with and accuracy of ± 0.5m , and the velocity with an accuracy of ± 0.05 m/s.
- Waypoints provided to the UAV, are always located in the centre of a grid cell. Since the sight is assumed to cover 2.5 km, every iceberg inside this grid cell will be detected.

4.2 Simulator architecture

The simulator consists of two main parts: One part simulating the iceberg trajectories, and one part simulating the search and track algorithm. These parts are unaware of each other, and runs in parallel during a simulation. To illustrate the progress in each of these parts, a series of maps will be used.

These maps are a representation of the simulation environment and have been modelled in geographic coordinates. Since the icebergs' position and velocity will be expressed in NED (North-East-Down), the MATLAB functions *geodetic2ned*() and *ned2geodetic*() have been used to convert between the coordinate systems. The maps are outlined below:

Mission map

The mission map has been modelled as a 20×20 grid map, and shows the progress of the current mission. This includes the UAV's flying path, detected icebergs and where the vessel is located. An example of this during a simulated mission can be seen in Figure 4.1. Here, the black triangle shows the UAV's current location, and the small red dots the path travelled so far. Detected icebergs are illustrated as black dots. From the figure we see that two icebergs have been detected so far. Further, the vessel in operation is illustrated as a red dot in the maps centre.

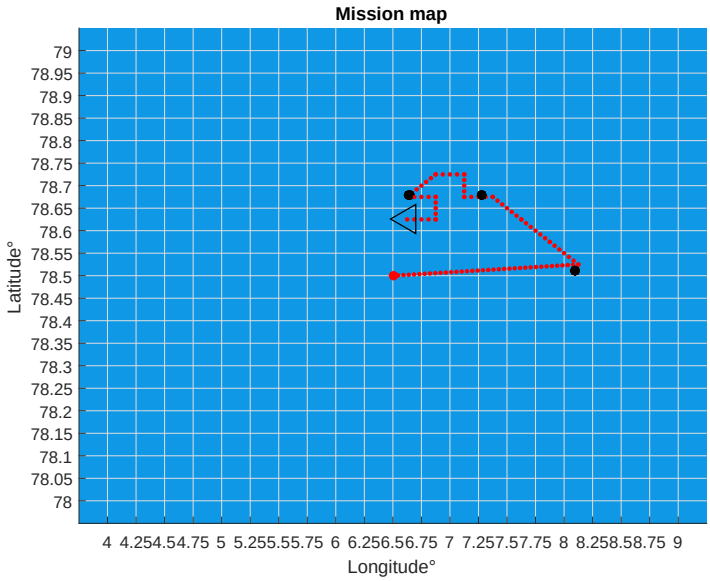

Figure 4.1: An example of how the UAV mission map could look during a mission.

The Mission map covers the operational area. As mentioned in Chapter 2, the size of this area is approximately 100×100 km^2. Each grid cell in the map, therefore covers approximately 5 km^2 of the operation area. The operation will take place west of Svalbard at the coordinates $78° - 79°$ latitude and $4° - 9°$ longitude. Because the operation takes place far north, $1°$ latitude ≈ 5 ° longitude. One grid cell therefore covers approximately $0.25°$ longitude and $0.05°$ latitude.

Iceberg Drift map

The Iceberg Drift map is a mapping of the iceberg trajectories, i.e., the true states of the icebergs during a simulation. The iceberg drift area is modelled as a 28×28 grid map, with the operational area in its centre. Prior to every simulation, n_i icebergs are generated at random as explained in Chapter 3, Section 3.1. These are placed randomly inside the Iceberg drift map, and the initial velocity is set to zero. Because of this, the icebergs need some time to adapt their velocity to the wind and current around them. Before the simulation starts, the icebergs are therefore given some time to move around in the Iceberg drift area and adapt to the weather.

An example of the iceberg drift map during a simulation can be seen in Figure 4.2. Here, each of the coloured dotted lines illustrates an iceberg trajectory. The blue arrows shows the velocity of the icebergs at the end of the simulation. The area inside the square rectangle is the operational area and has the same size as the mission map.

By making the Iceberg Drift map larger than the operational area, one allows the icebergs to drift in and out of the area, which is more realistic than simply deleting them if they leave the operational area. In a realistic scenario, new icebergs should also be able to enter the operational area during a simulation. This could be achieved by simply adding new icebergs to the edge of the operational area, but since the initial velocity is zero, the iceberg would need some time to get a velocity and acceleration that is realistic, considering the wind and current in the area. By adding the new icebergs to the edge of the iceberg drift area instead, this adapting process will happen before it potentially enters the operational area. In that way, making the iceberg drift area larger than the operational area, is beneficial in two ways.

If we study Figure 4.2 we see that one iceberg has crossed the green border (purple, moving towards south) and has entered the operational area. We also observe that three new icebergs has been added to the edge of the iceberg drift map during the simulation. These have no blue arrow, since the initial velocity is zero.

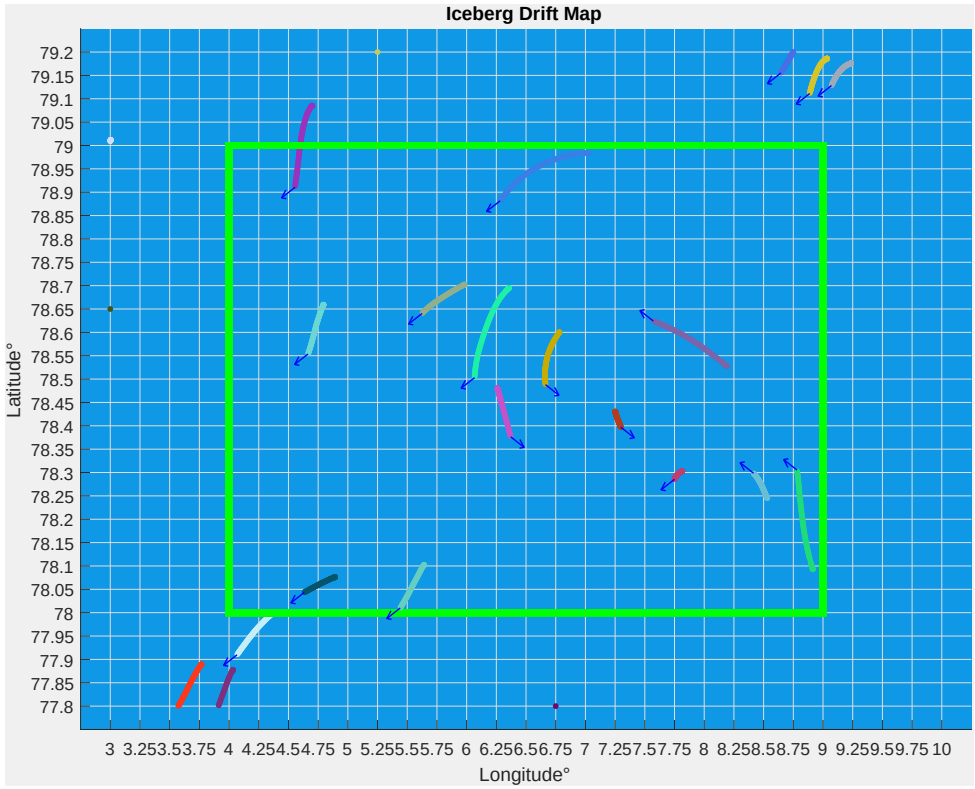

Figure 4.2: Example of the Iceberg Drift map. The arrows show the velocity direction of the icebergs at the end of the simulation. Icebergs without an arrow have either drifted out, or has a velocity component equal to zero.

Iceberg Drift vs. Iceberg Estimates map

During a simulation, the Iceberg Drift vs. Iceberg Estimates map will be used to visualize where the search and track algorithm estimates the icebergs to be, compared to where they actually are. Since we are searching and tracking in the operational area, this map covers the same area as the mission map. Figure 4.3 illustrates an example of the Iceberg Drift vs. Iceberg Estimate map, during a simulation. Here, the white dots show the true trajectory of the icebergs while the red dots illustrate the estimated trajectory of the detected icebergs. As we can see, one iceberg has been detected and is being tracked. The ellipse around the estimate, illustrates the covariance, and will be explained in more detail in section 4.4.

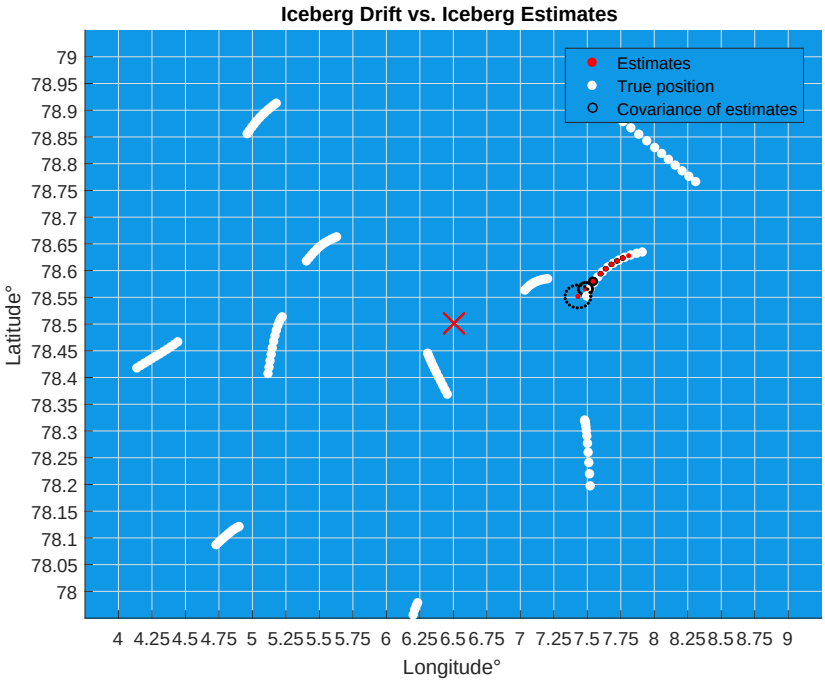

Figure 4.3: An example of the Iceberg Drift vs. Iceberg Estimate map. The red cross shows the vessel position.

Weather maps

The last two maps used in this simulation, are the Wind and Current Velocity Directions map, and the 2%-rule Weather map. An example of these maps are shown in Figure 4.4 and 4.5. The maps are zoomed in on a small area of the operational area, to make it easier to see the arrows. In full scale, both maps cover the same area as the Iceberg Drift map.

Figure 4.4 illustrates the wind and current direction components in the weather data. Since the wind components are much larger in magnitude than the current components, only the direction is plotted. The yellow arrows are meant to indicate that both the current and wind are moving in the same direction.

Figure 4.5 shows the velocity components computed by the 2%-rule. These velocities will be used both by the Kalman filter and the unknown threat model, to compute k_{tr} and u_{tr}.

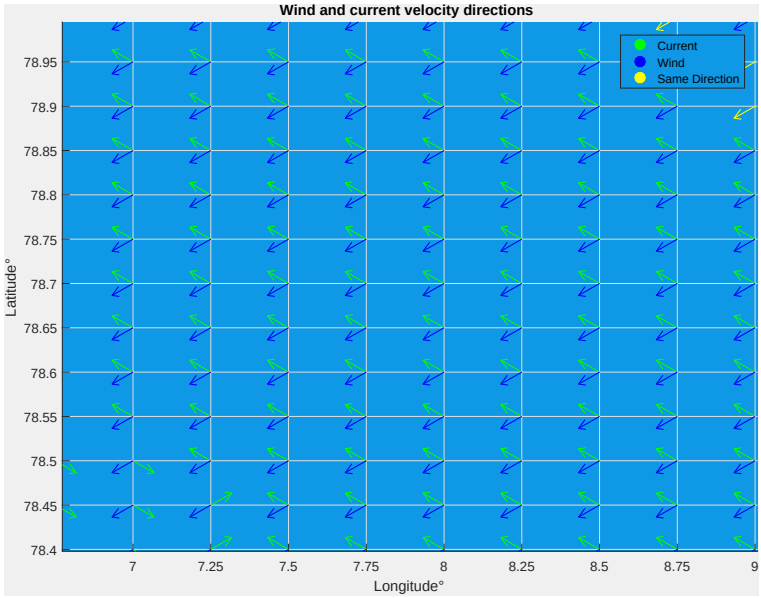

Figure 4.4: Shows the wind and current directions after 2 hours. Note that these arrows do not show magnitude.

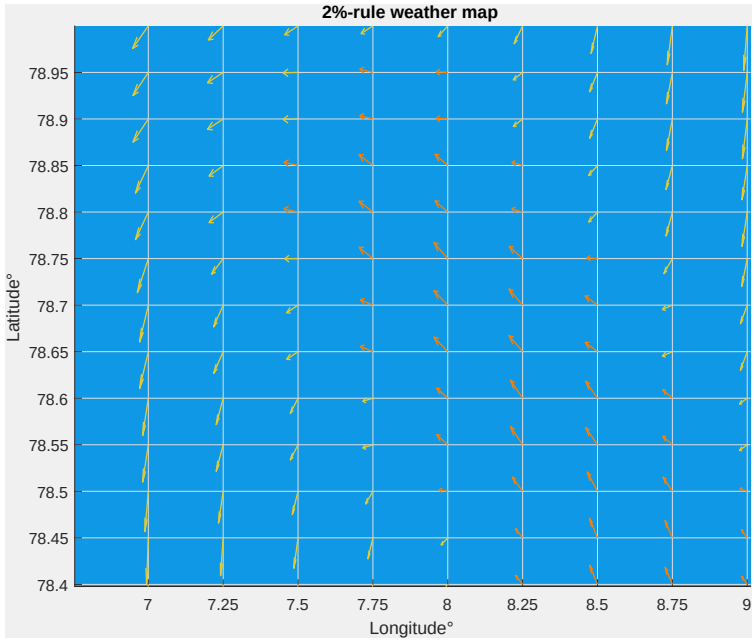

Figure 4.5: Shows the 2%-rule weather map after 2 hours.

4.3 Iceberg detection

When the first UAV mission is started, the number of detected icebergs is zero. Because of this, the first missions before any icebergs are detected, only take the unknown threat (u_{tr}) into account, when planning the path for the UAV. The path planning part of the algorithm will provide the UAV with waypoints. In the simulation, the UAV will fly from waypoint to waypoint, looking for icebergs.

To simulate that the UAV discovers an iceberg, information from the iceberg drift model will be used. The detection process can be described in three steps:

1. During a mission, the UAV constantly updates its own position. When this happens, the updated position is compared with the true states of the icebergs in the iceberg drift model.

2. If the position of the UAV falls within an iceberg's position ± 2.5 km, this means that we have detected an iceberg. Its position and velocity will be "measured" and used for tracking. To account for some uncertainties in the measurements, random errors in the range ± 5 m for position and ± 0.05 m/s for velocity, are added.

3. The iceberg has been detected. The last thing to do is to mark it as detected on the mission map. This is done by plotting a black dot at the position it was discovered.

4.4 Kalman filter

When an iceberg has been detected, the measurement is sent to the Kalman filter. To determine if we have located an iceberg we are already tracking, or if this is a new detection, the measurement has to be compared to the estimated positions of the icebergs we are tracking.

Each tracked iceberg has a set of estimated positions and velocities. Every time the Kalman filter is updated, we do not know how far into the future we need the next estimate. Therefore, a series of estimates are computed, each for different times. When the UAV detects an iceberg and measures its position and velocity, the measurement time is saved. When comparing this measurement to the estimated positions of the icebergs we are tracking, we can compare it to the estimates made for that exact time. If the measurement matches any of the estimated positions with ± 2.5 km, the measurement is classified as a rediscovery and the measurement is saved as an update for the matching estimate. If not, the measurement is saved as a new discovery and a new instance is created in the Kalman filter.

Since the waypoints provided to the UAV always are in the centre of a grid cell, the measurement has to be classified as a rediscovery when the difference between estimate and measurement is less than 2.5 km. Consider a grid cell included in the path because of a threat estimated to be located on the edge of that grid cell. With a range less than 2.5 km, the detection of this iceberg would have been treated as new discovery, even if the estimate was correct. A range of 2.5 km is therefore necessary in order for the detection to work as it should.

In order for the Kalman filter to provide us with as accurate estimates as possible, the **Q** and **R** matrices had to be tuned properly. Here, **Q** represents the process noise, i.e. the uncertainty in the model, while **R** represents the uncertainty in the measurements. As mentioned in Chapter 3, Section 3.2.2, the 2%-model does not have any information about the icebergs' size, shape and mass. Since these are important parameters when estimating the icebergs' drift, the model uncertainty must be set quite high. The measurements, on the other hand, are based on observations of the actual iceberg, which should be trusted much more than the model itself. **R** must therefore be chosen a lot smaller than **Q**. After some tuning the following matrices were used:

$$Q = \begin{bmatrix} 50000 & 0 & 0 & 0 \\ 0 & 50000 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}, \quad R = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}, \tag{4.1}$$

In addition to estimating the iceberg states, the Kalman filter keeps track of a covariance matrix, representing the estimate uncertainty, which increases proportional to the time since last measurement. How this could look during a simulation is illustrated in Figure 4.6. The direction of movement is from lower right towards upper left. First, an iceberg is discovered in the lower right corner. A new instance is created in the Kalman filter, and the measurement is used to predict its future state. Prior to the next mission, this estimate is marked as a red dot on the map (most lower right red dot). We see that the ellipse around this point is quite small, indicating a low covariance. The algorithm then decides that the UAV should prioritize to rediscover the iceberg corresponding to this estimate, and the estimated location is added to the UAV path. Arriving at the estimated location, it detects the iceberg, measures its position and velocity and updates the state in the Kalman filter.

If we follow the trajectory to where the estimates starts failing to follow the white (true) path (approximately at (78.45,7.75)), we see that this immediately gives an increased ellipse. After a few missions where it fails to detect the iceberg, the ellipse is so big that a big part of it covers the neighbouring grid cell. Because of this, this grid cell is also given weight in the next mission. By also

visiting this grid cell, the iceberg is rediscovered, the estimate is updated with a new measurement, and the estimate is again close to the white path. We also observe that the ellipse around the estimate decreases for the next estimates, which means that the covariance has been reduced after a new measurement was made.

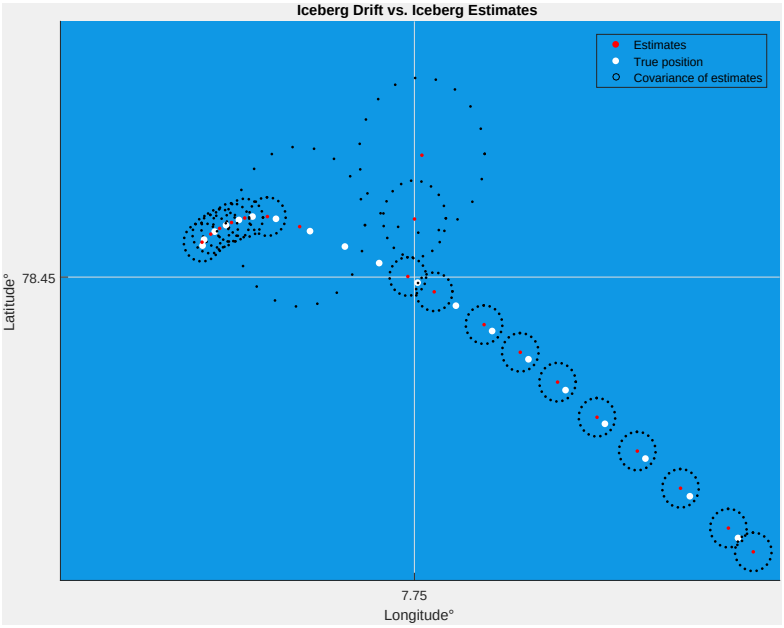

Figure 4.6: An example of how an estimate and its corresponding covariance matrix is illustrated in the simulation. Iceberg moving from lower right an towards north-west.

For the example illustrated in Figure 4.6, the iceberg was rediscovered again after some time. This is not always the case. To avoid that we end up with estimates that have a covariance matrix growing to infinity, the estimate is removed from the Kalman filter if the covariance exceeds a certain threshold. Since the path has a limited amount of nodes that can be included, this opens for other grid cells to be checked for threats. An example of a case where the Kalman filter loses track of the iceberg is illustrated in Figure 4.7. We see that the estimate starts to deviate from the actual position, leading to an increased ellipse. At approximately (78.34,6.68) the covariance has become so big that the estimate is removed from the Kalman filter. This can be seen by comparing the white trajectory and the red trajectory, and noting that the red trajectory suddenly stops.

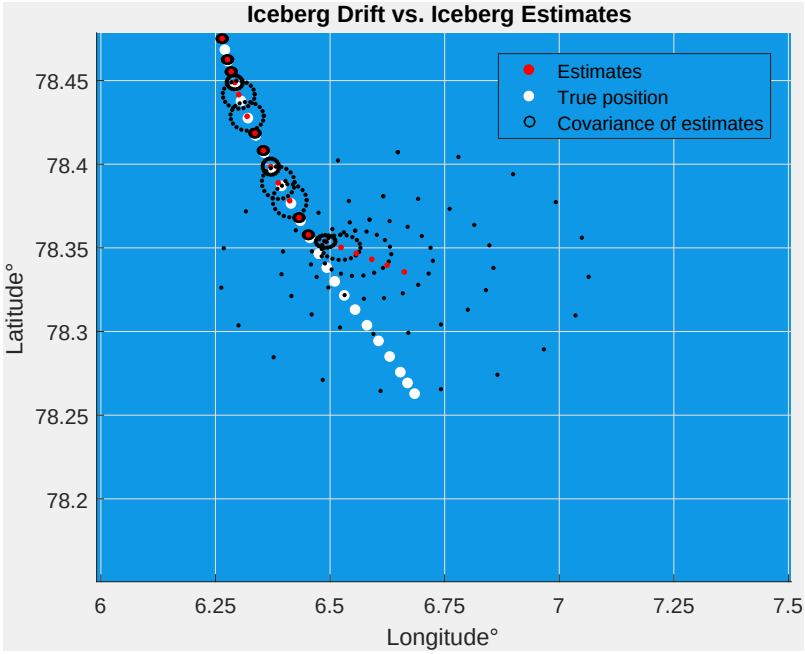

Figure 4.7: An example where the Kalman filter loses track of an iceberg. Iceberg moving southwards.

4.5 Finding a suitable value for δ

Prior to every mission an optimal path will be computed based on the threat each grid cell poses to the vessel in operation. As mentioned in Chapter 3.5, section 3.5.2, the TSP has high time complexity, which limits the set of grid cells one can visit each mission. Therefore, there is a maximum number of nodes, δ, that can be included in the path. Before starting the simulations we need to find the δ that suits our application the best.

To do this, the system was simulated with different values of δ. The result of this can be seen in Table 4.1. We see that the run-time increases exponentially with δ.

If the algorithm was to be tested in a real environment with a UAV, one could tolerate a run-time of a few minutes, considering that an average UAV-mission would take over an hour. Therefore, one would probably choose δ to be either 16 or 17, giving a computation time of approximately 2 and 10 minutes, respectively.

In the simulations, the time has been speeded up, giving an average mission time of only a few seconds. Since we cannot speed up the computation time of the TSP-DP algorithm, we have to decrease δ, to not slow down the simulation

drastically. Therefore, $\delta = 14$ has been used as the maximum accepted value throughout this thesis.

$\boldsymbol{\delta}$	**Run-time** (seconds)
10	0.0974
11	0.2179
12	0.5858
13	1.9548
14	6.8024
15	26.4022
16	114.4933
17	580.1254
18	3780.6123

Table 4.1: The TSP-DP algorithm run-time for different δ

4.6 Complete system

The complete system, and the information flow, can be seen in Figure 4.8. One cycle can be explained in 4 steps:

1. The first thing that happens, is that the weather data module reads the simulation time (0 at initialization), and downloads the wind and ocean current corresponding to this time.

2. The data is then fed to both the iceberg drift model and the search and track algorithm. The iceberg drift model uses the data to update the iceberg states. At the same time, the search and track algorithm uses the weather data and the Kalman filter, to find the most threatening grid cells in the grid map. The optimal path is computed, and waypoints are sent, together with the iceberg states, to the "UAV mission simulation" module.

3. At the "UAV mission simulation" module, a UAV mission is simulated. The UAV flies the path made up by the waypoints. While flying, it uses the iceberg states to check if an iceberg has been detected. Detected icebergs are stored as measurements.

4. When the mission is completed, the measurements are fed back to the search and track algorithm. Here they are used to update the Kalman filter. The simulation time is updated and fed back to the weather data model, where a new cycle starts.

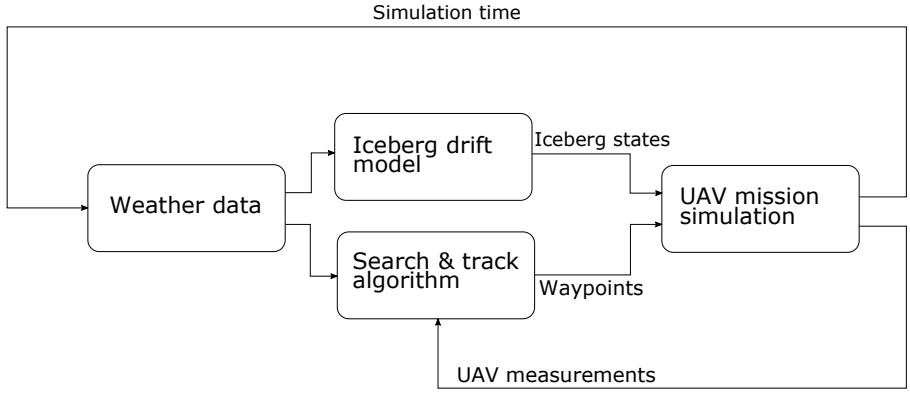

Figure 4.8: Graphical representation of the complete system and its information flow.

Chapter 5

Simulation Results, Analysis and Discussion

The main goal through the simulations is to test the search and track algorithm's performance and robustness. To be able to do this, it is important that all of our models behave as expected. Model testing will therefore be an important part of the simulations.

The simulations have been split into four main parts:

1. Analysing the iceberg drift model.
2. Study the independent behaviour of each of the search and track models.
3. Testing the performance of the combined search and track algorithm.
4. Investigate how integration of satellite and radar data affects the algorithm's behaviour.

In the first part we study the iceberg drift model and how the iceberg drift depends on different factors like size, shape, wind and current. When this has been tested, the search and track algorithm will be examined: In part two we will look at the separate models that make up the algorithm, while we in part three will study the behaviour of the complete algorithm. Its performance will be tested under different conditions, decided by parameters such as ice density wind and ocean currents. In the fourth part we study what happens if we include information from satellite data and ship radars to the algorithm.

5.1 Iceberg drift model

In order to test the search and track algorithm through simulations, a realistic iceberg drift model is important. This will be tested through the following simulations.

5.1.1 Important factors for iceberg drift

As mentioned in Chapter 3, Section 3.2.3, iceberg drift is hard to forecast and factors like shape, size, mass wind and ocean current affect its dynamics. Through the following simulations we will take a closer look at which factors that are important for the iceberg drift in our model.

To do this, we will simulate iceberg drift for 8 different icebergs. They will all start at the same position, but their size, shape and mass will be generated at random. Since they are starting a the same position they will also be influenced by the same external forces. The iceberg drift was simulated for 97 hours ($\approx$ 4 days). The result can be seen in Figure 5.1. Each of the coloured trajectories illustrates the drift of one iceberg. We see that all of the eight trajectories are different from the others. At the beginning, starting at the same position, they all move in a similar direction, but with different speed and curvature. After a while, the differences increases, and some of the icebergs start to move in opposite directions.

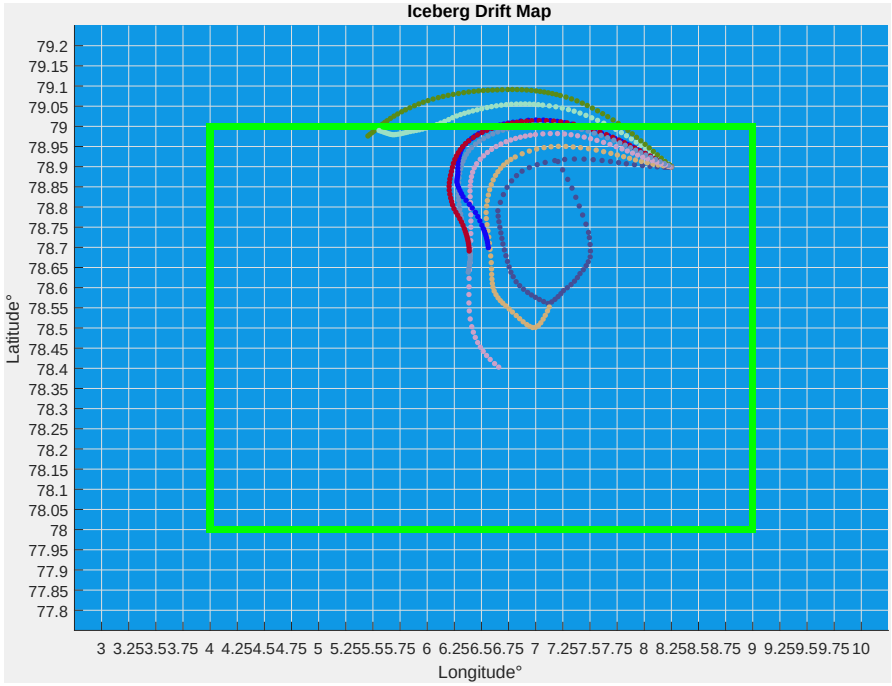

Figure 5.1: Iceberg trajectory for 8 icebergs after simulating in 97 hours (4 days). The area inside the green square is the operation area.

	End pos	**Keel(m)**	**Sail(m)**	**Mass(kg)**	**Shape**	$\mathbf{C}_w$	$\mathbf{C}_a$
Green	(78.98,5.53)	72.1	24.89	4.917×10^8	2	2.14	1.51
Light Blue	(78.99,5.56)	77	26.37	5.99×10^8	2	1.43	0.11
Red	(78.68,6.39)	30.1	11.43	3.578×10^7	3	0.29	0.24
Blue	(78.69,6.56)	26.6	10.24	2.469×10^7	3	0.98	1.14
Grey	(78.63,6.38)	81.2	27.64	7.024^8	1	1.37	2.06
Pink	(78.39,6.7)	88.2	29.76	9.001×10^8	2	1.13	1.42
Orange	(78.56,7.14)	55.3	19.64	2.219×10^8	3	0.51	2.23
Purple	(78.94,7.14)	9.8	4.21	1.23×10^6	2	0.48	0.8

Shape: Rectangular = 1, Triangular 2, Elliptic = 3

Table 5.1: Iceberg properties for the 8 icebergs in Figure 5.1

Table 5.1 shows some of the most important properties for each iceberg. The icebergs are listed based on how far north their start trajectories lie, i.e. green is at the top of the list and purple is at the bottom. Their end position and colour are also included to make it easier to connect the trajectories to the data in the table.

If we compare the trajectories in Figure 5.1 with the properties in Table 5.1, we see that both of the trajectories that differ the most from the rest (green and light blue) have a triangular shape. This could indicate that the keel shape is important for the iceberg's drift. On the other hand, the iceberg with the pink trajectory also has a triangular keel shape. Since this trajectory is more similar to the others, it seems like other factors are influencing the drift as well. It is therefore difficult to draw any conclusions about the keel shape from this.

When it comes to weight, we see that the lightest iceberg (purple) has the most curved trajectory, while heaviest one (pink) has a more straight trajectory towards south. Since more heavy icebergs have a keel that goes deeper, these icebergs are more affected by the ocean current. Since the heaviest iceberg (pink) has moved the furthest south, this could indicate that the current force points southward. Figure 5.2 shows the current and wind velocity components after 97 hours. Since the wind force is much larger than the current force in magnitude, only the directions are plotted. The green arrows show the average current velocity direction, while the blue arrays show the wind velocity direction. The yellow arrays are for the special case when wind and current directions are the same. From Figure 5.2 we see that the current is indeed going in a southward direction in the pink iceberg's area. In the lightest iceberg's (purple) area, the wind is heading to north-east, which corresponds to the direction this iceberg is moving. From this it could seem that the weight and keel length is important for the iceberg trajectory.

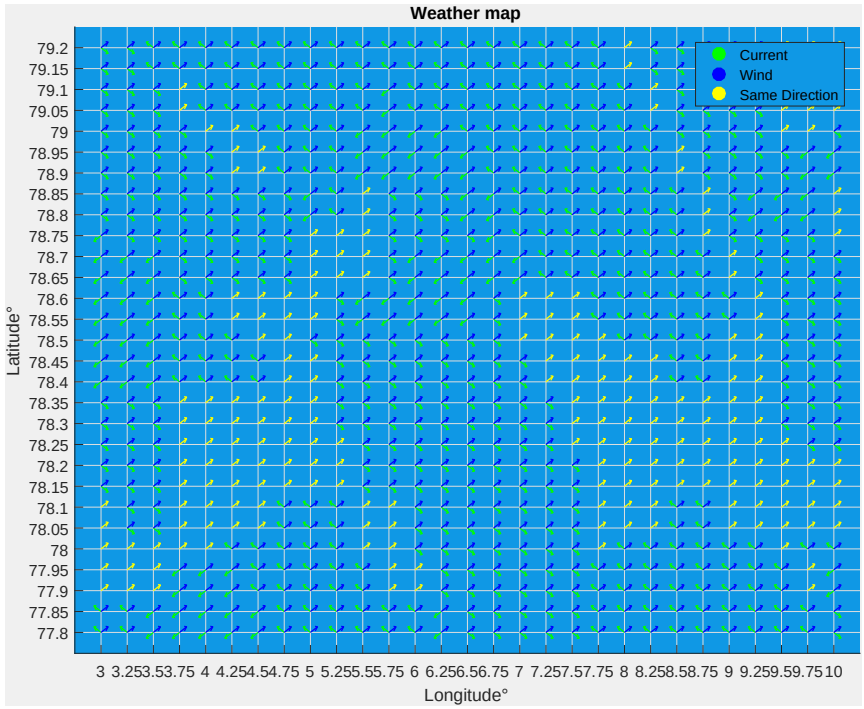

Figure 5.2: Directions of the wind and current velocity components after simulating in 97 hours. Note that these arrows do not show magnitude.

If we go back and study the table and trajectories, what seems to affect the trajectories the most, are the water and air drag coefficients C_w and C_a. The larger the water drag vs. air drag relationship $\frac{C_w}{C_a}$ is, the less the trajectory curves. For the two trajectories that differ the most from the rest we see that $\frac{C_w}{C_a} > 1$. For the other trajectories, which all curves a lot more we see that $\frac{C_w}{C_a} < 1$. With different values for C_w and C_a, we see from equation (3.8) and (3.9) in Chapter 3, Section 3.1, that the ratios for which the icebergs are affected by wind and current are different. If $C_w > C_a$ the computed current force is given more weight than the computed wind force. Figure 5.3 shows the initial current and wind velocity directions. At the position the icebergs start from, we see that the current and wind force are acting in different directions. Since the two trajectories with $\frac{C_w}{C_a} > 1$ are more affected by current than the others, these tend to drift more towards north-west, which is the current velocity direction. Since the wind is heading towards the south-west, we see from the other icebergs that the lower the ratio $\frac{C_w}{C_a}$, the more the wind force cancels out the current force, causing a more straight westward trajectory. This indicates that the water and air drag coefficients seem to have a big impact on the iceberg drift.

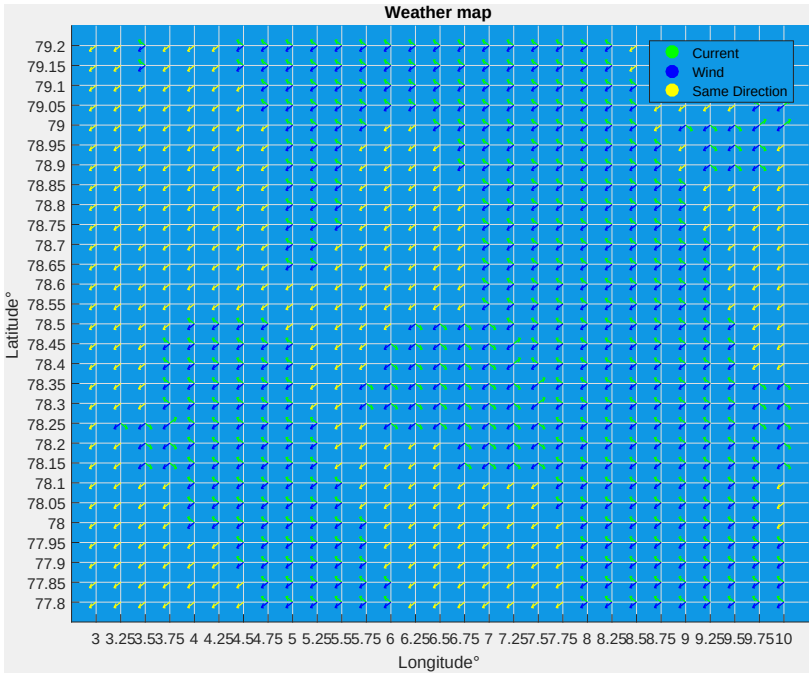

Figure 5.3: Directions of the initial wind and current velocity components. Note that these arrows do not show magnitude.

To further test this, we simulate the same system, but now we put $C_w = C_a = 1.5$ for all icebergs. The result can be seen in Figure 5.4. We now see that the trajectories are more similar, especially in the first part of the simulation. What is interesting here, is that we see three different groups of trajectories. If we study Table 5.1 (ordered in the same way as above), we see that every iceberg in each group has the same keel shape. This could indicate that the iceberg shape actually plays an important role for the iceberg trajectory. Figure 5.5 shows another simulation for 8 icebergs where $C_w = C_a$. Corresponding iceberg properties can be seen in Table 5.3. The same pattern is repeated here: When the keel shape is rectangular, icebergs tend to drift westwards, while icebergs with a triangular keel shape, tend to drift further south. For the icebergs with an elliptic keel shape things are more random, but the trajectories seem somewhat correlated.

Even though there seem to be a pattern, it is hard to draw conclusions from this. Anyway, the trajectories seem to be less random when $C_w = C_a = 1.5$ for all icebergs, indicating that the iceberg shapes affects its drift. The drag coefficients are included to account for the uncertainties in iceberg surface roughness, keel depth and mass. Generating coefficients at random is important to account for these uncertainties, but gives a somewhat random behaviour which is difficult to analyse.

Through these tests and simulations we have seen that iceberg drift depends on more than only ocean current and wind. Factors like weight, keel length and shape all seem to play an important role. This said, it was hard to conclude with anything specific, since the randomized drag coefficients seem to affect the trajectories significantly. We conclude that the icebergs in fact have a quite random behaviour and that the uncertainties in shape, surface roughness, keel depth and mass makes it hard to predict and analyse iceberg dynamics.

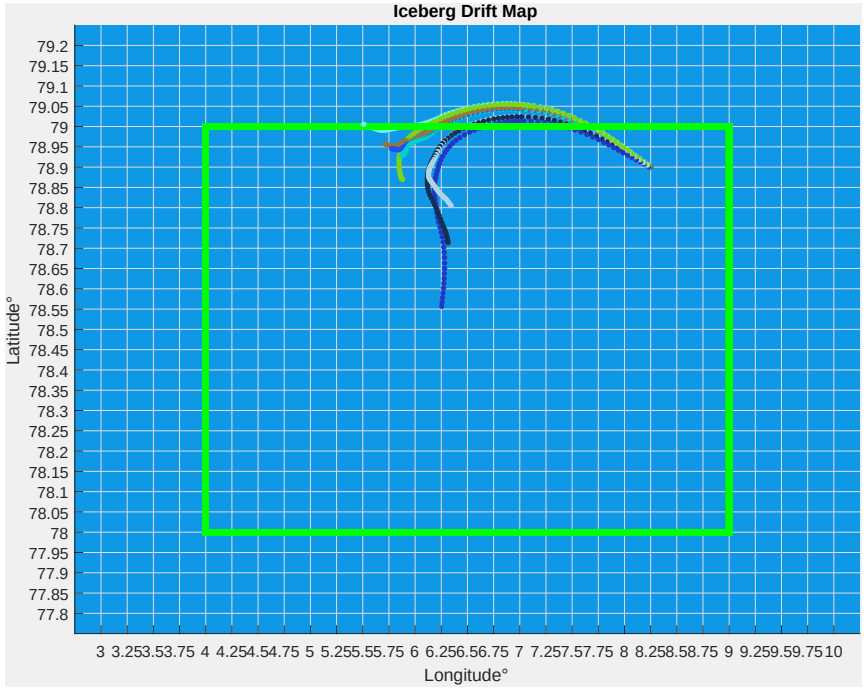

Figure 5.4: Iceberg trajectories for 8 icebergs with $C_w = C_a$. The area inside the green square is the operation area.

	End pos	**Keel(m)**	**Sail(m)**	**Mass(kg)**	**Shape**
Light blue	(79,5.5)	66.5	23.14	3.858×10^8	1
Brown	(78.96,5.71)	65	22.7	3.619×10^8	3
Blue 1	(78.95,5.77)	61.6	21.62	3.067×10^8	3
Light blue 2	(78.93,5.87)	54.6	19.4	2.135×10^8	3
Green	(78.87,5.88)	94.5	31.64	1.107^9	3
Black	(78.71,6.31)	74.9	25.73	5.512×10^8	2
Grey	(78.8,6.35)	54.6	19.42	2.135×10^8	2
Blue 2	(78.55,6.25)	91.7	30.8	1.012×10^9	2

Table 5.2: Iceberg properties for the 8 icebergs in Figure 5.4

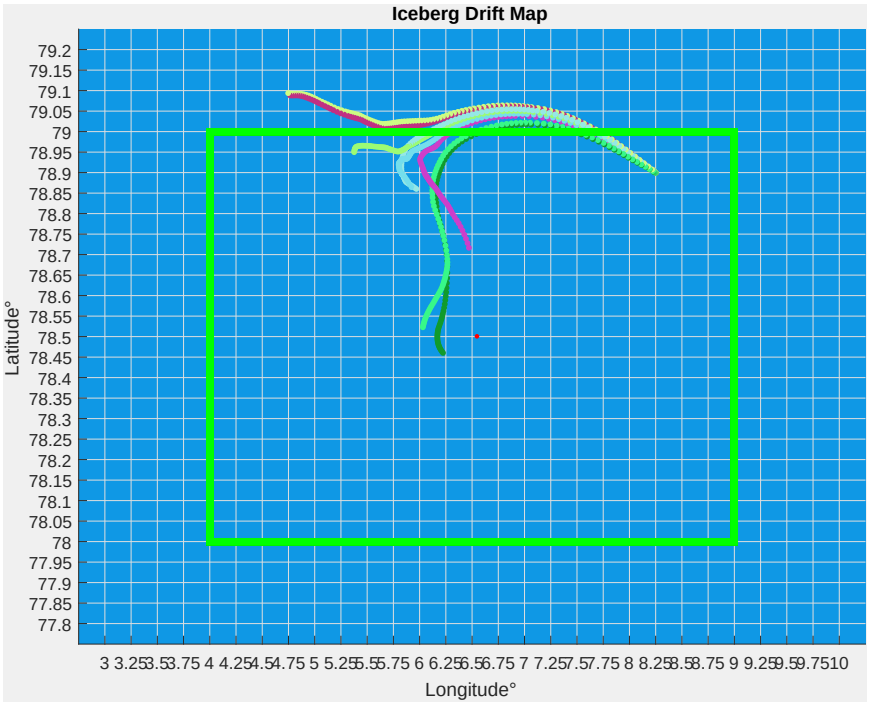

Figure 5.5: Iceberg trajectory simulation with $C_w = C_a$. The area inside the green square is the operation area.

	End pos	**Keel(m)**	**Sail(m)**	**Mass(kg)**	**Shape**
Yellow	(79,1.4.73)	61.6	21.62	3.066×10^8	1
Dark Red	(79.09,4.75)	70	24.22	4.5×10^8	1
Light Green	(78.95,5.37)	74.9	25.73	5.551×10^8	1
Light blue 1	(78.86,5.98)	86.1	29.12	8.373×10^8	3
Light blue 2	(78.86,5.94)	62.3	21.84	3.172^8	3
Purple	(78.71,6.48)	39.2	14.5	7.9×10^7	2
Green	(78.52,6.03)	90.3	30.4	9.66×10^8	2
Dark Green	(78.46,6.23)	97.3	32.47	1.209×10^9	2

Table 5.3: Iceberg properties for the 8 icebergs in Figure 5.5

5.1.2 Analysing the accuracy of the 2%-rule

Before we start testing the search and track algorithm, there is one important aspect that has to be investigated. In the search and track algorithm we do not know the icebergs' shape, size and mass, which we have seen are important factors for the iceberg drift. The best we can do, is to approximate this with a rule which uses the information we have, which is ocean current and wind. For this

we will use the 2% rule. As mentioned in Chapter 3, Section 3.2.2, the 2%-rule assumes that icebergs drift with the mean of the ocean current plus 2% of the wind. In this section we want to investigate how far off this model is, compared to the true iceberg drift.

This will be done by simulating the drift for 5 different icebergs, using the two different models simultaneously. The result of a 27 hour simulation can be seen in Figure 5.6.

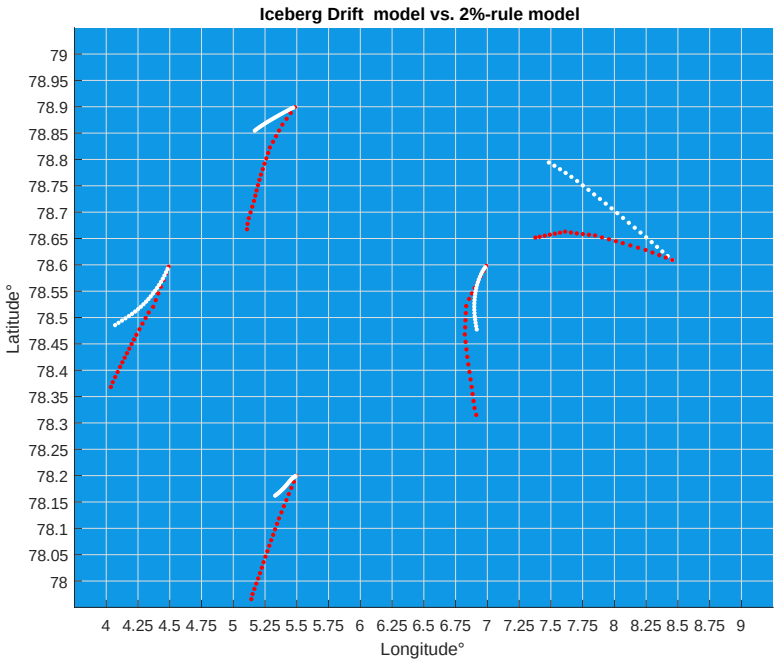

Figure 5.6: Iceberg drift simulation for 5 icebergs using two different models. White shows the true drift, while red shows the drift estimated by the 2%-rule.

We see that there are five icebergs all starting at different positions. The white trajectories illustrate the actual iceberg drift. Each white trajectory has a corresponding red trajectory, starting at the same position. This is the drift predicted by the 2%-rule. From the figure we see that the approximated average trajectories computed by the 2%-rule are similar, but different compared to the true trajectories. What seems to be the biggest difference between the two models is that the predicted speed is much higher than what it actually is. To account for this, the velocity components computed by the 2%-rule was scaled by a factor of 0.7. The same simulation as above, with the scaled 2%rule, can be seen in Figure

5.7.

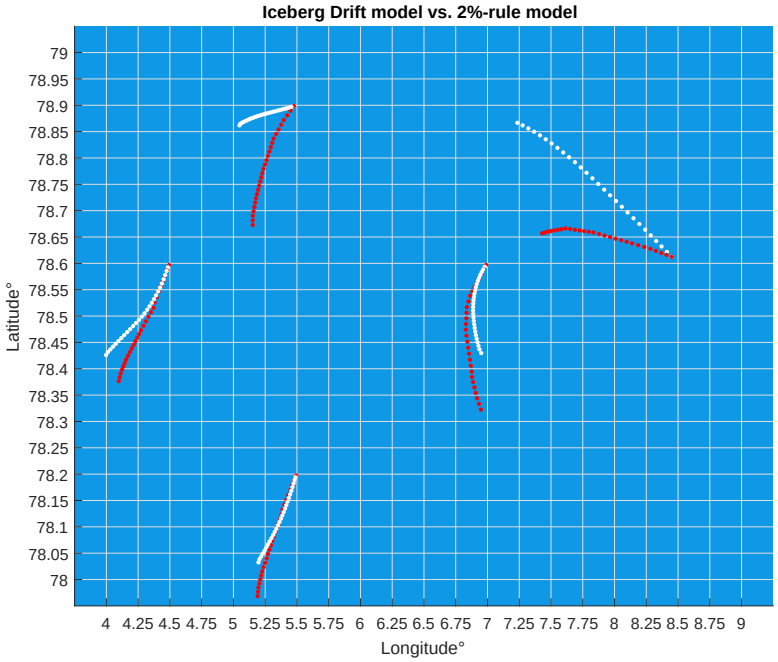

Figure 5.7: Iceberg drift simulation using two different models after speed scaling. White shows the true drift, while red shows the drift estimated by the 2%-rule.

We see that the speed for the predicted trajectories in Figure 5.7 is more similar to the actual iceberg speed, compared to Figure 5.6. The 2%-rule will therefore be slightly changed for the rest of the simulations to:

$$\mathbf{v}_{ice} = 0.7(\bar{\mathbf{v}}_c + 0.02\mathbf{v}_w) \tag{5.1}$$

From Figure 5.7 we see that the deviation between the predicted trajectories and the true trajectories increases with time. For the Kalman filter model, which will use the 2%-rule to estimate the iceberg drift, it is therefore important that the model does not rely on the estimated trajectories for too long. This can be avoided by updating the estimates with measurements of the actual position and velocity. It is therefore crucial to the algorithm that the most threatening icebergs are rediscovered by the UAV and updated as often as possible. If the filter has to rely on the estimates from the 2%-rule for too long, the estimated position will end up being far away from the true position. If this happens, the algorithm could lose track of the iceberg, which could be fatal for the vessel in operation.

5.2 Study of the known and unknown threat model

In this part the two models, that in combination make up the search and track algorithm, will be tested separately. This can be done by setting a or b equal to zero in the prioritizing formula:

$$y_{xy} = ak_{tr} + bu_{tr}$$

The models will be tested by simulating the system under different scenarios. By testing the models separately it will be easier to understand the results when we later are to test the combined model.

5.2.1 Known threat model

The first thing we will look at is the known threat model. To use the search and track algorithm in track mode only, we set $a = 1$ and $b = 0$. Since the search part is turned off and we don't have a way to detect icebergs, the simulation is initialized with 20 detected icebergs. We want to test how the Kalman filter tracks these known threats.

In the first simulation δ, i.e., the maximum number of nodes that could be added to the path, was set to 10 in order to see which grid cells that are prioritized. The Iceberg Drift vs. Iceberg Estimate map prior to the first mission can be seen in Figure 5.8. Since all estimates (red) were initiated at the true position (white) we see that these are located at the same position. These estimates will now be used by the known threat model to determine which grid cells should be a part of the UAV flight path. Figure 5.9 shows the outcome of the first mission. If we compare this with Figure 5.8, we see that the 10 grid cells included in the path, are those with the estimates closest to the vessel (red dot in the centre). The black dots show the icebergs that has been discovered by the UAV on this mission. Further, if we compare the two figures, we see that 9 of the 10 icebergs were discovered at the estimated position. The iceberg estimated to be at grid cell (78.35,5.5) was not discovered. This was probably because the iceberg moved into the neighbouring grid cell while the UAV was flying. The other not discovered icebergs, were too far away to be prioritized in the path.

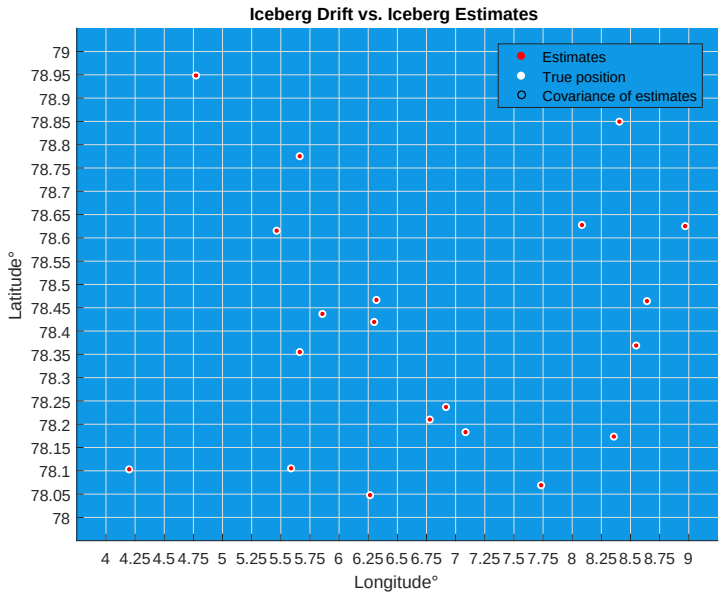

Figure 5.8: Estimates compared to true position of the icebergs before first mission. The vessel is located at (78.5,6.5)

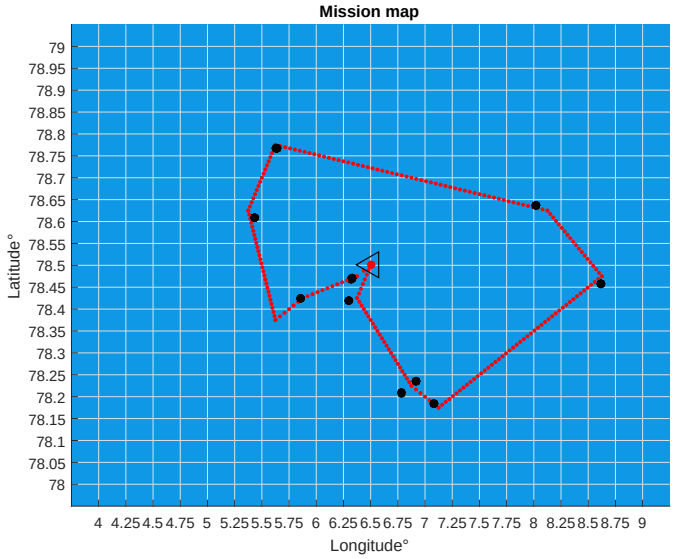

Figure 5.9: First UAV mission when $\delta = 10$.

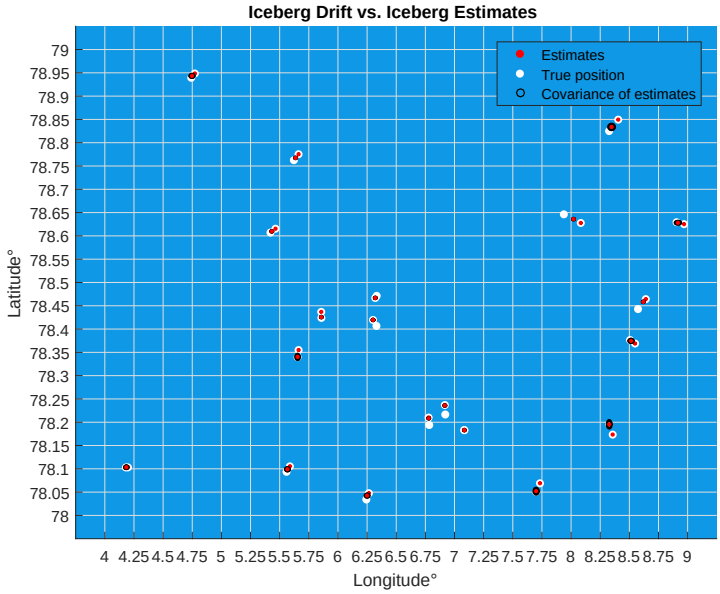

Figure 5.10: Updated estimates after the first mission. The vessel is located at (78.5,6.5).

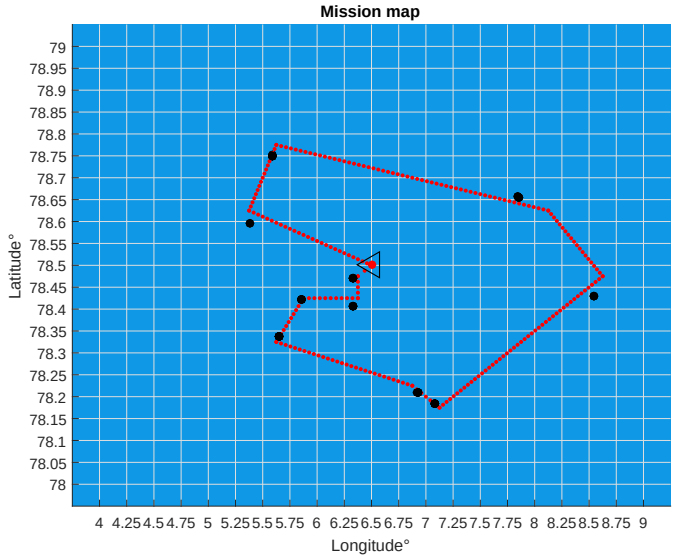

Figure 5.11: Second UAV mission when $\delta = 10$.

After the mission has ended, the estimates are updated. The updated estimates together with the true iceberg position can be seen in Figure 5.10. If we compare this to Figure 5.8, we see that those icebergs that were not rediscovered on the last mission have gotten an increased covariance on the estimate (black circle around estimate). We see that the assumption that one of the icebergs had drifted to the neighbouring grid cell, was correct. We also observe that the new estimate for this iceberg has been moved to the neighbouring grid cell, which indicates that it should be rediscovered on the next mission. Figure 5.11, which illustrates the second UAV mission, shows that this is indeed the case.

To study the effect the covariance matrix has on the iceberg tracking, the system was simulated in track only mode with 3 icebergs. δ was now sat to 14. The Iceberg Drift vs. Iceberg Estimate map after simulating for 42 hours can be seen in Figure 5.12.

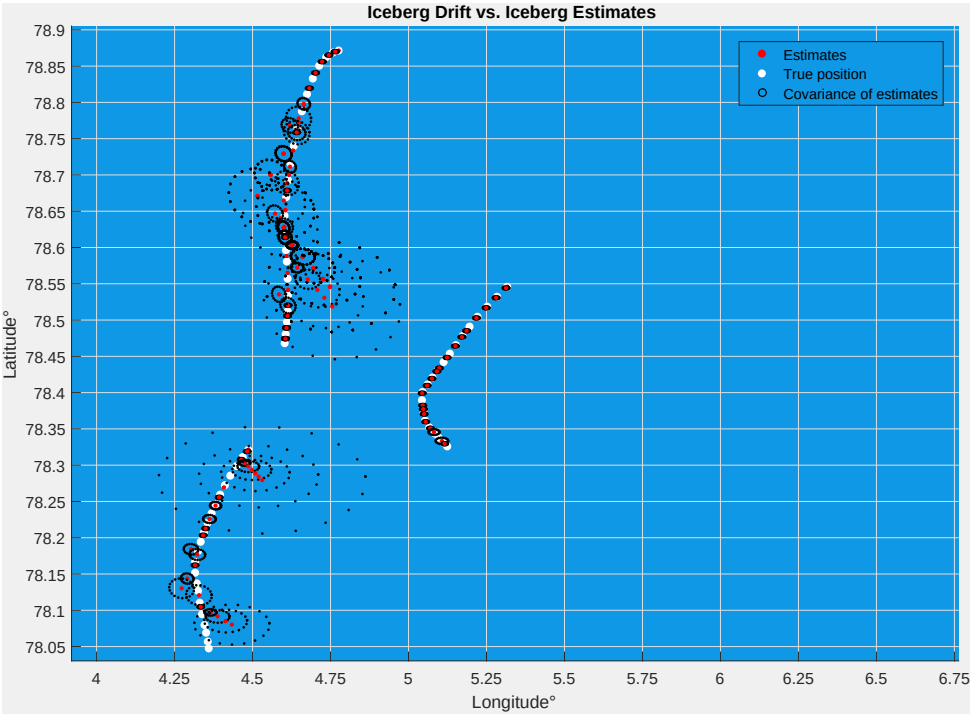

Figure 5.12: The tracking algorithm's behaviour with three icebergs and $\delta = 14$. The vessel is located at (78.5,6.5).

If we study the lower left iceberg we see that the covariance matrix has been quite active in this tracking process. The first two estimates are satisfactory. After this, the drift model and the estimate start to deviate, which increases the

covariance, illustrated with an increased ellipse. Even though the Kalman filter estimates the iceberg to be in the wrong grid cell, the increased covariance matrix makes the UAV also prioritize the grid cells around. In this way the iceberg is rediscovered when the covariance gets big enough. Since the location of the estimate is far away from where the iceberg is rediscovered, the UAV treats this as a discovery of a new iceberg. Because of this, we now have two estimates of the same iceberg in the Kalman filter. This is not a big problem since the covariance of the old estimate will continue to increase until the estimate is deleted from the list of tracked icebergs. In this sense, the old estimate is replaced with the new, recently updated estimate, which is much better suited to be used to track the icebergs' position.

The same process is repeated for the same iceberg later in the simulation, which we can see by following the trajectory downwards. This time it is not able to rediscover it. Also the upper iceberg goes through a similar tracking process, but in this case the Kalman filter does not lose track of it. The iceberg closest to the vessel seem to be estimated well through the whole simulation, and is not reliant on the covariance in order for the Kalman filter to keep track of it.

The results from this simulation indicate that the covariance matrix does its job by helping the tracking algorithm to keep track of the icebergs. For this to work, it is important that the UAV can search several grid cells close to where the iceberg is located. If the number of tracked icebergs is larger than the maximum number of nodes we can visit during a mission, the tracking algorithm's performance will be degraded. An example of this can be seen in Figure 5.13 where 5 icebergs are to be tracked, when only 5 grid cells can be visited. We see that when it starts losing track of two icebergs at the same time, it has to give up tracking the iceberg to the left, which is the one farthest away.

Through the last simulations we have seen that the known threat model has the desired effect on the UAV's search path: Icebergs that are close to the vessel in operation are given higher priority to be rediscovered, compared to the ones far away. On the same time, we have also seen that the algorithm has some limitations when it comes to the number of icebergs we are able to track simultaneously. Since the maximum number of nodes we can have in our path is 14, it is important that we keep the number of icebergs we are tracking lower than this, in order to get the best performance possible. If this is not the case, it will still prioritize the ones that are closest, and in that sense the biggest threat, but the risk of losing track of threatening icebergs will increase.

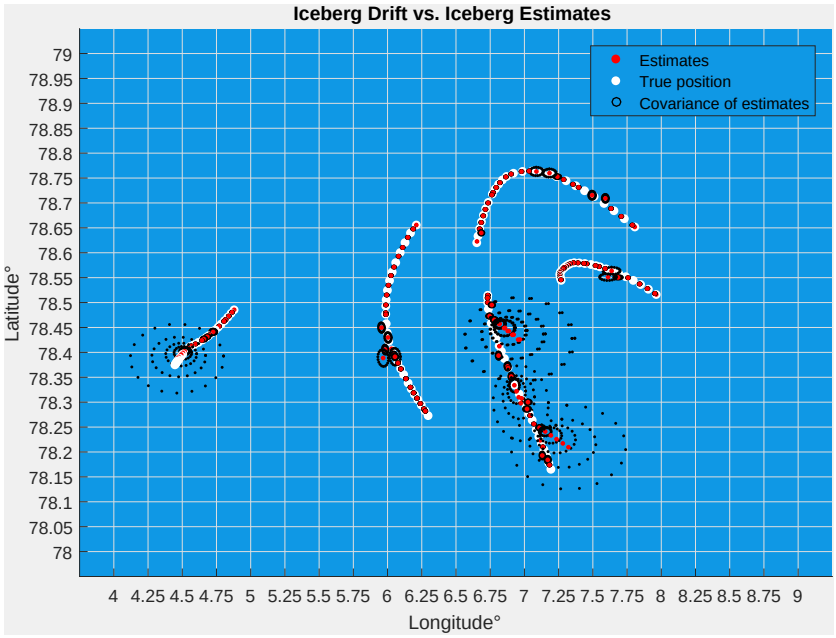

Figure 5.13: Tracking algorithm's behaviour with five icebergs and $\delta = 5$. The vessel is located at (78.5,6.5).

5.2.2 Uknown threat model

We now put $a = 0$ and $b = 1$, to study the unknown threat model and how it affects the UAV's flight path. By quantifying how threatening each grid cell is to the vessel in operation, when only distance and direction of movement is considered, the UAV can be used to detect the most threatening icebergs.

For the next tests and simulations δ was set to 14. The initial iceberg distribution can be seen in Figure 5.14. We see that there are four icebergs that seem threatening, considering their distance from the vessel. If we study Figure 5.16, which shows the initial velocity components computed with the 2%-rule for the grid cells closest to the vessel, only the grid cells east and north-east of the vessel seems to be moving towards it. The outcome of the first mission can be seen in Figure 5.15. As expected, only the grid cells to the east and north-east of the vessel has been searched. Of the four icebergs that seemed threatening based on their distance from the vessel, only the iceberg at (78.54,6.57) has been discovered. The three other icebergs that seemed threatening in Figure 5.14, have not been discovered since they are moving away from the vessel according to Figure 5.16. Also, two icebergs east of the vessel have been discovered. If we study Figure 5.16, we see that the predicted velocity of these icebergs point towards the vessel, which is why these grid cells have been included in the path.

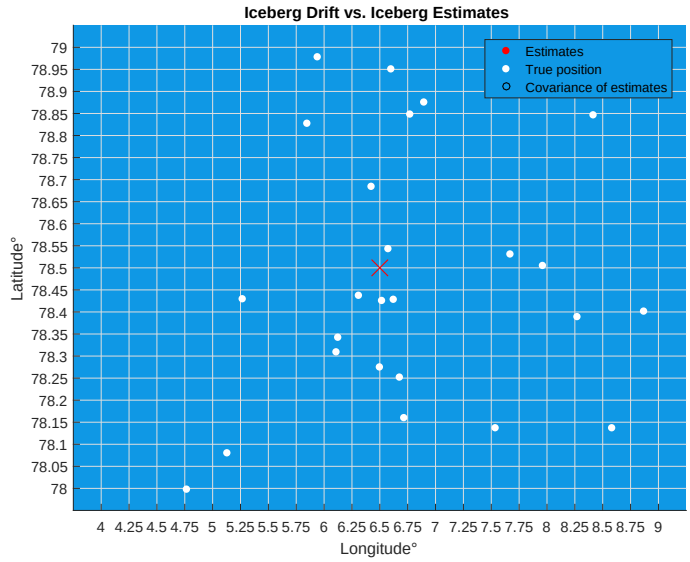

Figure 5.14: Ice distribution before first mission. The red cross shows the vessel position.

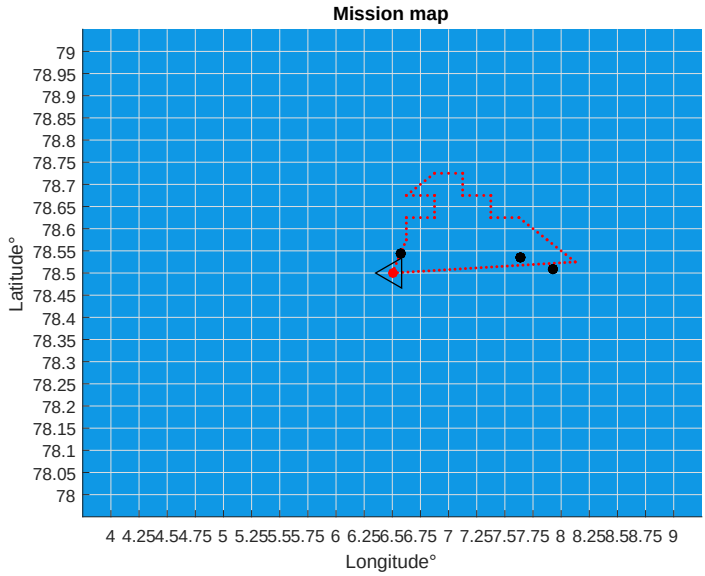

Figure 5.15: Initial UAV mission in search only mode.

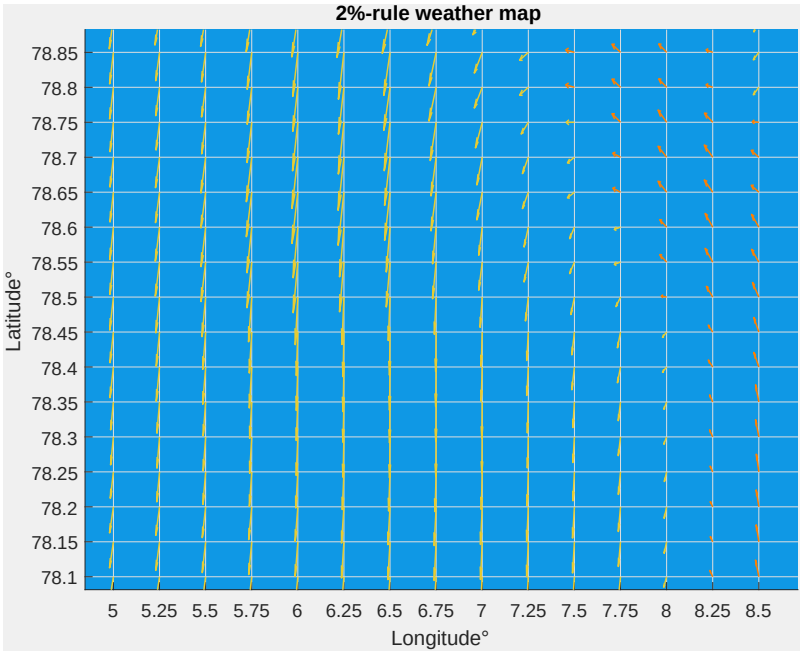

Figure 5.16: Initial velocity components computed by the 2%-rule for the grid cells closest to the vessel.

As we can see, the UAV flies to the most threatening grid cells, prioritizing those that are both close and moving towards the vessel. This is the desired behaviour, but considering the uncertainty in the 2%-rule, grid cells that are closer than 10 km from the vessel, should be searched regardless of the estimated direction of movement. As an example, this would mean that the grid cells containing the icebergs south of the vessel in Figure 5.14, would have been searched. The simplest solution to this is to use a radar on the vessel to search the closest area in a radius of 10 km. In this way we can keep the unknown threat model as it is, and let the known threat model take care of the icebergs discovered by the radar. The possibilities of integrating radar data into the algorithm will be studied more carefully in Section 5.4.

Since the weather data is downloaded for a fixed period of time (01/10/2016-01/11/2016), the computed 2%-rule weather map at time $t = t_k$ in one simulation will be equal to the 2%-rule weather map in another simulation at $t = t_k$. This means that each simulation started at $t = 01/10/2016$ will have an initial mission as in Figure 5.15. The weather data is updated every 6 hours. Figure 5.17 shows how the 2%-rule map looks in an area close to the vessel after simulating for 48 hours, when we started at 01/10/2016. We see that directions have changed: Now the grid cells to the north and north-west of the vessel seem to be the most

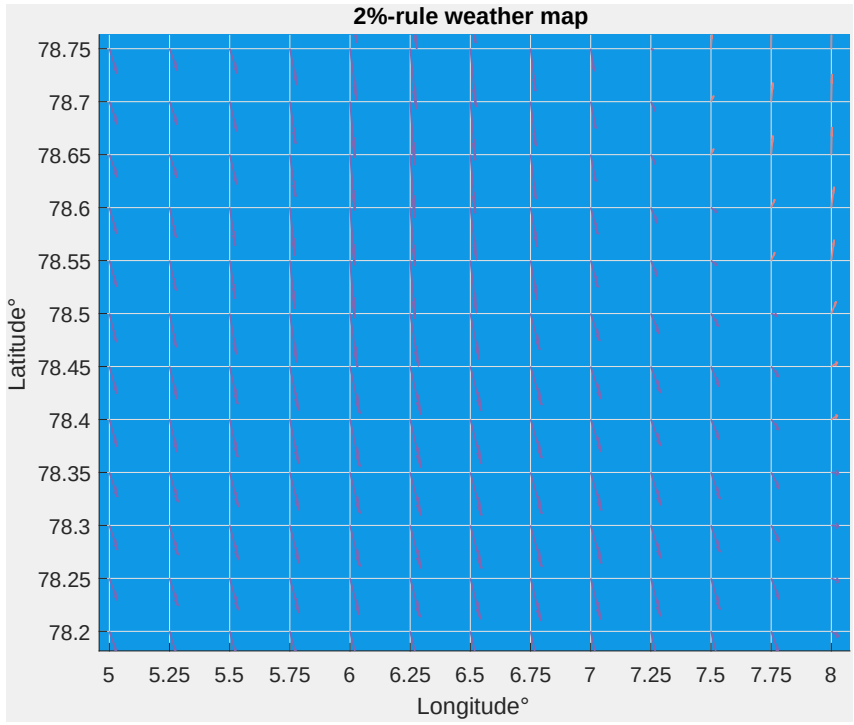

Figure 5.17: Velocity components computed by the 2%-rule for the grid cells closest to the vessel, at 03/10/2016.

threatening. The corresponding mission, when this map is used to compute the path, can be seen in Figure 5.18. As expected, the grid cells that seemed to be the most threatening has been searched. One iceberg has been detected.

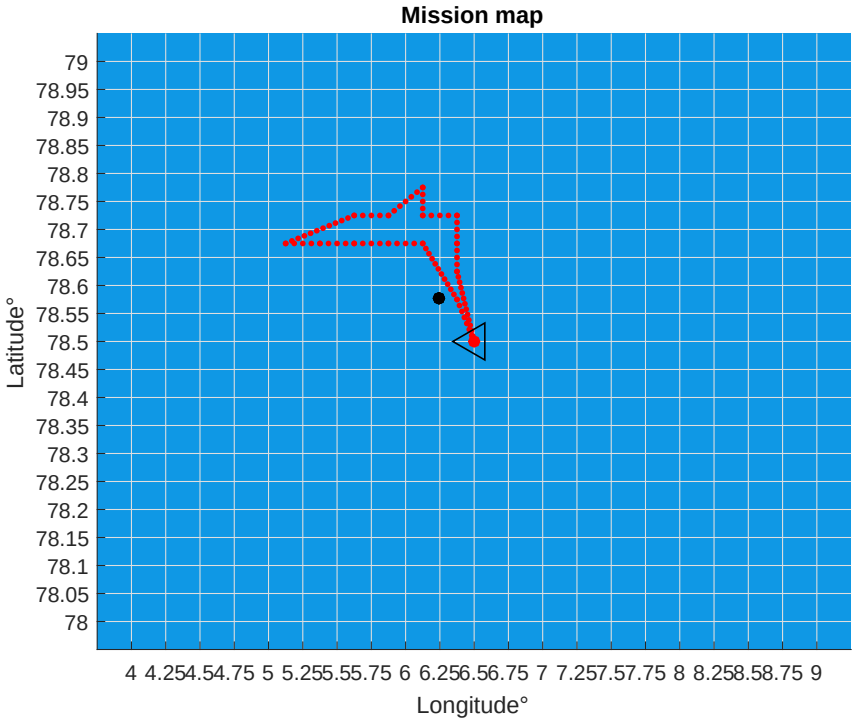

Figure 5.18: UAV mission when 2%-rule weather map from Figure 5.17 is used to compute the path.

The tests and simulations above indicate that the algorithm, when running in search only mode, does the expected job: Prioritizes icebergs that are both close and moving towards the vessel. Grid cells containing icebergs that are predicted to move in the opposite direction are given low priority and are not included in the UAV's path, even if they are close to the vessel. Because of the uncertainty in the 2%-rule, there will always be a risk that these icebergs are actually drifting towards the vessel. Since the chance that these icebergs are detected is small, this can put the vessel in danger. To compensate for this, we will therefore assume that we have a ship radar on the vessel that detects the closest icebergs in a radius of 10 km.

5.3 Combined algorithm

We are now ready to test the complete algorithm. To do this we set $a = b = 0.5$. We want to study how the algorithm combines the two tasks of search and tracking, and how robust it is to changes in different parameters.

This will be done with a series of different simulations. First we will look at the algorithm's performance through a longer simulation of 68 hours. After that, the algorithm's robustness to an increase in ice density will be tested, before we investigate what happens if we start changing the weather more rapidly. At the end we will do some simulations to test how well the tuning mechanism works.

5.3.1 Simulating over a period of 68 hours

In the first simulation we will study how the algorithm behaves over a longer simulation of 68 hours. For this case we will use 30 icebergs. After generating these at random, they were placed in the drift area as shown in Figure 5.19.

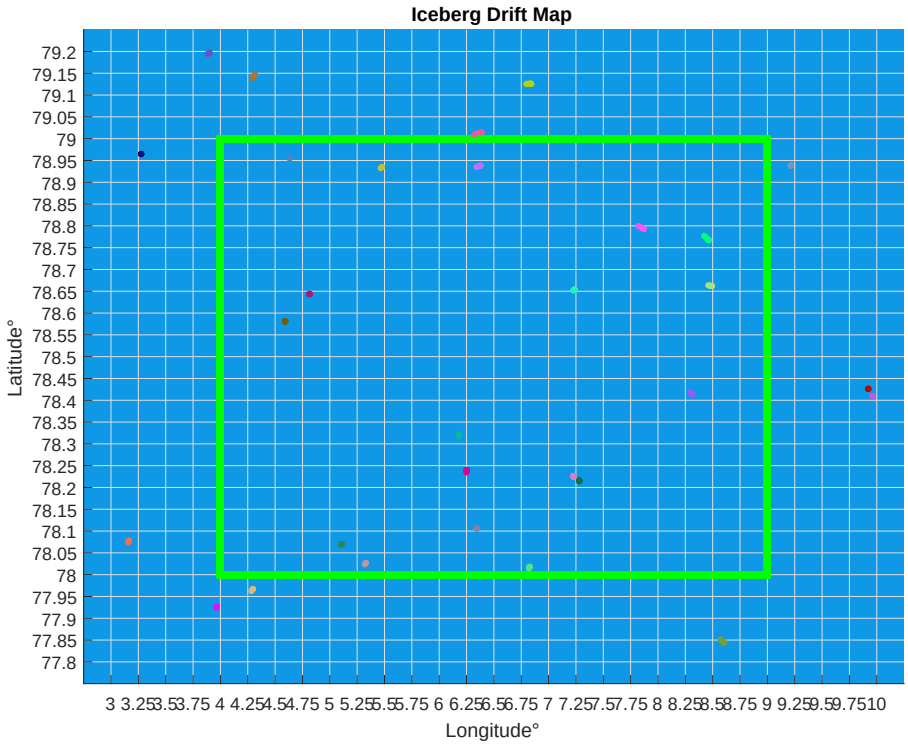

Figure 5.19: Position of the icebergs before the 68 hour simulation.

We see that there are 18 icebergs placed inside the operational area. The simulation was set to start 01/10/2016. We know from Figure 5.15 that the weather data for the first 6 hours will make the UAV search in the area north-east of the vessel. From the Iceberg Drift vs. Iceberg Estimate map plotted after 6 hours, shown in Figure 5.20, we see that one iceberg is approaching the vessel from the north-east. It has not been detected yet, but if it keeps moving towards the vessel, it should be detected soon.

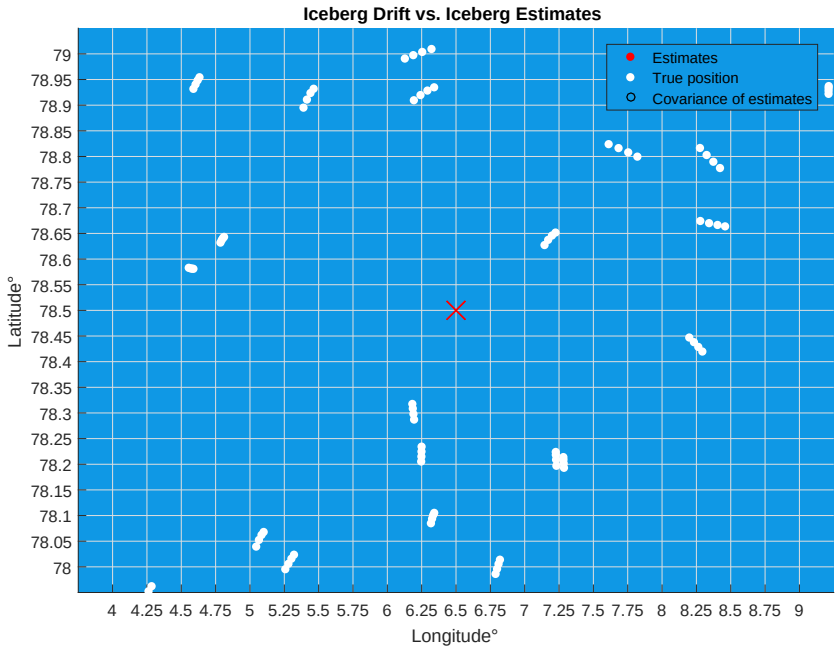

Figure 5.20: Estimates vs. true drift after map simulating for 6 hours. The red cross shows the vessel position. Zero icebergs have been detected yet, hence no estimates.

The Iceberg Drift vs. Iceberg Estimate map after simulating for 26 hours can be seen in Figure 5.21. We see that the iceberg that was approaching the vessel after 6 hours has been detected. In total four icebergs have been detected and are now being tracked. For the iceberg closest to the vessel we see that the Kalman filter have had some problems estimating its trajectory, and that the increased covariance has been important in order to keep track of it. This is probably because it is moving away from the vessel and that rediscovery of this iceberg has not been given as high priority as the other icebergs, which seems to be heading more towards the vessel. Since it has not been rediscovered as often, we see that the increased covariance has made the UAV search in the neighbouring grid cells,

where it has been rediscovered. Figure 5.22 shows the last UAV mission. For this mission, we see that all the tracked icebergs were rediscovered.

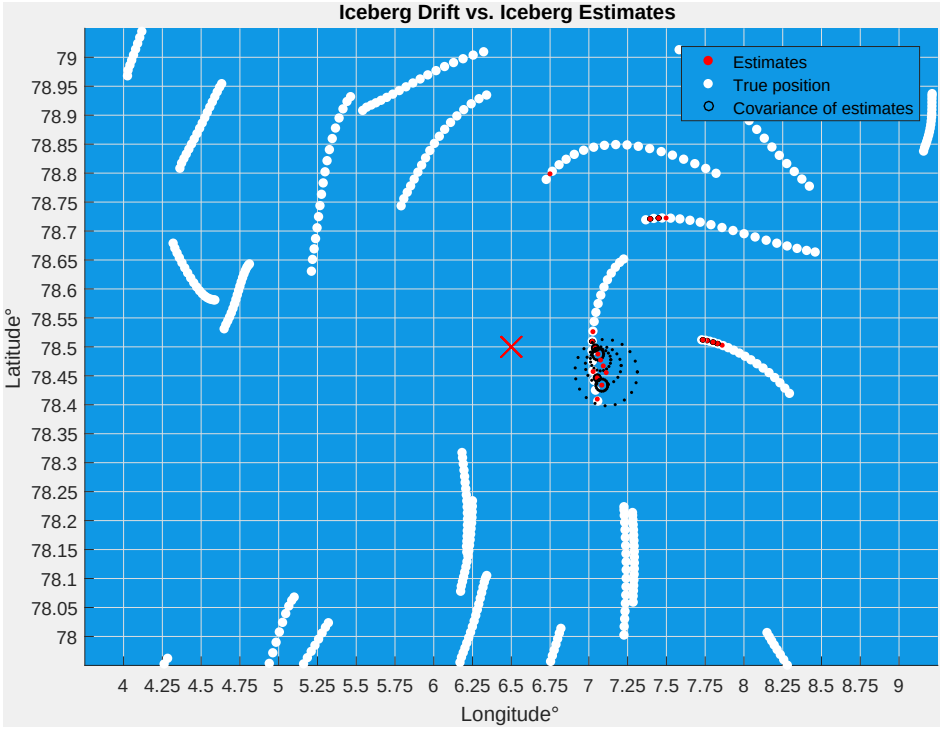

Figure 5.21: Estimates vs. true drift after simulating for 26 hours. The red cross shows the vessel position.

Figure 5.23 shows the Iceberg Drift map after 26 hours. We see that some icebergs have drifted out of the operation area (inside green rectangle) and some have drifted in. We also see that some icebergs have drifted out of the whole drift area, which means that they are removed from the simulation. Some new icebergs have also been added to the edge.

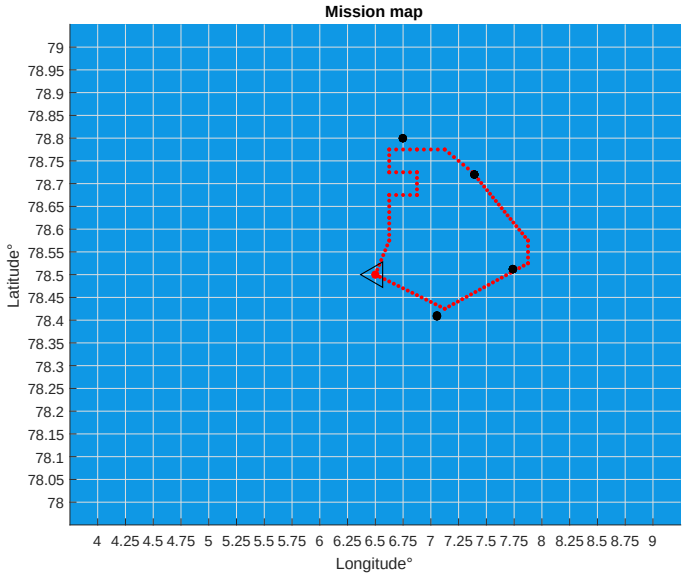

Figure 5.22: UAV mission after simulating for 26 hours.

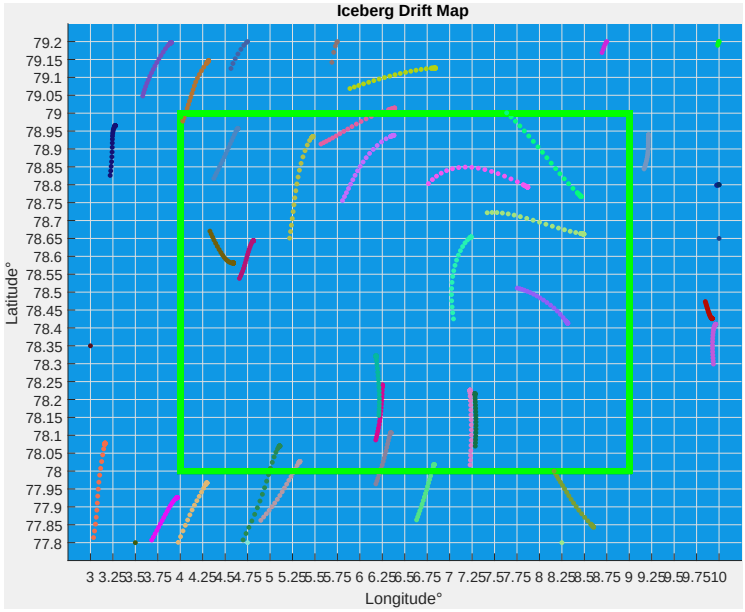

Figure 5.23: Iceberg drift map after simulating for 26 hours. The vessel is located at (78.5,6.5).

The Iceberg Drift vs. Iceberg Estimate map after simulating for 46 hours can be seen in Figure 5.24. We see that the iceberg that was closest to the vessel after 26 hours has now drifted away from it, and is no longer being tracked. We also see that the tracked iceberg furthest to the east of the vessel has increased its covariance, which indicates that it has not been rediscovered for a while. From Figure 5.25, which shows the UAV mission after 46 hours, we see that this iceberg was not rediscovered. If we study the 2%-rule weather map used to evaluate the unknown threats after 46 hours, which can be seen in Figure 5.27, we see that the iceberg furthest to the east is predicted to move away from the vessel, which could explain why it is not given priority in the search and track mission.

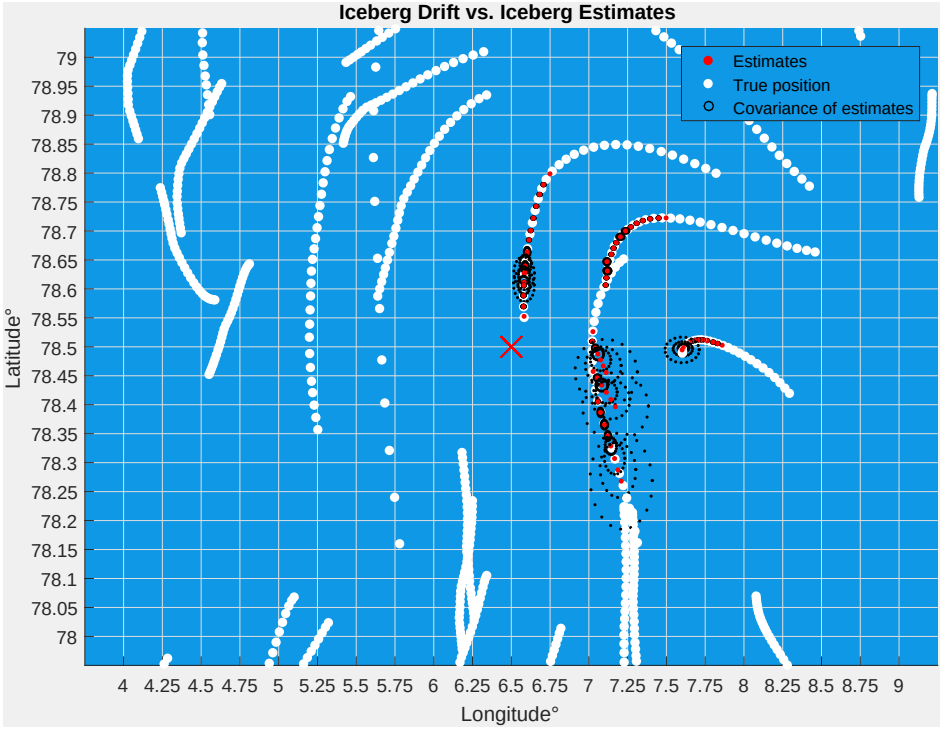

Figure 5.24: Estimates vs. true drift after simulating for 46 hours. The red cross shows the vessel position.

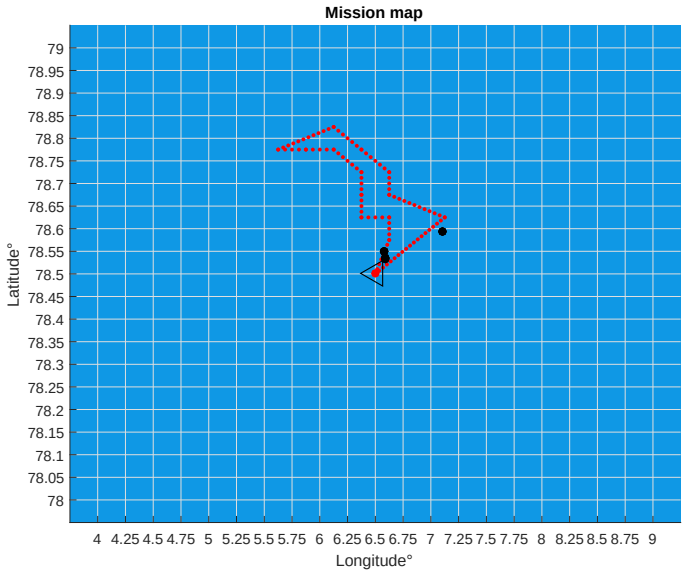

Figure 5.25: UAV Mission after simulating for 46 hours.

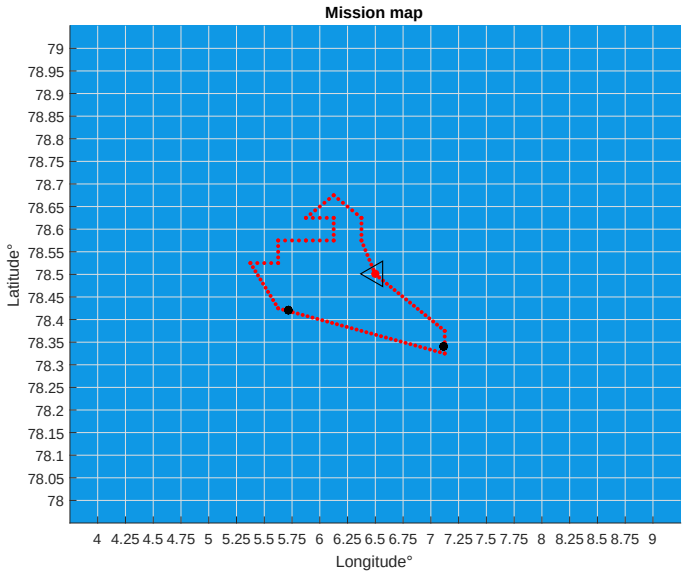

Figure 5.26: UAV Mission after simulating for 68 hours.

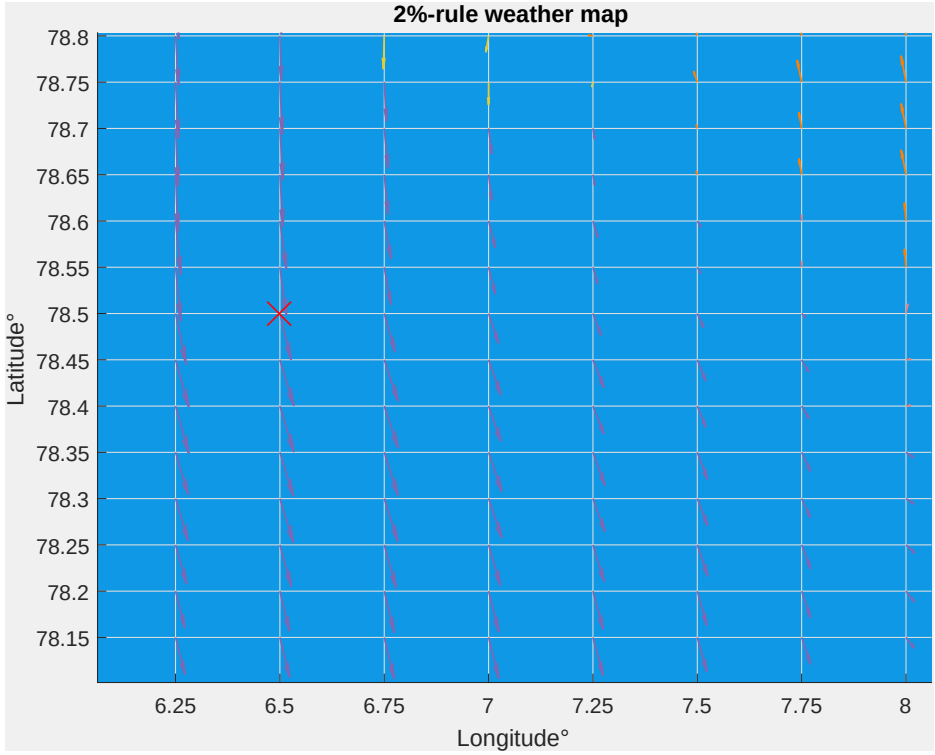

Figure 5.27: Velocity components computed by the 2%-rule for the grid cells closest to the vessel after 46 hours. The red cross shows the vessel position.

The Iceberg Drift vs. Iceberg Estimate map after 68 hours can be seen in Figure 5.28. The map contains a lot of information, but two things can be observed: The iceberg furthest to the east has been removed from the tracking list, and a new iceberg has been detected, west of the vessel. The corresponding UAV mission after 68 hours can be seen in Figure 5.26.

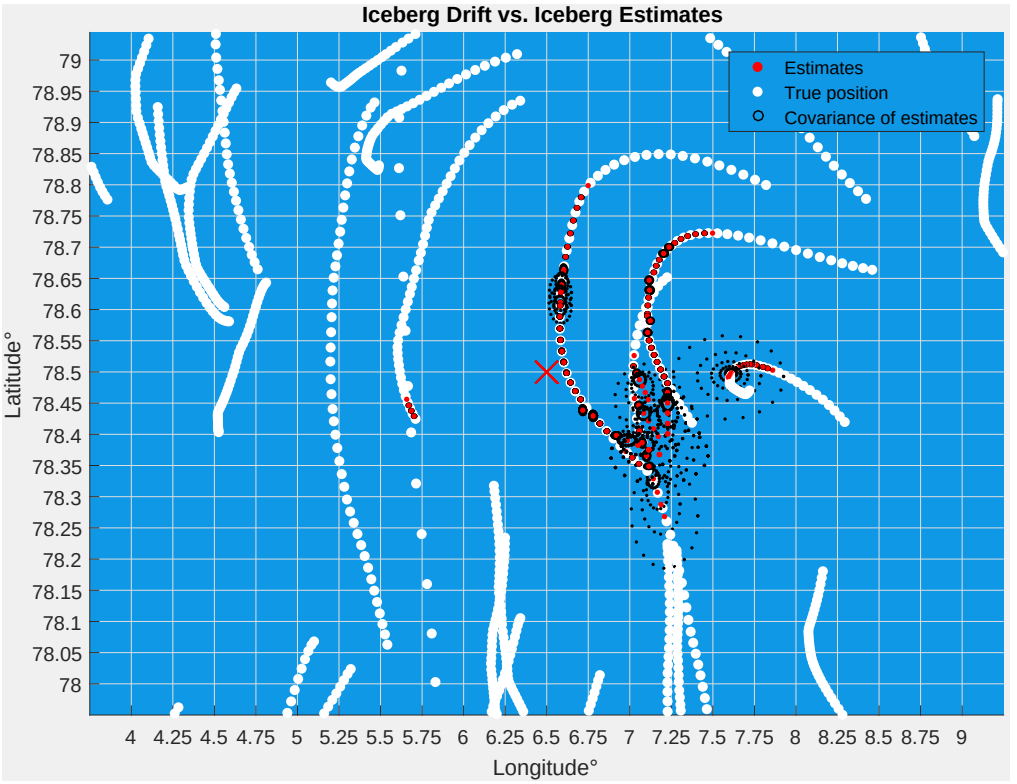

Figure 5.28: Estimates vs. true drift after simulating for 68 hours. The red cross shows the vessel position.

This ends this simulation. The behaviour so far indicates that the algorithm has the desired performance. Through the 68 hours of simulation, none of the most threatening icebergs were left undiscovered. Also, it is clear that both search and tracking are prioritized when deciding the UAV's mission path. Which grid cells to prioritize for searching is decided based on the 2%-rule map. If we study the Iceberg Drift vs. Iceberg Estimate maps in Figure 5.20, 5.21, 5.24 and 5.28, and compare to the corresponding mission maps in Figure 5.22, 5.25 and 5.26, we see that the UAV seems to search in the areas where the most threatening icebergs actually are. This is a good sign, and shows that the 2%-rule is not too far away from the true dynamics. Also, we see that the UAV paths contain both grid cells that are considered to be unknown threats, and icebergs that need to be rediscovered.

5.3.2 Increasing the ice density

We now want to study, is how robust the algorithm is to an increase in ice density. To do this, we will simulate the system with different numbers of icebergs, n_i, placed in the operation area. In the simulation above we simulated with 18 icebergs in the operation area (30 in total in the drift area), for which the algorithm seemed to have the desired behaviour.

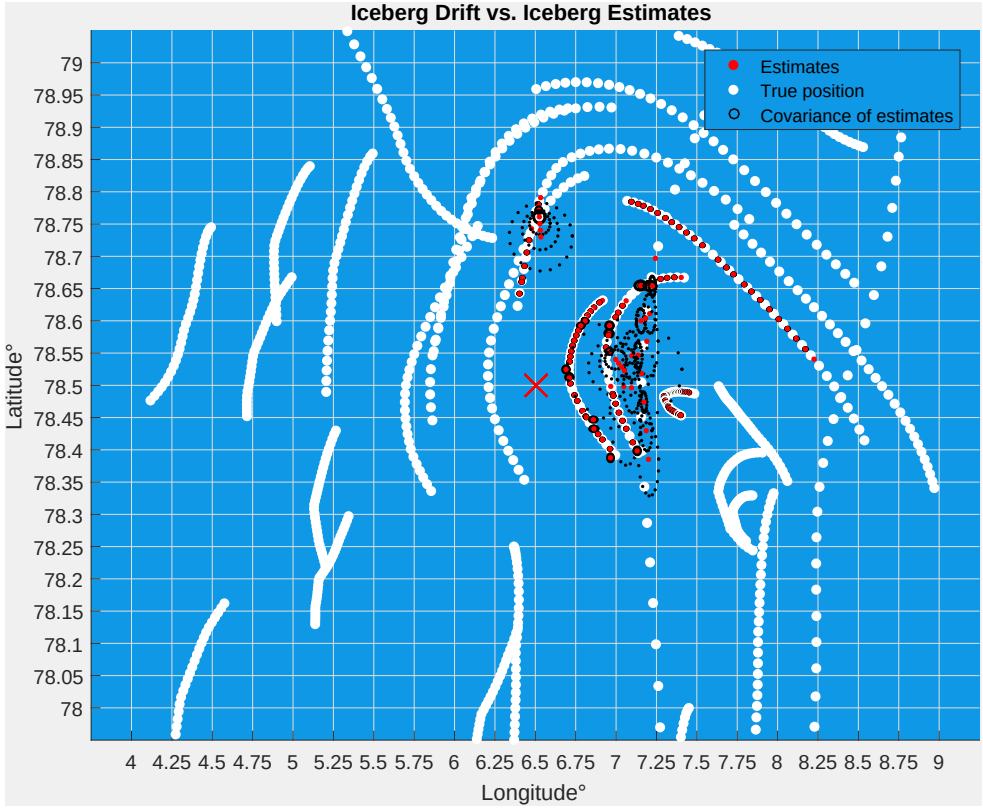

Figure 5.29: Estimates vs. drift map after simulating for 48 hours with 30 icebergs in the operational area. The red cross shows the vessel position.

For the next simulation we put $n_i = 30$, and place them randomly inside the operation area. The simulation was started 01/10/2016, and the simulation lasted for 48 hours. The result of the Iceberg Drift vs. Iceberg Estimate map after 48 hours can be seen in Figure 5.29. We see the same pattern as before: Icebergs located north and east of the vessel are detected and tracked, while the others are left undiscovered. If we study the trajectories of the icebergs around the vessel, they seem to be moving towards south-west. That the algorithm has chosen to prioritize the area north and east of the vessel, therefore seems correct. From the

trajectories of the undiscovered icebergs we see that none of them have had a velocity pointing straight towards the vessel during the simulation, which indicates that it has been right to not prioritize them in the mission.

For the case with 30 icebergs, this simulation indicates that the algorithm still has the desired behaviour. Considering the uncertainty in the 2%-rule, one could argue that the undiscovered iceberg passing by on the western side of the vessel, could be a threat to the vessel and should have been discovered. With the assumption stated above, assuming that we have a radar on the vessel that will detect the closest icebergs in a radius of 10 km, this seems like an acceptable behaviour.

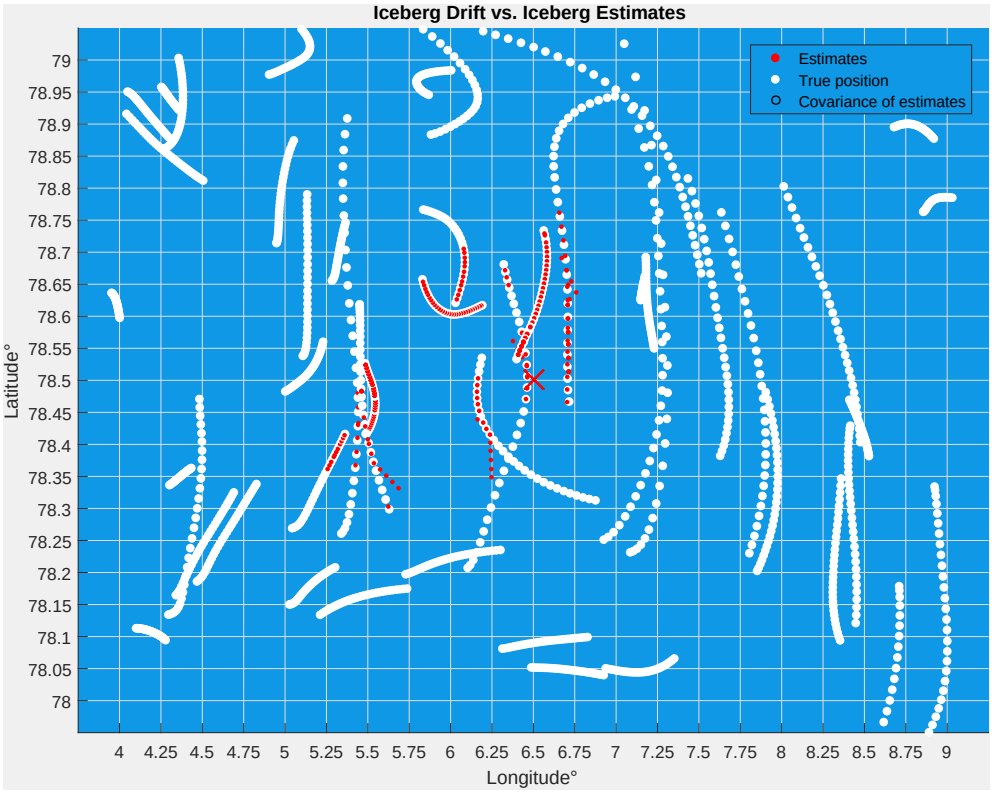

Figure 5.30: Estimates vs. drift map after simulating for 48 hours with 50 icebergs. The red cross shows the vessel position. Covariance plot is turned off.

For the next simulation, we increase the number of icebergs in the operation area to 50. Up until now, every simulation has been started at 01/10/2016. To test the algorithm under a different weather pattern than before, the simulation was started 9 days later, i.e. 10/10/2016. Since 50 icebergs will leave a lot of information on the Iceberg Drift vs. Iceberg Estimate map, the covariance ellipses were removed from the plot for this simulation.

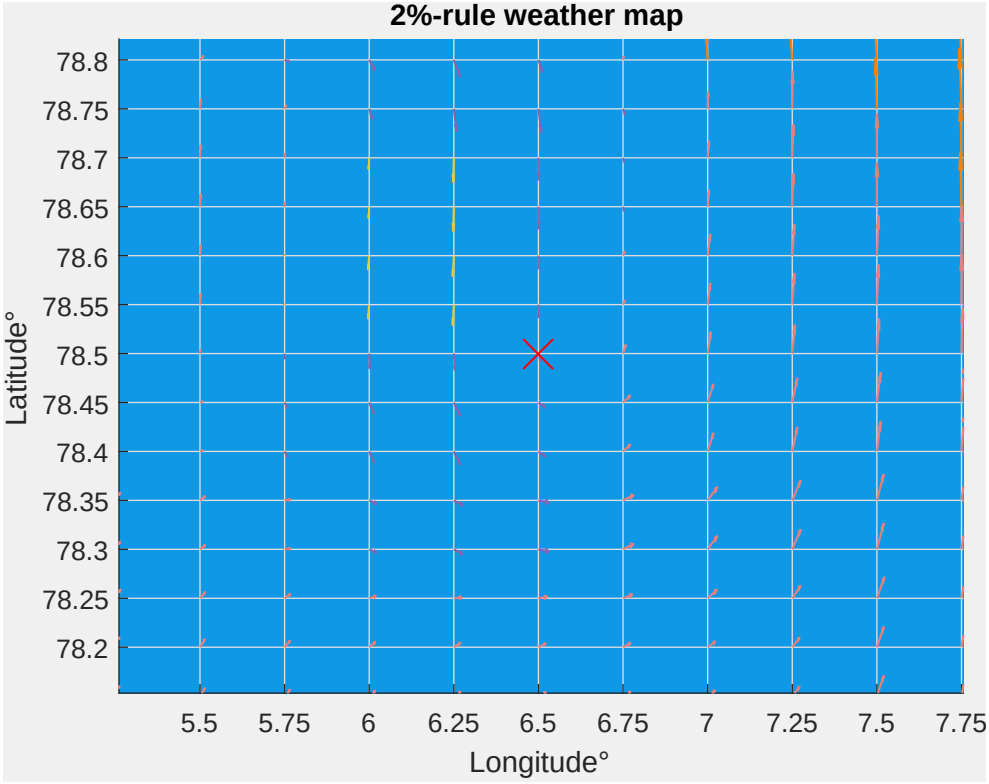

Figure 5.31: 2%-rule map for 10/10/2016. The red cross shows the vessel position.

The Iceberg Drift vs. Iceberg Estimate map after simulating for 48 hours can be seen in Figure 5.30. We see that for this case, icebergs have been discovered in all directions of the vessel. If we compare this to Figure 5.31, which shows the initial 2%-rule weather map for the grid cells around the vessel, we see that the velocity components points in different directions. Therefore, we do not get the same pattern as before, where most of the discovered icebergs were located north-east of the vessel. If one looks closely, the icebergs are actually moving in opposite directions on each side of the vessel: Southward on the western side, northward on the eastern side. This corresponds well with Figure 5.31.

When it comes to performance, it seems like all icebergs that are or have been a threat to the vessel, have been discovered. Some of them are still being tracked, while others have been removed from the tracking list. We see that the search and track radius is limited to ≈ 30 km, due to the maximum nodes the path can include. Since we can only visit the 14 most threatening grid cells, the higher the ice density is, the smaller the search and track radius will become.

The chance that there will be 50 icebergs in the operational area simultaneously is very small, but the simulation result shows that even under these conditions, the most threatening icebergs are discovered and tracked. This indicates that the algorithm has high robustness when it comes to ice density. One of the reasons to this robustness, is the algorithm's property that it prioritizes the closest threats. In this way, an increase in iceberg density will not have such a big impact, since only the closest icebergs will get priority.

5.3.3 Rapidly changing weather

In the next simulations, we will look at the algorithm's performance under rapidly changing weather. Every second hour, the weather data will be updated with data from a random day in the period 01/10/2016-01/11/2016. That the weather will change this often and this randomly, is not very realistic, but a good way to test the algorithm's robustness. For these simulations, 30 icebergs will be placed randomly inside the operational area.

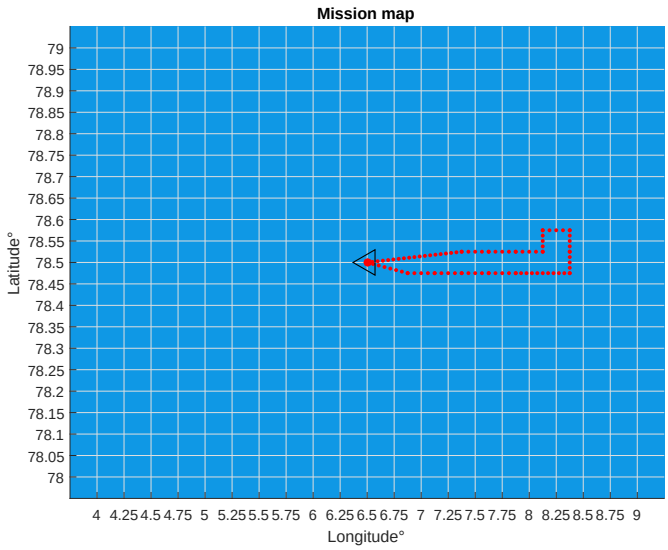

Figure 5.32: First UAV mission when simulating with 30 icebergs and rapidly changing weather.

The first two UAV missions in this simulation can be seen in Figure 5.32 and 5.33. We see that the two paths are very different, indicating that there has been a huge change in weather between the two missions.

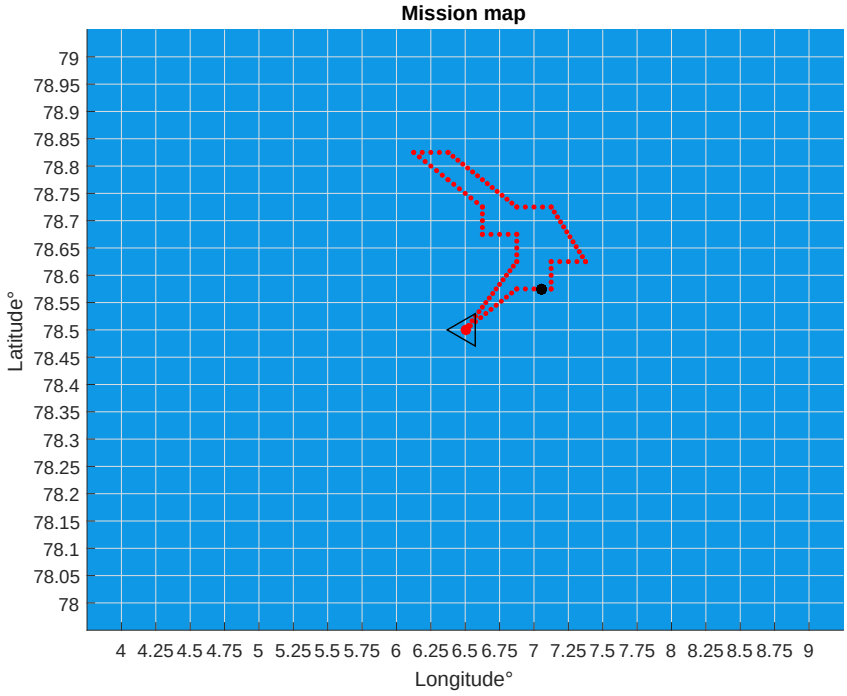

Figure 5.33: Second UAV mission when simulating with 30 icebergs and rapidly changing weather.

The Iceberg Drift vs. Iceberg Estimate map after simulating for 48 hours can be seen in Figure 5.34. We see that the iceberg trajectories seem to be shorter, compared to the trajectories in Figure 5.29 and 5.30, where the weather is changing as normal. In this case, since the weather is changing often, the direction of the icebergs' acceleration will vary with it. Therefore, the chances of gaining a high speed in one direction decreases, which gives the shorter trajectories.

Since the icebergs are moving slower, it should be easier to keep track of them. From Figure 5.34 we see that this is not the case, and that most of the tracked icebergs have a high covariance, indicating that they have not been rediscovered in a while. Since the areas that are threatening to the vessel in operation varies with the weather, the discovered icebergs span a much larger area than before. This makes it harder for the UAV to rediscover the tracked icebergs as often as before. The direct consequence of this is an increased covariance, a higher chance

of a bad estimate, which again increases the chances of the algorithm losing track of the iceberg.

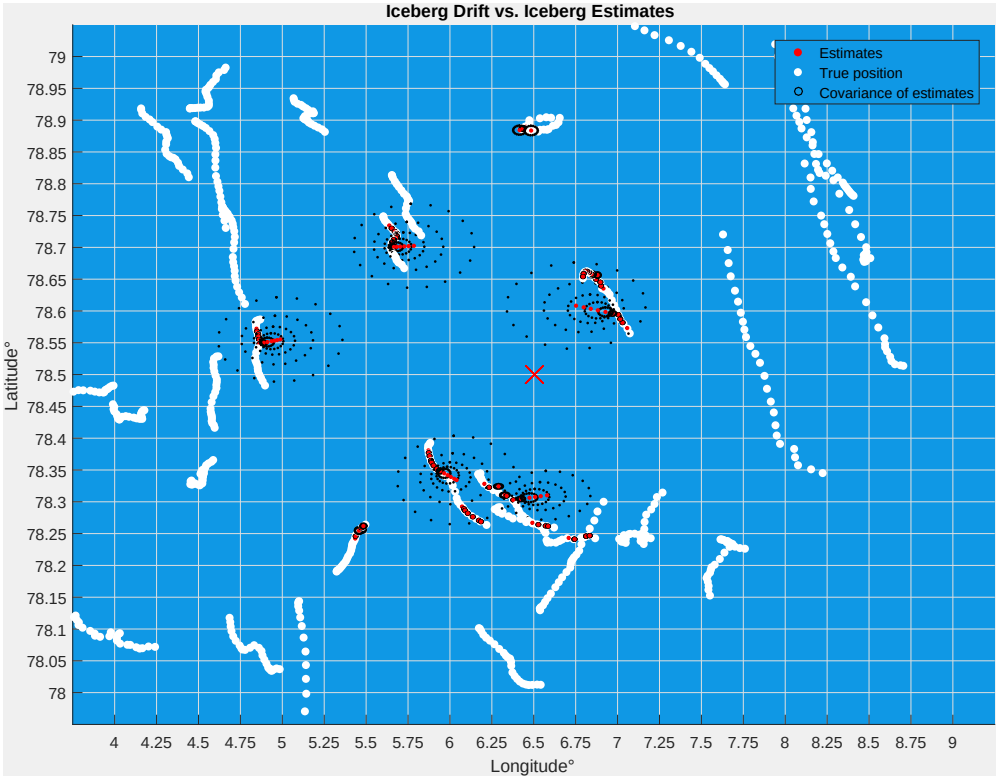

Figure 5.34: Drift vs. Estimate map with rapidly changing weather after simulating for 48 hours. The red cross shows the vessel position.

Through this simulation, we have seen that the performance of the tracking part of the algorithm is decreased when the weather varies more often. The chances of detecting more icebergs seems to be increased, but since the weather is varying more rapidly and more random, it is difficult to classify the icebergs as threatening or not. The biggest problem for the algorithm with the rapidly changing weather, is that the inertia in the actual iceberg drift is much larger than the inertia in the estimated velocity components, computed by the 2%-rule. If the weather changes, the velocity components for each grid cell will change immediately, while the icebergs will need a some time before their velocity has adapted to this. In this way, the model used to estimate the drift gets less accurate, which decreases the algorithm's performance. Luckily, it is very unlikely that the weather will vary this much and this often, but this is something one should be aware of when applying the search and track application in areas with severe weather.

5.3.4 Tuning mechanism

The last thing we will study when it comes to the performance of the search and track algorithm, is the tuning mechanism, and how it affects the algorithm's behaviour. We will test this with two simulations. One where $a = 0.7$ and $b = 0.3$, and one where $a = 0.3$ and $b = 0.7$, i.e., one simulation where tracking should be prioritized more than search, and one where search should be prioritized over tracking. To make it easier to compare the results, the two simulations will use the same set of icebergs. To make sure that many icebergs are located close to the vessel, the icebergs were generated in a small area of ± 0.2 latitude and ± 1.25 longitude, relative to the vessels position. 25 icebergs were used. In addition, both simulations were started at 11/10/2016, giving them the same weather data. The simulation time was 24 hours.

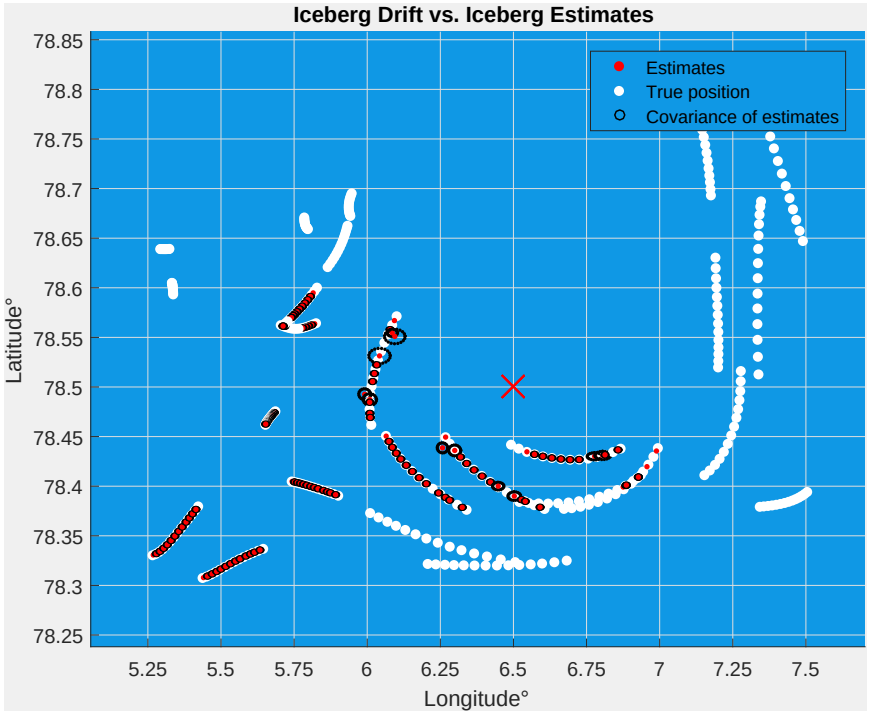

Figure 5.35: Drift vs. Estimate map when $a = 0.7$ and $b = 0.3$, i.e., tracking is more prioritized. The red cross shows the vessel position.

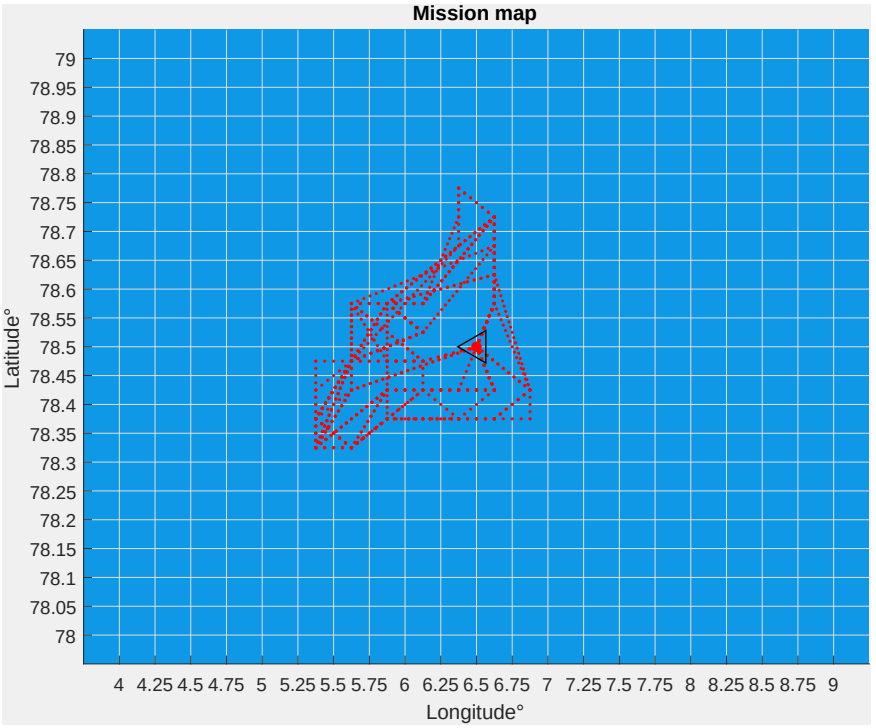

Figure 5.36: The grid cells covered by the UAV through the simulation, when $a = 0.7$ and $b = 0.3$.

The results of the two simulations can be seen in Figure 5.35-5.38. If we first compare Figure 5.35 and Figure 5.37, which show the Iceberg Drift vs. Iceberg Estimate maps for the two simulations, we see that the tracking abilities have been dramatically reduced when changing a from $a = 0.7$ to $a = 0.3$. For the case where tracking is prioritized, all icebergs that are detected, are tracked with high precision through the rest of the simulation. For the case where search is prioritized, we see from the covariance ellipses that the tracking has been much less successful. Actually, the algorithm has lost completely track of three of the icebergs detected.

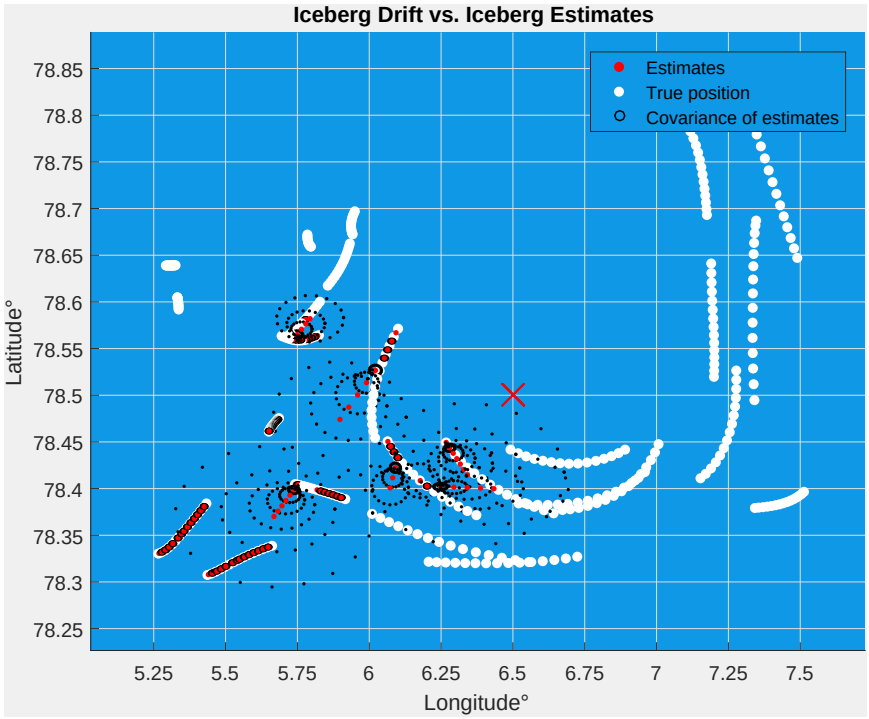

Figure 5.37: Drift vs. Estimate map when $a = 0.3$ and $b = 0.7$, i.e. search is more prioritized. The red cross shows the vessel position.

If we compare Figure 5.36 and Figure 5.38, which show the total area covered by the UAV through the simulations, we see that the area covered is larger when tracking is prioritized, compared to when search is prioritized. One would expect the opposite, but this has a simple explanation: Through the 24 hours of simulation, the weather and hence the velocity components for each grid cell, does not change too much. Because of this, since the search model prioritizes the most threatening areas when direction of movements is considered, the UAV ends up searching the same areas. Since going back to these areas is prioritized over rediscovering tracked icebergs, the algorithm loses track of the icebergs when they leave this area. For the case where tracking is prioritized, much more weight is given to actually following these icebergs, even if they leave the area that is considered the most threatening. Since the icebergs are moving, the area covered by the UAV will get bigger, compared to the case where tracking is not prioritized.

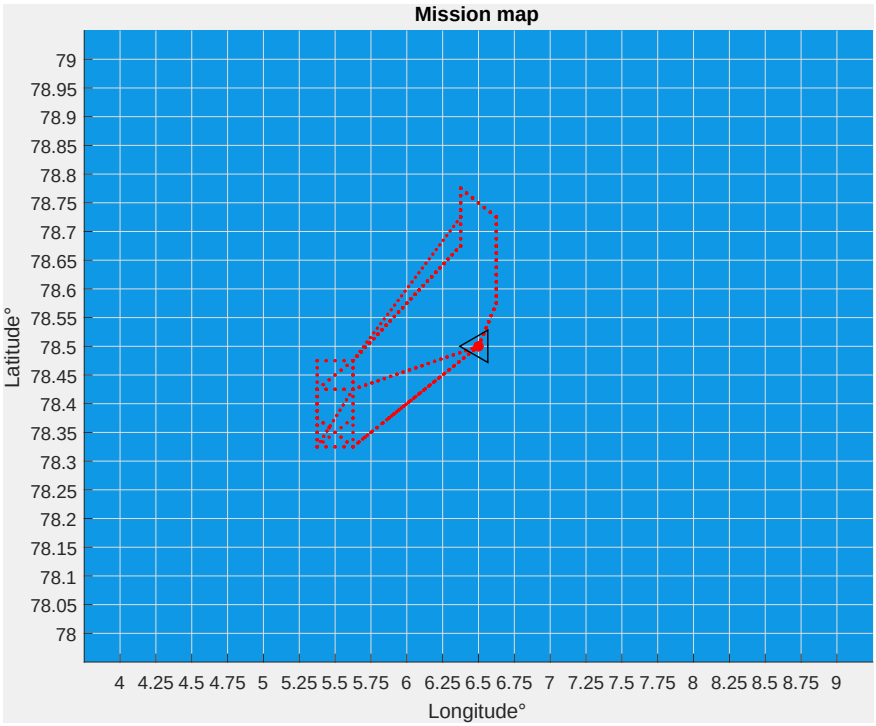

Figure 5.38: The grid cells covered by the UAV through the simulation, when $a = 0.3$ and $b = 0.7$

The results from these simulations, indicate that the tuning mechanism works as it should. By increasing a and lowering b, the algorithm puts a bigger effort into rediscovering the tracked icebergs, compared to the case where $b > a$. By increasing b and lowering a, the most threatening areas are prioritized. For slow changing weather, the results indicate that $a > b$ gives the biggest search area, but since the search model is supposed to focus on the most threatening areas, this is the desired behaviour.

5.4 Part 3: Satellite and radar data

In this part, we want to investigate how the integration of radar and satellite data will affect the search and track algorithm's behaviour. First, we propose a way radar data could be added to the Kalman filter in order to make the system more safe. How this works will be tested through some simple simulations. After this, we study how satellite data can be used to improve the algorithm's performance. This will be tested assuming two different iceberg classification accuracies.

5.4.1 Radar

Through the tests and simulations in Section 5.2-5.3, we have assumed that we have a radar on the vessel that detects the closest icebergs in a radius of 10 km from the vessel. This can easily be integrated into the algorithm, by adding the icebergs that are detected by the radar to the Kalman filter. If we assume that the reflected radar-signal can be used to calculate the icebergs' position, the position can be added as the estimated location. Since we do not know the icebergs' velocity, this can be set to 0. Two simulations where performed to see how this would impact the algorithm's behaviour.

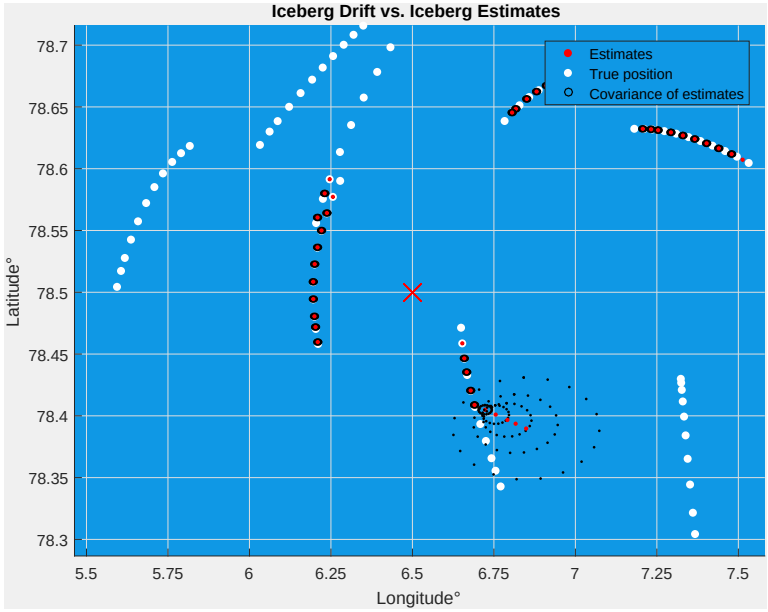

Figure 5.39: Drift vs. Estimate map after 16 hours with radar. The red cross shows the vessel position.

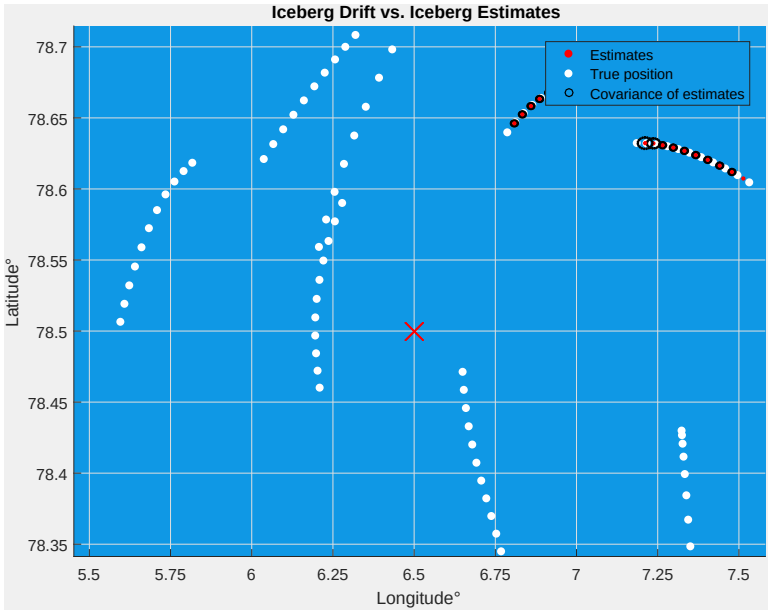

Figure 5.40: Drift vs. Estimate map after 16 hours without radar. The red cross shows the vessel position.

Figure 5.39 and 5.40 show the Iceberg Drift vs. Iceberg Estimate map for the two simulations. In the first simulation, radar data was used. In the second simulation, radar data was turned off. Both simulations were done with the same icebergs and weather data. Simulation time is 16 hours. If we compare the figures, we see that two more icebergs are detected when we are using radar (Figure 5.39), compared to the case when we are not using radar (5.40).

If we study the iceberg on the western side of the vessel, we see that it passes the vessel undetected, when simulating without radar. Because of the uncertainty in the 2%-rule, this is not a desirable behaviour, especially when it is this close to the vessel. By using the radar, the iceberg is detected when it is 10 km away from the vessel. After this, the iceberg is tracked by the UAV. If we look at the other iceberg that is detected by the radar, it does not seem to be a very big threat to the vessel, considering the trajectory it makes. On the other hand, icebergs that are this close should be tracked, at least for a while, just in case they turn and start moving towards the vessel. From Figure 5.39 we see that the UAV gives less priority to rediscover the iceberg south-east of the vessel when it has drifted further away. Eventually, it is removed from the Kalman filter.

From these simple simulations we have seen how radar data could be included into the algorithm, to make sure no icebergs get closer than 10 km from the vessel. This makes a more safe system, keeping track of all the nearest icebergs.

The biggest challenge here is to decide when these icebergs should be removed from the Kalman filter. For the search and track algorithm, which have a maximum number of nodes that can be added to the path, it is important that as many nodes as possible are available for a mission. Tracking some of the icebergs detected by the radar for too long, could potentially affect the search and track algorithm's performance. A solution could therefore be to remove these icebergs from the tracking list, if they leave the 10 km zone. This would require changes to the search and track algorithm, which is beyond the scope of this thesis. It will therefore not be tested here. Considering that most icebergs that get as close as 10 km from the vessel will already have been discovered by the search and track algorithm, the number of icebergs tracked by the Kalman filter, detected by the radar, will be low. For most cases the solution proposed and tested here will be sufficient.

5.4.2 Satellite data

The combined search and track algorithm's performance and robustness has through the simulations in Section 5.2-5.3 been studied and analysed. The last thing we want to look at, is the possibilities of integrating information from satellite data into the algorithm.

As explained in Chapter 3, Section 3.4, a satellite image can be modelled as a matrix $\mathbf{I}$, containing information about possible iceberg locations, in addition to some added noise. Depending on the supplier, the satellite images can only be obtained at certain time intervals. Through the following simulations, this time interval has been sat to 36 hours. A satellite image will therefore be provided to the algorithm at the start of every simulation, and every 36 hours after this. To make it easier to see which icebergs are discovered from the satellite image, the radar has been turned off.

We are to do two simulations. The first one assumes that the computer vision used to analyse the satellite images, classifies icebergs with 100% accuracy. This means that every iceberg location revealed through matrix $\mathbf{I}$, gives the exact position of one of the icebergs in the iceberg drift model. The second simulation will look at the case where the computer vision classifies icebergs with only 75% accuracy. To make it easier to compare the results, the same icebergs and weather data will be used for both simulations. The simulation is set to start 20/10/2016. 25 icebergs will be used.

We start by looking at the case where the satellite images are 100% correct about the icebergs' location. The initial Iceberg Drift vs. Iceberg Estimate map, prior to the first mission, can be seen in Figure 5.41. Since the satellite image provides the Kalman filter with 100% accurate estimates about the icebergs' location before the simulation starts, all estimates in Figure 5.41 are located at the same position as the true icebergs.

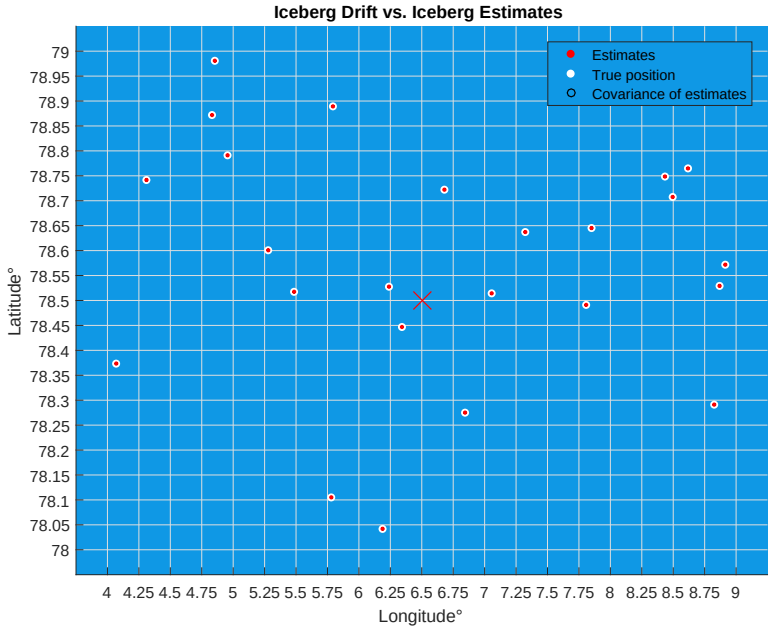

Figure 5.41: Initial Drift vs. Estimate map when satellite data is 100% correct. The red cross shows the vessel position.

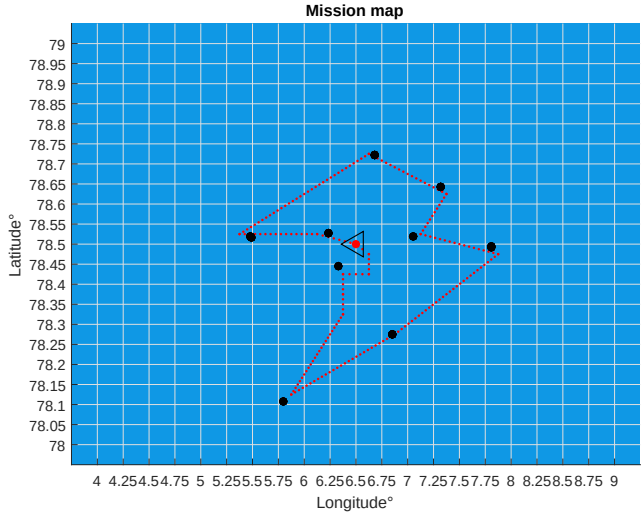

Figure 5.42: First UAV mission when satellite data is 100% correct.

The first UAV mission can be seen in Figure 5.42. We see that 12 out of 14 grid cells included in the path, contains an iceberg. Since the estimates are assumed to be 100% correct, the grid cells close to the vessel containing these icebergs, are assigned a high probability for being a threat. Because of this, they will get a high priority when determining which grid cells to visit. This is a nice property, since the icebergs observed on the satellite image needs to be detected fast, before they leave the location they were observed at. By detecting the closest ones on the first mission, one can measure the velocities and use this to estimate their trajectories.

When using satellite data, the first missions after an updated image is obtained, will prioritize to detect the icebergs that are observed to be close to the vessel. If this number is close to δ (in this case 14), unknown threats will not be given much priority on these missions. This could be a drawback, especially if the accuracy of the satellite images are low. For this case, where the accuracy is 100%, detecting these icebergs should be given the highest priority, whatsoever.

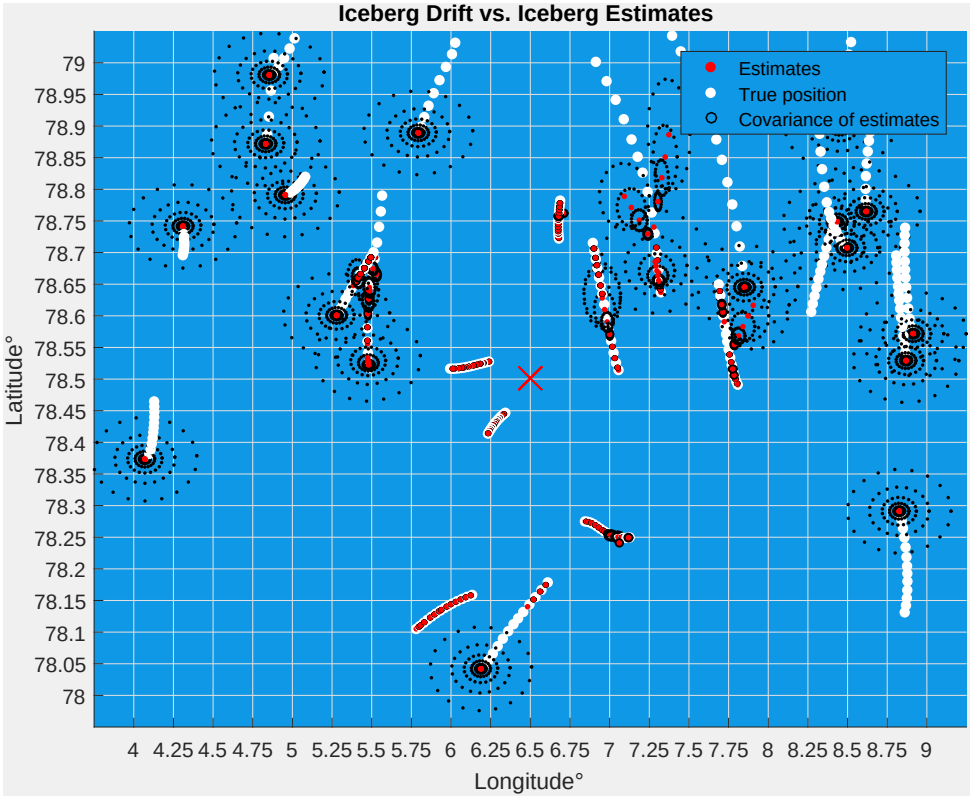

Figure 5.43: Drift vs. Estimate map after simulating for 35 hours when satellite data is 100% correct. The red cross shows the vessel position.

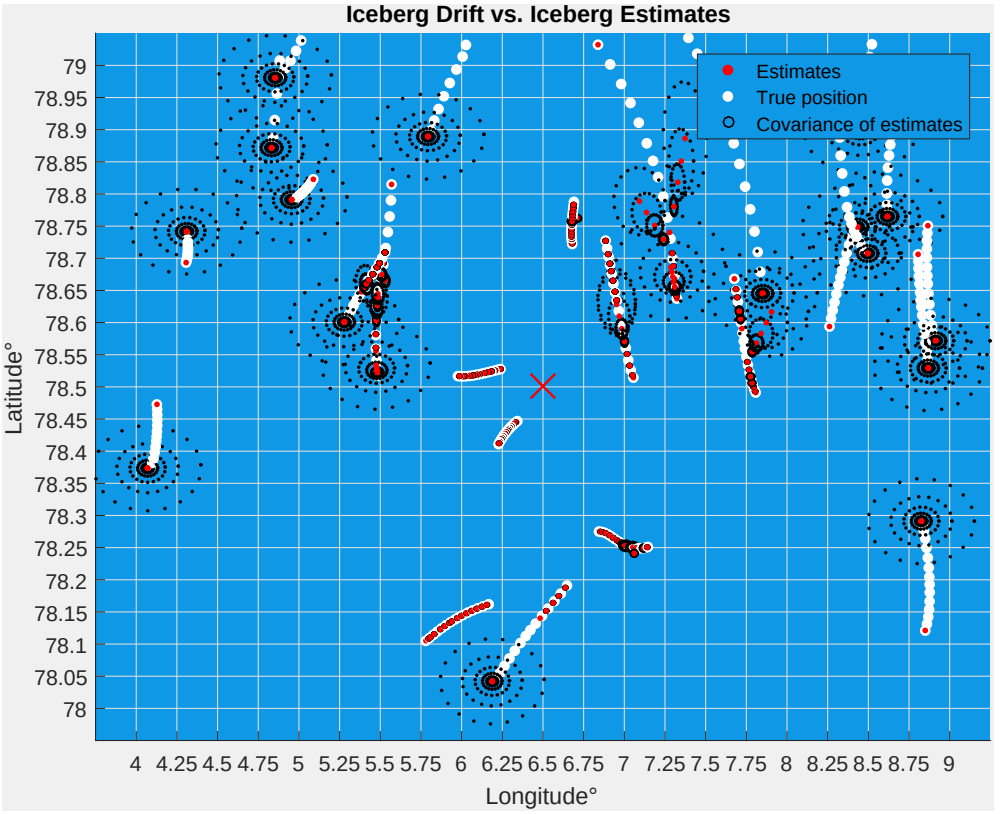

Figure 5.44: Drift vs. Estimate map after simulating for 37 hours when satellite data is 100% correct. The red cross shows the vessel position.

Figure 5.43-5.44 shows the Iceberg Drift vs. Estimate map one hour before (35 hours) and one hour after (37 hours) the Kalman estimates have been updated with satellite data. From Figure 5.43, which shows the estimates before the update, we see that all estimates of icebergs that are far away from the vessel, are still at the same location as when the simulation was started. The fact that the ellipse around these icebergs is big, also indicates that these icebergs have been removed from the Kalman filter's tracking list. This is as expected, since they are far away and in that sense not very threatening to the vessel.

The icebergs closer to the vessel have been tracked more accurate through the simulation. This is also the case for the iceberg starting at (78.11,5.78). Even though this iceberg is not very close, it is moving towards the vessel, which has given it priority in the UAV path.

From Figure 5.44 we see that the updated satellite image have made the Kalman

filter predict the icebergs' locations at the exact position. From here on, a similar pattern as explained above will be repeated: Those icebergs that are close to the vessel will be detected fast, while those that are far away will not be given priority and eventually be removed from the Kalman filter.

For the case where the accuracy of the UAV's iceberg classification is 75%, 25% of the information provided from the satellite image is incorrect. This means that, in average, 25% of the observed icebergs are not actually icebergs. The initial Iceberg Drift vs. Iceberg Estimate map of this scenario can be seen in Figure 5.45. We see that some of the estimates plotted on the map, does not correspond to an iceberg's position. Also, we see that some of the icebergs have not been classified as icebergs on the satellite image (does not have a red dot on it).

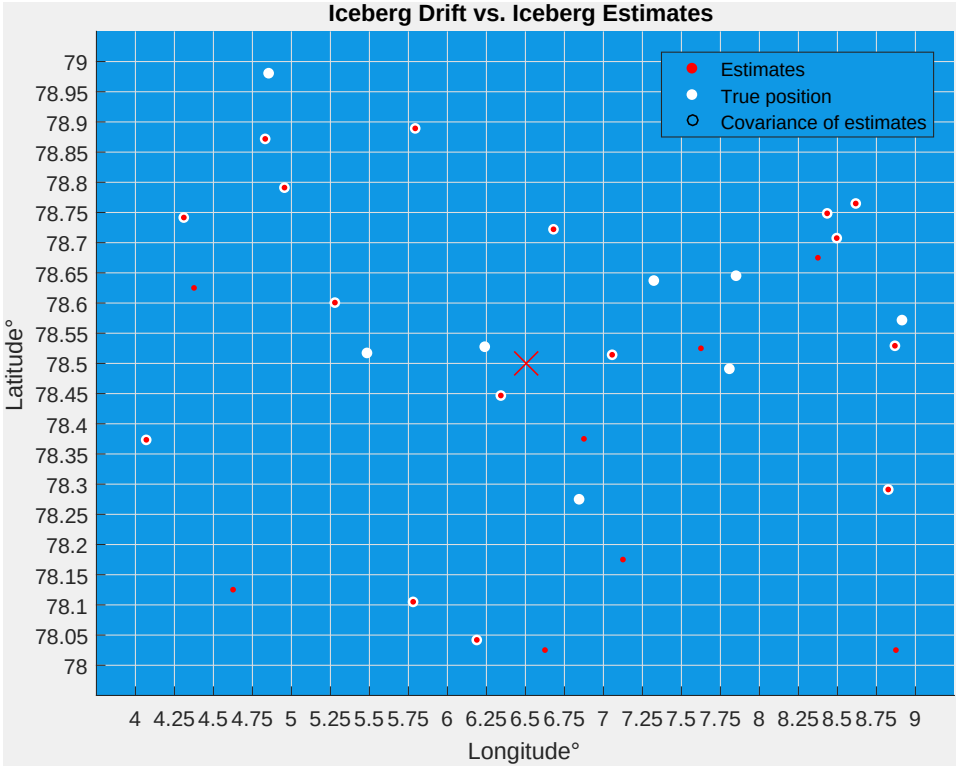

Figure 5.45: Initial Drift vs. Estimate map when satellite data is 75% correct. The red cross shows the vessel position.

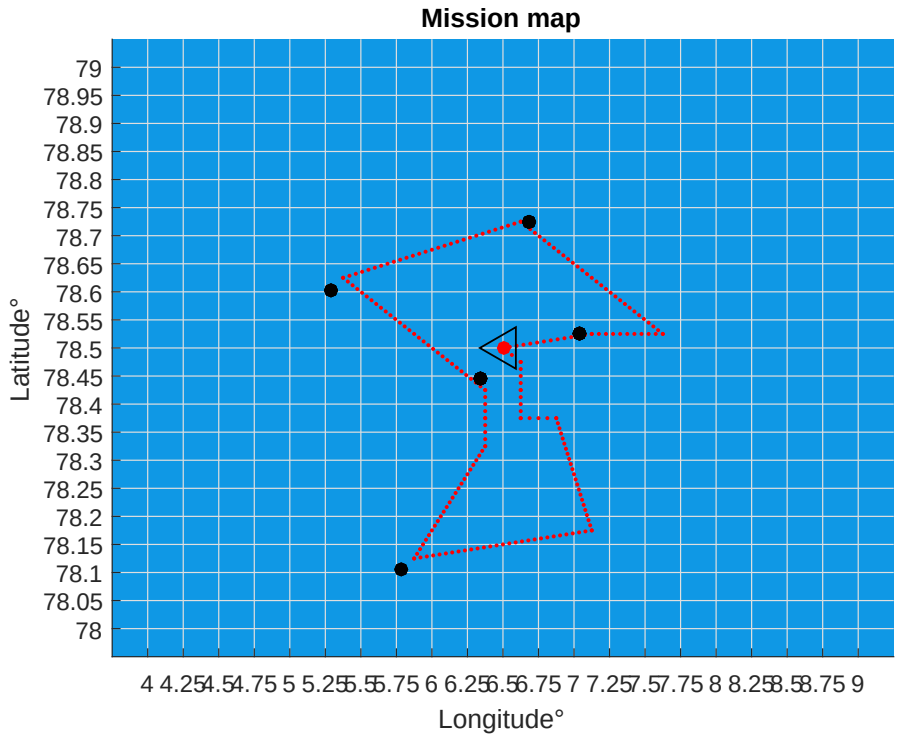

Figure 5.46: Initial UAV mission map when satellite data is 75% correct

The initial mission can be seen in Figure 5.46. If we compare the grid cells included in the path, to where the estimates are located in Figure 5.45, we see that three of the satellite estimates that turned out to be wrong, are included in the path. Considering that only 14 grid cells can be included in the path, this could reduce the algorithm's performance, if these estimates keeps holding up 3 nodes for a longer time.

Figure 5.47 shows the Iceberg Drift vs. Estimate map after 15 hours of simulation. Figure 5.48 shows the mission corresponding to this. We see that even now after 15 hours, one of the nodes included in the path origins from one of the satellite estimates that turned out to be wrong. Since the maximum number of nodes that can be included in the path is 14, the uncertainty in the satellite image can reduce the algorithm's performance, since the wrong estimates occupies one of the nodes in the path.

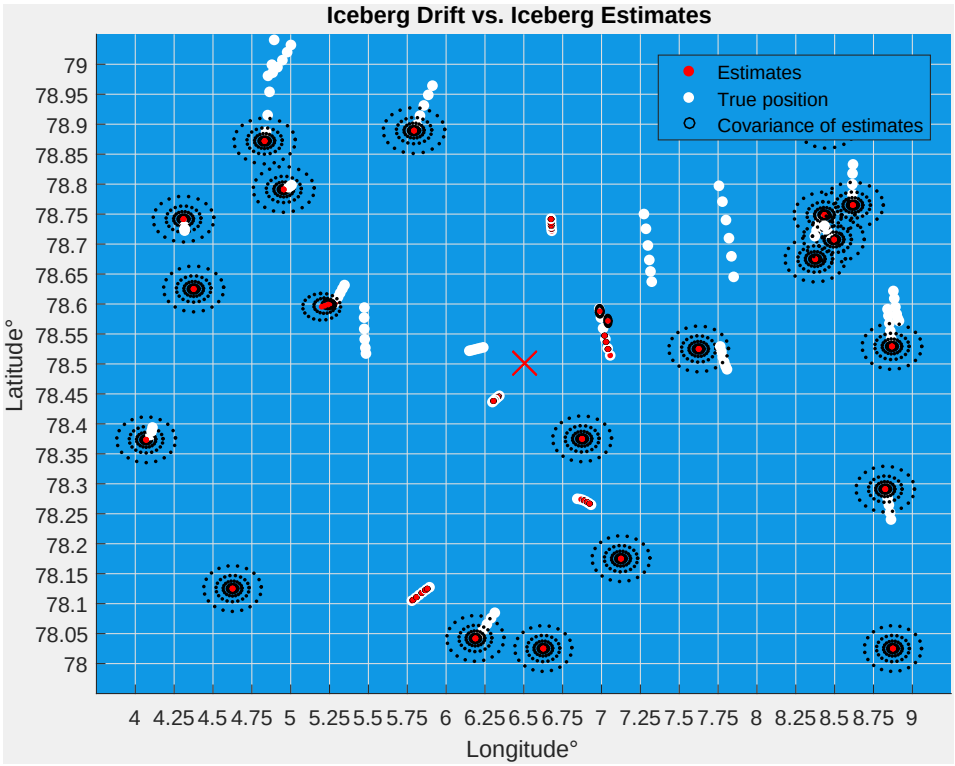

Figure 5.47: Drift vs. Estimate map after simulating for 15 hours when satellite data is 75% correct. The red cross shows the vessel position.

On the other hand, if the number of icebergs in the area is low, the proposed way of integrating the satellite data into the algorithm will be sufficient. Since no icebergs will be discovered at the locations the satellite image estimates, this estimate will be removed from the tracking list relatively fast. If we can accept a few missions where some nodes are held up by these wrongly classified icebergs, the solution will hold.

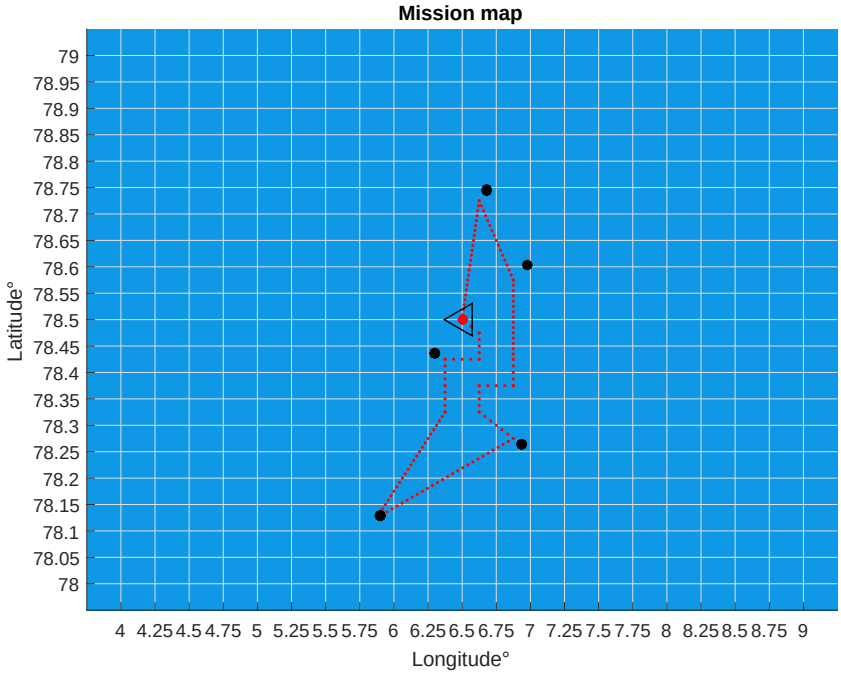

Figure 5.48: UAV mission map after simulating for 15 hours when satellite data is 75% correct.

By keeping the accuracy of the iceberg classification to a minimum of 75%, it is clear that integrating satellite data into the algorithm will improve its performance, simply because we get more information about the icebergs than we had. How to integrate this data into the algorithm still remains a question. A solution has been proposed, but for this to work, the number of icebergs in the area has to be lower than 14. If this is not the case, one should consider to integrate the satellite data in another way than what is proposed and tested here. One solution could be to dedicate a few missions after the updated satellite data is received, to search the areas the satellite image indicates icebergs are located. After these missions, those that are detected can be added to the Kalman filter, before the search and track algorithm continues as normal.

For the case where the classification accuracy gets lower than 75%, the chances that we will get a lot of wrong information gets so large, that it can cause more confusion than improvement. If we assume that the icebergs are large enough to be detected on a satellite image, an accuracy above 75% should be achievable.

Chapter 6

Concluding Remarks and Further Work

In this thesis, an algorithm that combines iceberg search and tracking has been developed, and its performance and viability has been studied. This has been done with the goal to protect a vessel or oil rig from icebergs. The algorithm developed can easily be applied in any Arctic area, but in this thesis, an area west of Svalbard has been considered.

To test the algorithm's performance and robustness, a simulator has been developed. The simulator consist of two main parts: One part that simulates the iceberg drift, and one part simulating the UAV missions, in addition to tracking and detection of icebergs. The algorithm is unaware of the true states of the icebergs, and uses weather data from wind and ocean current, together with the 2%-rule and UAV measurements, to predict and track the iceberg dynamics.

Some simulations were done to test the accuracy of the 2%-rule. The results indicated that the deviation between the estimates and the true iceberg trajectories, increases with time. We therefore concluded that an iceberg that is being tracked, should be rediscovered as often as possible, in order for the algorithm not to lose track of it. More simulations were carried out, in order to test which factors that were important for the iceberg drift. We found that in addition to wind and ocean current, the keel shape and iceberg mass seemed to have a big impact. One of the simulations also indicated that the three different keel shapes considered, divided the iceberg trajectories into three distinct groups.

A great deal of effort has been put into testing the algorithm's performance and robustness. First, a series of simulations were done to test how each of the search and track models performed separately. Both models seemed to have the desired behaviour, but some restrictions were found. When running in track only mode,

icebergs that were close to the vessel were given high priority in the path, while those far away were given less. We found that the number of icebergs we were able to track simultaneously, had an upper limit equal to the number of nodes included in the path. If the number of tracked icebergs exceeded this number, the chances of losing track of an iceberg increased. For the search only case, the simulations indicated that the UAV prioritized to search the most threatening areas. As a consequence of this, icebergs close to the vessel, located in areas estimated to move away from the vessel, were not discovered. Because of the uncertainty in the 2%-rule, this could put the vessel in danger. It was concluded that a ship radar that can detect the closest icebergs in a radius of 10 km, can be used to eliminate this threat.

To test the combined algorithm's performance and robustness, four different scenarios were considered. The first scenario focused on simulating the combined algorithm with $a = b = 0.5$, over a longer period of time (68 hours). The results indicated that the algorithm had the desired behaviour: Both known and unknown threats were prioritized in the UAV's path, the most threatening icebergs were discovered, and the most threatening icebergs were tracked. In the second scenario we wanted to see how robust the algorithm was to an increase in ice density. Even if the iceberg density was 30 or 50 icebergs, the algorithm prioritized to search and track the closest areas and icebergs. We found that a higher ice density decreases the search radius for the UAV. Ice densities far above 50 can make the search radius so small that it does not detect icebergs before it is too late. We can therefore not guarantee that the algorithm has the desired performance when the ice density exceeds 50 icebergs. Through the third scenario, we wanted to test the algorithm's robustness to a rapid change in weather. We found that this increased the total area covered by the UAV. On the other hand, rapidly changing weather, gives unpredictable iceberg drift, which reduced the performance of the tracking part of the algorithm. It was concluded that the algorithm's total performance was degraded with rapidly changing weather, but that the likelihood that the weather will change this fast and random, is low. The last scenario focused on testing the tuning mechanism. The results indicated that the tuning resulted in the desired behaviour: When $a > b$, tracking was prioritized, and when $a < b$, search was prioritized.

Some final simulations were done to investigate how the integration of radar and satellite data would affect the combined algorithm's performance. For the radar case, the detected icebergs were added to the Kalman filter. This made a more safe system, but introduced more icebergs to be tracked for the algorithm. Because of the limitation on the number of nodes in the UAV's path, an alternative solution was proposed, removing the icebergs detected by the radar as soon as they got further away than 10 km from the vessel. Satellite images with different uncertainty were modelled in order to test how integration of real satellite data would affect the algorithm. The results indicated that the performance was

improved as long as the iceberg classification accuracy was above 75%. As for the radar case, this also introduced more icebergs to be tracked by the algorithm, which could become a problem. A solution dedicating a few missions after each update, to discover the icebergs on the satellite image, was proposed.

The results of this study support the viability of a combined search and track algorithm for UAV search and track missions. We have seen that there are still some issues to be solved, but under certain conditions and assumptions, the algorithm has the desired behaviour. Future work should therefore focus on relaxing these assumptions and conditions, making the algorithm more robust. To specify, the following tasks are proposed as future work:

- Investigate other ways of computing the optimal path, such that the maximum number of nodes in the path can be increased.
- Include estimation of the iceberg mass and shape into the algorithm, to further improve the iceberg tracking.
- Improve the algorithm, such that it becomes more robust to rapidly changing weather. One solution could be to have an emergency mode, which prioritizes differently under severe weather.
- Implement the proposed solution to handle the increased amount of icebergs to be tracked, when using a ship radar.
- Further investigate the possibilities of including satellite data into the algorithm, by using real data and classification tools.
- Perform experiments with a UAV for proof of concept.

Bibliography

[1] K. Eik (2008). *Review of Experience within Ice Management*. Presented in The Journal of Navigation. The Royal Institute of Navigation.

[2] Prepared for the Technology & Operations Subgroup (2015). *ICE MANAGEMENT*. Paper #6-8 in Arctic Potential: Realizing the Promise of U.S Arctic Oil and Gas Resources. Working Document of the NPC Study.

[3] E. Skjong, S.A. Nundal, F.S. Leira, and T.A. Johansen (2015). *Autonomous search and tracking of objects using model predictive control of unmanned aerial vehicle and gimbal: Hardware-in-the-loop simulation of payload and avionics**. Conference Paper for International Conference on Unmanned Aircraft Systems, Denver. Centre for Autonomous Marine Operations and Systems, NTNU.

[4] J.R. Riehl, G.E. Collins, and J.P. Hespanha (2011). *Cooperative Search by UAV Teams: A Model Predictive Approach Using Dynamic Graphs*. Presented in IEEE Transactions on Aerospace and Electronic systems. University of California, Santa Barbara.

[5] J. Haugen, and L. Imsland (2013). *Optimization-Based Autonomous Remote Sensing of Surface Objects Using an Unmanned Aerial Vehicle*. Presented in ResearchGate. Norwegian University of Science and Technology.

[6] A. Albert, and L. Imsland (2015). *Mobile Sensor Path Planning for Iceberg Monitoring using a MILP Framework*. Presented in ResearchGate. Norwegian University of Science and Technology.

[7] Copernicus Marine Environment Monitoring Service. URL: `http://marine.copernicus.eu/services-portfolio/access-to-products/`. Downloaded 31.01.2017.

[8] Google earth V 7.1.8.3036. (April 11, 2017). Greenland Sea / Svalbard, Norway. $78°02'40.30"N$ $12°43'45.24"E$, Eye alt. 637.91 km. Image Landsat / Copernicus, Image IBCAO.

[9] Google earth V 7.1.8.3036. (April 11, 2017). Barents Sea / Svalbard, Norway. 73°23′24.67"N 11°44′48.02"E, Eye alt. 4360.45 km. © 2016 Google, US Dept of State Geographer, Image Landsat / Copernicus, Image IBCAO.

[10] L.E. Andersson, F. Sciblia and L. Imsland (2017). *Draft: A Study on an Iceberg Drift Trajectory.* Presented on the ASME 2017 36^{th} International Conference on Ocean, Offshore and Arctic Engineering. NTNU, Trondheim.

[11] A. Barker, M. Sayed and T. Carriers (2004). *Determination of Iceberg Draft, Mass and Cross-Sectional Areas.* Presented on the International Offshore and Polar Engineering Conference (2004). Canadian Ice Service, Environment Canada, Ottawa, Canada.

[12] I.S. Stuart and J.D. Miller (1983). *Icebergs: Their Physical dimensions and the presentation and application of measured data.* Presented in Annals of Glaciology, volume 4. Petro-Canada Resources Inc., Alberta, Canada.

[13] I.D. Turnbull, N. Fournier, M. Stolwijk, D. McGonigal (2015). *Operational drift forecasting in Northwest Greenland.* Published in Cold Regions Science and Technology. Cenate Associates International Ltd, Alastair Ross Technology Centre, Alberta, Canada.

[14] S.D. Smith and E.G. Banke (1983). *The influence of winds, currents and towing forces on the drift of icebergs.* Published in Cold Regions Science Technology. Atlantic Oceanographic Laboratory, Bedford Institute of Oceanography, Dartmouth, Canada.

[15] B.F. Howell Jr. (1970). *Coriolis Force and New Global Tectonics.* Presented in the Journal of Geophysical Research Vol. 75, No. 14, May 10, 1970. Department of Geology and Geophysics, The Pennsylvania State University, University Park, Pennsylvania.

[16] T.J.W. Wagner, R.W. Dell and I. Eisenman (2016). *An analytical model of iceberg drift.* Presented in the Journal of Physical Oceanography. Institution of Oceanography, University of California San Diego, La Jolla, California.

[17] K. Eik (2009). *Iceberg drift modelling and validation of applied metocean hindcast data.* Presented in Cold Regions Science and Technology 57, 2009. NTNU, Norway.

[18] R.G. Brown and P.Y.C. Hwang (2012). *Introduction to Random Signals and Applied Kalman Filtering. p. 141-147.* John Wiley & Sons, Inc.

[19] Polar View Sea Ice Monitoring Service, Sentinel-1 image subset. Image acquisition date/time: 2017-05-03 07:03:59. 717375 UTC. Spheroid WGS84. Latitude of true scale: 70N.

[20] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein (2001). *Introduction to Algorithms, 2nd edition. Chapter 34 & 35.* The MIT Press.

[21] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein (2001). *Introduction to Algorithms, 2nd edition. Chapter 15.* The MIT Press.

[22] R. Bellman (1962). *Dynamic Programming Treatment of the Travelling Salesman Problem*.* Presented in JACM. RAND Corporation, Santa Monica, California.