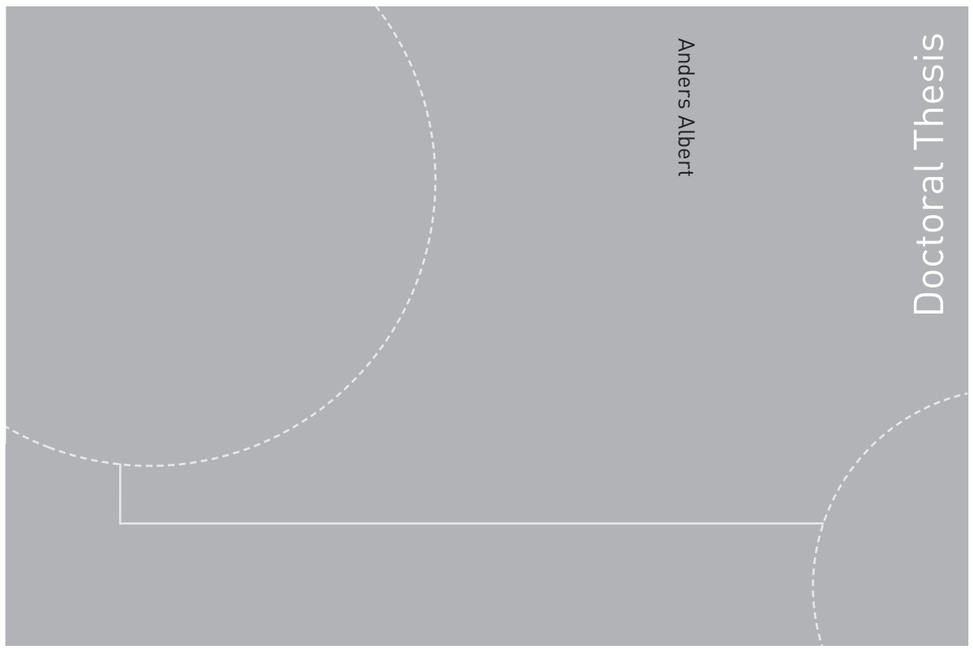
ISBN 978-82-326-3240-4 (printed version)
ISBN 978-82-326-3241-1 (electronic version)
ISSN 1503-8181



Doctoral theses at NTNU, 2018:225

Anders Albert

Unmanned Aerial Vehicle(s) Trajectory Planning for Target Searching and Tracking

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2018:9-W

NTNU

NTNU
Norwegian University of
Science and Technology

Anders Albert

Unmanned Aerial Vehicle(s) Trajectory Planning for Target Searching and Tracking

Thesis for the degree of Philosophiae Doctor

Trondheim, August 2018

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-3240-4 (printed version)
ISBN 978-82-326-3241-1 (electronic version)
ISSN 1503-8181

Doctoral theses at NTNU, 2018:225

Printed by Skipnes Kommunikasjon as

Summary

In this thesis, we introduce three trajectory planning algorithms for nonholonomic mobile sensors used for *target tracking* and the extended problem *target searchings and tracking*. These problems are motivated by the real-world application ice management, which includes searching for and tracking of icebergs. In addition to ice management, there are multiple other applications that can utilize similar problem formulations such as search and rescue, border patrol, traffic monitoring, environmental monitoring, combat scenarios, and wild animal tracking.

Chapter 2 of this thesis is a review of the literature on mobile sensor networks for target searching and/or tracking problems. In the review, the focus is on the trajectory planning and target filtering algorithms. The most common algorithms for each task are presented. After Chapter 2, the mobile sensor is assumed to be a fixed-wing unmanned aerial vehicle, and the number of mobile sensors is one for Chapter 4 and 5 and fewer than the number of targets for Chapter 3.

The contribution of Chapter 3 is a practical implementation of the target visitation algorithm, which is a combinatorial optimization formulation. This implementation is demonstrated both in simulation and a practical experiment. Compared to similar approaches it applies a static combinatorial formulation to a dynamic problem by making heuristic adjustments. In addition to demonstrate the result in a practical experiment. In Chapter 4, we use a cascaded formulation of an optimal control problem and show how to merge the equality constraints into the objective function. Then, we combine collocation and single shooting to get a implementation that can be more computationally effective than using collocation alone. The contribution of Chapter 5 is twofold. First, we derive a result that enable us to use set time constraints for how often a target must be revisited in the *target searching and tracking* problem. Second, we use this result combined with

the two techniques from Chapter 3 and 4 to make a path planning algorithm. We demonstrate the performance of the algorithm compared to multiple base cases in simulations. Compared to other approaches this algorithm applies both combinatorial and optimal control formulations. The motivation for mixing the two techniques is that optimal control problems often cannot solve non-convex problems for global optimality. Furthermore, mixed integer linear programming can solve smaller instances of these problems, but it is difficult include nonlinear constraints in the formulation. Combining these approaches manages to utilize the strengths of both techniques.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU). The research has been conducted at the Department of Engineering Cybernetics during the period between August 2013 and December 2017. My supervisor has been Prof. Lars Imsland and co-supervisor has been Prof. Thor Arne Johansen. This work has been part of the IME's Faculty's research project, "CAMOS Coastal and -Arctic Maritime Operations and Surveillance", and the research has also been supported by AMOS - Center for Autonomous Marine Operations and Systems.

Acknowledgements

I would like to thank my supervisor Lars Imsland for solid feedback as well as many useful professional discussions. In addition, I would like to thank colleagues and staff at the Department of Engineering Cybernetics. An extra thanks to Trond Andresen for an enjoyable and educational cooperation on the courses TTK4105 and TTK4220.

A special thanks to the Acrobatic Rock'n'Roll group at NTNUI and particularly my dance partner Line Nordsveen. My last nine years would not have been so much fun without you.

Two friends deserve special thanks. My brother Espen Albert and my friend Raymond Toft. You both had patience to discuss my research for hours as well as providing useful input. Thank you.

Finally, I would like to thank my family and girlfriend Nina Stien for all the support during the last four years.

Contents

List of Tables

List of Figures

List of Abbreviations

FOV	Field of view
ILP	Integer Linear Programming
MILP	Mixed Integer Linear Programming
MSN	Mobile Sensor Network
NLP	Nonlinear Programming Problem
OCP	Optimal Control Problem
TSP	Traveling Salesperson Problem
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

This thesis concerns trajectory planning for nonholonomic sensor platforms utilized for both *target tracking* and *target searching and tracking*. The real-world application is ice management, specifically the surveillance of drifting icebergs.

1.1 Problem Formulation

1.1.1 Target Tracking

When we use the term target tracking in this thesis, we will reference the following definition.

Definition 1 (Target Tracking)**.** *Target tracking is the process of utilizing a set of mobile sensors to continuously estimate the locations of a known number of targets given an initial estimate for each target.*

Target tracking can be an ambiguous term as it is used for multiple purposes. First, multi-target tracking is often used for the problem of data association. This is the problem of matching a set of measurements to a set of estimates. This is a subproblem in our definition of target tracking. Second, the term is also used within image processing. Here it relates to identifying features of a target in a video feed, and track it through the frames. This is a similar, but different problem, which usually utilizes different tools.

Notice that we refer to a *set of mobile sensors* in the definition of target tracking. This is usually called a mobile sensor network and can be defined as

Definition 2 (Mobile Sensor Network)**.** *A mobile sensor network is a set of agents with locomotion, communication and sensing ability.*

With the exception of the review in Chapter 2, we will not use the term mobile sensor, but instead unmanned aerial vehicle (UAV). A UAV is a type of mobile sensor, and more specifically we will be assuming a fixed wing UAV. Furthermore, we will focus on small networks, meaning fewer sensors than targets. In Chapter 4 and 5 we will only consider a single UAV.

A simple version of the target tracking can be illustrated as follows. Consider an open area with five moving targets. Then, with a priori estimate of the target's locations the task is to utilize one UAV with a limited field of view (FOV) to keep track of their locations. The situation is illustrate in Figure 1.1.

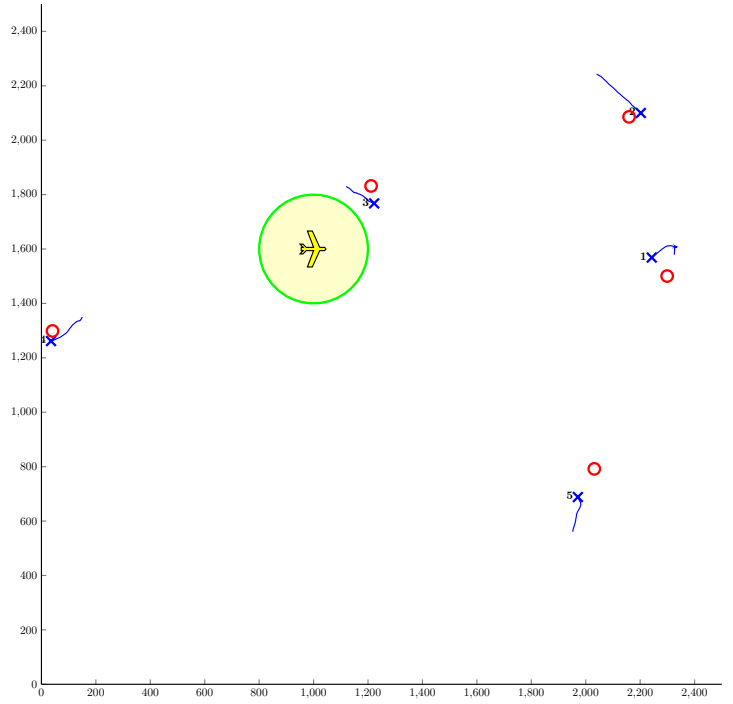


Figure 1.1: Illustration of simple target tracking scenario. Five targets are illustrated with numbers and blue X'es. Their recent movements are a solid blue line with a position estimate given as red circles. The sensor is drawn as a yellow polygon with a light-yellow to indicate the limited field of view.

There are multiple subproblems within target tracking. First, a typical way to estimate the location of each target is to have a set of filters. Say we use five filters for five targets. Upon observing target number three, it is not a trivial problem to match this observation to the correct filter. This is the problem of data association mentioned above. Another problem is communication. There might be multiple mobile sensors utilized for solving the problem with a limited communication

range. Measurements from the sensors need to be coordinated among the sensors and sometimes also with a base station. Third, the area can contain obstacles and no-fly zones. Although all these are important subproblems to consider in a real application, we will disregard them in this thesis. We will only be concerned with a centralized trajectory planning for the mobile sensors, except for the review in Chapter 2 where we also discuss distributed trajectory planning as well.

Throughout this thesis we will use the same models for the targets as well as the mobile sensors. The target model is a near-constant velocity model, which can be written as follows:

$$\dot{\boldsymbol{\xi}} = \begin{bmatrix} \dot{\boldsymbol{s}}_i \\ \dot{\boldsymbol{v}}_i \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \boldsymbol{\xi} + \boldsymbol{w}(t) \tag{1.1}$$

where $\xi \in \mathbb{R}^4$ is the state of the target consisting of position and velocity, $s \in \mathbb{R}^2$ and $v \in \mathbb{R}^2$. The target moves with a constant velocity except for a process noise of Gaussian distribution, which can be described as $\boldsymbol{w}(t) \sim \mathcal{N}(\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T, \boldsymbol{Q})$.

Even though we apply a specific model for the targets, except for chapter 4, we only utilize the position of the targets. This means we could have applied any model for the targets in the trajectory planning algorithms presented in this thesis. Even for the algorithm in chapter 4 it would be possible to use another model for the targets.

The mobile sensors are assumed to be nonholonomic in this thesis. A nonholonomic vehicle is a vehicle with fewer degrees of freedom than its movement space. Perhaps the most typical example of this is a Dubins vehicle, which is the kinematic model we use for the mobile sensors in this thesis:

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} U\cos(\psi) \\ U\sin(\psi) \\ u \end{bmatrix} \tag{1.2}$$

$$-u_{\text{lim}} \leq u \leq u_{\text{lim}}. \tag{1.3}$$

where $z \in \mathbb{R}^3$ is the state of the sensor consisting of position and heading, $x, y, \psi \in \mathbb{R}$. The actuator, u, is limited with $\pm u_{\text{lim}}$. This kinematic model is considered sufficient for modeling the nonholonomic sensor for the trajectory planning.

Target tracking will be discussed in more detail in Chapter 2, and in Chapter 3 and 4 we will suggest target tracking algorithms.

1.1.2 Target Searching and Tracking

Target searching and tracking is a generalization of target tracking, which can be defined as follows:

Definition 3 (Target searching and Tracking)**.** *Target searching and tracking is the process of utilizing a set of mobile sensors to find and continuously estimate the locations of an* $unknown$ *number of targets.*

In contrast from target tracking, this also involves the process of searching. This poses an additional challenge for trajectory planning as searching and tracking must be balanced. To illustrate the problem consider Figure 1.1 without the red circles indicating the initial position estimate.

This problem will be discussed in more detail in Chapter 2 and an algorithm will be suggested in Chapter 5.

1.2 Real-World Applications

The main real-world application in this thesis is ice management. This can be considered part of ice defense, which also contains activities such as breaking or towing ice as well as decisions involving ice (Haugen 2014). We will focus on searching and tracking icebergs, where UAVs are considered a cheap and efficient platform (Eik 2008, Lešinskis and Pavlovičs 2011).

We use two concrete scenarios for illustration. First, a boat traveling through arctic areas. Here, it is important for the boat to know the locations of icebergs in its vicinity. A set of UAVs can be utilized to search the area around the planned path of the boat for icebergs and report back to the boat. This enables the boat to plan its course while taking icebergs into consideration, see Figure 1.2.

Another scenario is drilling in arctic areas. Consider the situation illustrated in Figure 1.3. We suggest that the drilling buoy has three levels of safety zones. For example, in the outer zone (green) icebergs are monitored. If an iceberg enters the middle zone (yellow) an action like breaking or towing is prompted. Finally, if an iceberg enters the inner zone (red) drilling operation is shut down and the rig is prepared for impact. Here, one or multiple UAVs can be used for the monitoring.

In addition to iceberg tracking there are multiple other applications that can utilize problem formulations that can be classified as target searching and/or tracking. First, we have search and rescue operations (Latif et al. 2016). Consider the situation after a shipwreck. In the surrounding area of the wreck, there can be multiple lifeboats and people floating in the water. Then, UAVs could assist the rescue operations by searching and tracking the position of the objects floating in the water. Another application is border patrol (Girard et al. 2004). Here, UAVs can be utilized to cover a large border area, and help redirect limited resources to stop illegal crossings. An application less relevant to trajectory planning is wild life tracking (Juang et al. 2002, Lalooses et al. 2005). Here, the animals can serve as mobile

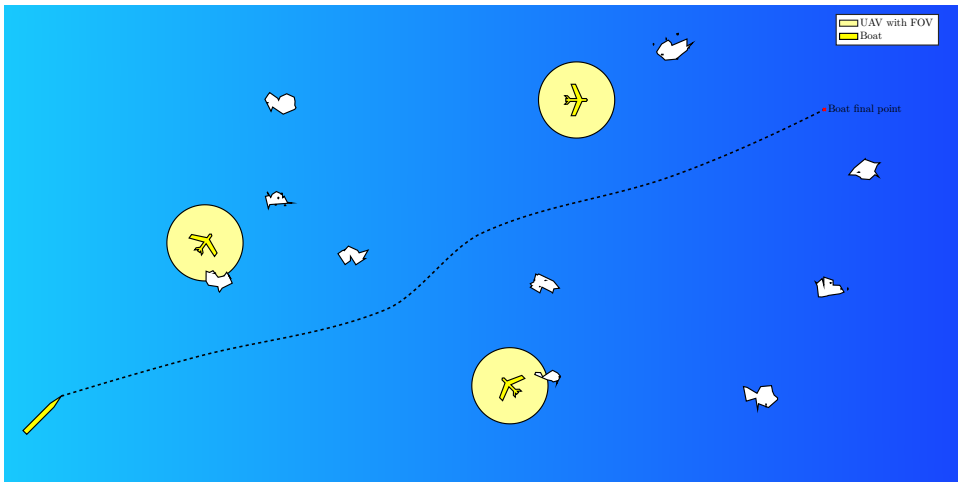


Figure 1.2: Iceberg management. Scenario 1. A boat travelling, yellow polygon, in an arctic area with floating icebergs. Three UAVs, yellow polygons with circular field of view, search for and track icebergs near the planned path for the boat. Icebergs are illustrated as white irregular shapes.

platforms by mounting sensors on them, where the goal is to learn movement patterns of the animals. However, it is also possible to consider UAVs tracking the animals from the air, which would require trajectory planning. In urban environments, traffic monitoring can be accomplished by UAVs (Puri 2005). This also requires trajectory planning based on target tracking. In addition to the civilian applications, target searching and tracking are relevant for military operations (Glade 2000). For example, in a combat scenario it would be useful to know the size and position of enemy forces.

1.3 Optimization Methods

In this thesis, we apply two different optimization techniques. First, we apply mixed integer linear programming (MILP). This is a kind of optimization problem where at least some of the variables are restricted to be integers and the objective function is linear. The perhaps most famous problem, which is also applied in this thesis, is the traveling salesperson problem (TSP). Second is optimal control problems (OCP). This was developed in parallel in the U.S. and the Soviet Union during the Cold War. The work was led by the mathematicians Pontryagin and Bellman (Pesch et al. 2009). OCP is concerned with finding an actuator input given an optimal criteria.

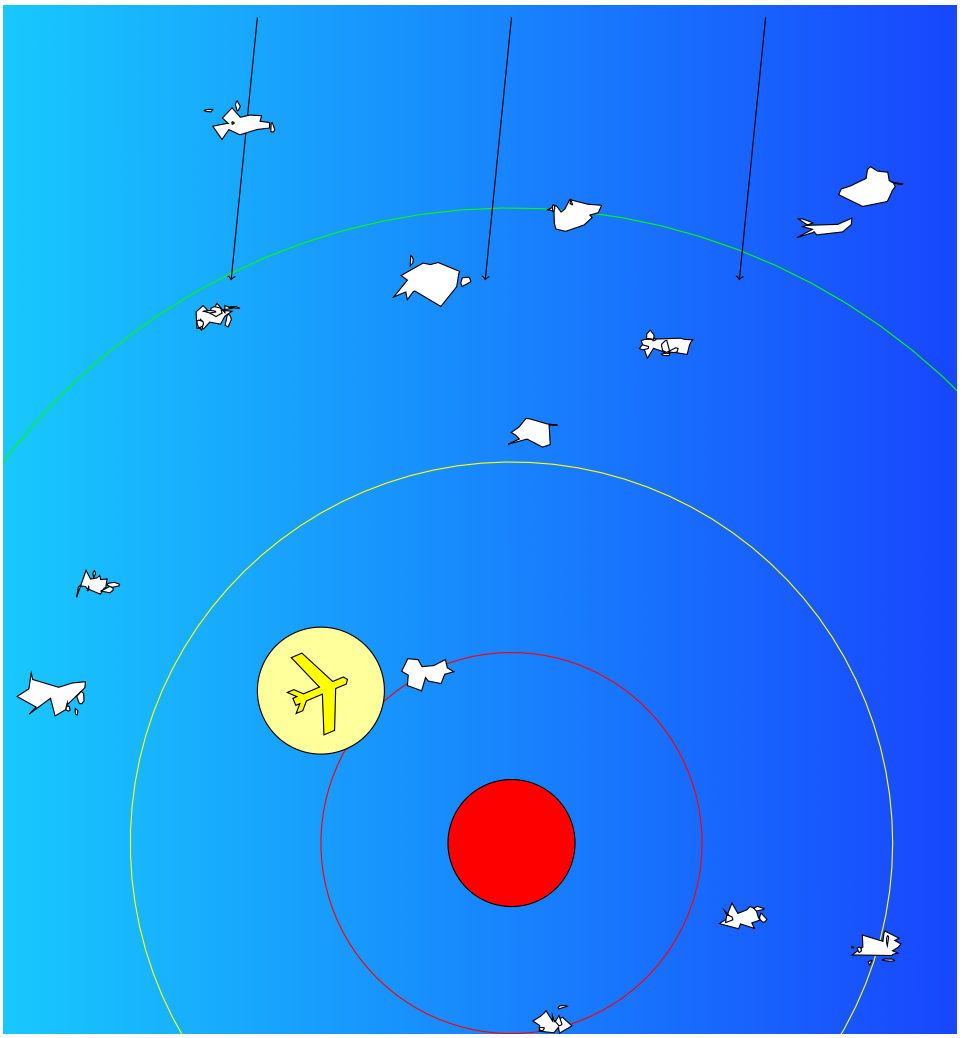


Figure 1.3: Iceberg management. Scenario 2. Drilling rig is illustrated with red circle. Three levels of safety zones are drawn with solid lines colored red, yellow and green. The weather direction is drawn as three black arrows.

1.3.1 Mixed Integer Liner Programming

A general mixed integer linear programming (MILP) problem can be written as

$$\min_{x\in\mathbb{R}^n, z\in\mathbb{Z}^m} f(x,z) \tag{1.4a}$$

s.t.

$$g(x,z) = 0 \tag{1.4b}$$

$$h(x,z) \leq 0 \tag{1.4c}$$

Problems that only use integer variables are often referred to as integer linear programs (ILP), which is a subclass of MILP. For example, TSP is an ILP. Given a linear objective function an ILP can be written

$$\min_{z\in\mathbb{Z}^m} \sum_i c_i z_i \tag{1.5a}$$

s.t.

$$\sum_j a_{ij} x_j \leq b_i \quad \forall i = [1, ..., m] \tag{1.5b}$$

where c_i is the cost of variable z_i. It can be written as a vector $\boldsymbol{c} \in \mathbb{R}^m$. The parameters a_{ij} and b_i are the constraints, and can be written as a matrix and a vector, $A \in \mathbb{R}^{m\times m}$ and $\boldsymbol{b} \in \mathbb{R}^m$.

MILP problems are often exponential in nature, which makes them hard to solve. Fortunately, for the problem size presented in this thesis a modern MILP solver can solve it sufficiently within reasonable time. We apply MILP formulations in both Chapter 3 and 5 and in both cases we utilize IBM's CPLEX (IBM 2015). For more details on integer programming see Chen et al. (2010).

1.3.2 Optimal Control Problem

An optimal control problem (OCP) can be written as

$$\min_{x(\cdot),u(\cdot)} \int_{t_0}^{T} L(x,u)dt + E(x(T)) \tag{1.6a}$$

s.t.

$$\dot{x} = f(x,u) \quad x(0) = x_0 \tag{1.6b}$$

$$h(x,u) \geq 0 \tag{1.6c}$$

$$r(x(T)) \geq 0 \tag{1.6d}$$

The term $L(x,u)$ is often referred to as the Lagrange term, while the terminal cost, $E(x(T))$, is typically called the Mayer term. In this optimization problem $f(x,u)$

is the dynamic function of the state and actuator, $x(t), u(t)$. The constraints are noted $h(x, u)$ with terminal conditions set to $r(x(T))$ The control horizon is noted $t \in [t_0, T]$.

To solve OCP there are in general two tactics. Either first "discretize and then optimize", the direct approach, or "optimize and then discretize", the indirect approach. In this thesis, we will be utilizing the former approach, which again can be divided into single and multiple shooting as well as collocation. The difference between these methods are how the dynamic equation, (1.6b), is approximated with algebraic equations. The two methods applied in this thesis are single shooting and collocation. Single shooting is a sequential method, which uses an integration method for approximating the dynamic equation. Collocation instead approximates this with a polynomial of a set degree. Furthermore, the collocation method we use divides the time horizon into intervals and approximates the dynamic equation with polynomials over each interval.

In all direct approaches the OCP from equation (1.6) is transformed into a large nonlinear programming problem (NLP), which can be written

$$\min_{x \in \mathbb{R}^n} f(x) \tag{1.7a}$$

s.t.

$$g(x) = 0 \tag{1.7b}$$

$$h(x) \leq 0 \tag{1.7c}$$

Depending on the method we have chosen, the NLP will have different properties. If we use single shooting, it will typically be small and dense, and an active set method will most likely be the most efficient for solving it. For collocation, the NLP will typically be large and sparse, and an interior point method will probably solve it more efficiently.

In this thesis, we use an interior point method called IPOPT (Wächter and Biegler 2006). To formulate the problem, we utilize CasADi (Andersson 2013a). For more information about optimal control see Biegler (2010) and Betts (2010).

1.4 Outline and Contributions

This thesis is divided into four parts, where each part can be read independently.

Chapter 2 presents a literature review on the problems of target searching, target tracking and the combination of both utilizing a mobile sensor network. This work focuses on the trajectory planning algorithm for the mobile sensors (high-level controller) as well as the filter algorithms for estimating the states of the targets.

This work is based on Albert and Imsland (2018b).

Chapter 3 introduces a trajectory planning for multiple UAVs solving a target tracking problem. The algorithm utilizes a MILP formulation and the CPLEX solver. Compared to other approaches from the literature, this algorithm manages to utilize a static combinatorial formulation for a dynamic problem by utilizing heuristic adjustments. It also use a practical experiment for demonstration, which few other approaches from the literature use.

This work was published in Albert et al. (2017), which is an extension of Albert and Imsland (2015).

Chapter 4 contains a comparison between state-of-the art and our suggested approach to implementing optimal control problems with a cascade form. The typical approach is to use collocation alone, while we use combination of collocation and single shooting. The second method can be more computationally efficient, which is demonstrated in a target tracking formulation with a single UAV.

This work was published in Albert et al. (2016).

Chapter 5 suggests a trajectory planning algorithm for target searching and tracking for a single UAV. This algorithm uses a two-layer formulation, one containing a MILP- and the other an optimal control formulation. The motivation is to use a MILP optimization to find a near-optimal trajectory and then use it to initialize the optimal control problem such that the resulting NLP is optimal while considering the movement constraints of the UAV. Compared to other state-of-the-art methods this combines to well known techniques to mitigate the shortcoming of each individual technique. In addition, we present a result called the necessary visitation period. This enable us to use characteristic of a target to calculate how often it must be visited to have the estimation error of the target's position within set limits.

This work is based on Albert and Imsland (2018a) and Albert and Imsland (2017).

1.5 Publications

This thesis is based on the following publications

- Albert and Imsland (2015) Mobile sensor path planning for iceberg monitoring using a MILP framework. In *Informatics in Control, Automation and Robotics (ICINCO)* 12th International Conference on (Vol. 1, pp 131-138). IEEE

- Albert et al. (2016) Numerical Optimal Control Mixing Collocation with Single Shooting: A Case Study *IFAC-PapersOnLine*, 49(7), 290-295

- Albert et al. (2017) UAV Path Planning using MILP with Experiments. *Modeling Identification and Control*, 38(1), 21-32.
- Albert and Imsland (2017) Performance bounds for tracking multiple objects using a single UaV. In *Unmanned Aircraft Systems (ICUAS)*, 2017 International Conference on (pp. 1539-1546). IEEE
- Albert and Imsland (2018a) Searching and Tracking of Multiple Moving Objects using a Single UAV. Accepted for publication in *Journal of Intelligent & Robotic Systems*
- Albert and Imsland (2018b) Survey: Mobile Sensor Networks for Target Searching and Tracking. *Cyber-Physical Systems* 2018, 1-42

Chapter 2

Survey: Mobile Sensor Networks for Target Searching and Tracking

This work is based on Albert and Imsland (2018b).

Mobile sensor networks can be employed in multiple applications, such as search and rescue, border patrol, battle scenarios, and environmental monitoring. In this survey, we review the literature utilizing mobile sensor networks in applications classified as target searching and/or tracking. Our contribution is threefold. First, we focus on the diverse types of filters applied to estimating the state of the targets. Second, we present the most common approaches to high-level trajectory planning for the sensors in the network to do target searching and/or tracking. Finally, we classify the literature based on the problem formulation used and solution characteristics. At the end of the survey, we discuss the current state of the literature and possible directions for future research efforts.

2.1 Introduction

2.1.1 Mobile Sensor Network

A mobile sensor network (MSN) is *any group of agents where at least some have locomotion ability along with sensing and communication abilities*. The agents are normally referred to as sensor platforms or just sensors. A platform can be equipped with one or multiple sensors in addition to processing and communication units. An example can be a group of unmanned aerial vehicles (UAVs), each

equipped with a camera used to monitor traffic (Zhang et al. 2015). In this chapter, we will focus on controllable sensors. However, the sensor platforms do not need to be controllable. For example, humans in a city could serve as sensor platforms, and smart phones with environmental sensors could be utilized in an application to gather local weather data.

2.1.2 Target Searching and Tracking

Target searching and tracking (TST) *is the problem of estimating the location of one or multiple targets in a given area*. The area is referred to as the environment and can be an urban area, the sea surrounding a drilling platform, a forest, and so on. It can also contain obstacles, no-fly zones, hostiles, and so forth. The targets might be mobile or stationary and/or evasive or non-evasive. Examples of targets are people, icebergs, fire border (wildfire), and animals. There are numerous real-world applications, such as search and rescue, border patrol, battle scenarios, wildlife tracking, environmental monitoring, such as ice management, wildfire, and traffic, that can be classified as target searching and tracking (Jadaliha and Choi 2013, Juang et al. 2002, Lalooses et al. 2005, Latif et al. 2016, Lesinskis and Pavlovics 2011, Pereira et al. 2009, Zhao et al. 2014b). The recent availability of cheap sensor platforms like Unmanned Aerial Vehicles (UAVs) make Mobile Sensor Networks relevant for these applications.

An MSN is well-suited for TST. The multiple sensor platform can spread out across the environment and move to make up for a limited field of view. A typical TST scenario is described in the 2D-plane. An example is illustrated with Figure 2.1, in which four sensors with communication constraints search for and track twelve moving targets.

2.1.3 Motivation

UAVs as sensor platforms and suitable sensor payloads are becoming cheaper and more available. There are already numerous companies offering UAV sensor platforms and payloads for different applications (Aeryon 2007, ProxDynamics 2008, Sensefly 2009). Most of these solutions require a human operator to control the UAV. However, to exploit the full potential of a UAV or multiple UAVs acting as sensor platforms, it is necessary for them to operate more autonomously. A human operator per sensor platforms is less efficient than having the sensor platforms report back only the interesting information to the human operator. For a set of sensor platforms to do high-level tasks such as search for and track targets more advanced algorithms are necessary, which we will discuss in detail in this survey.

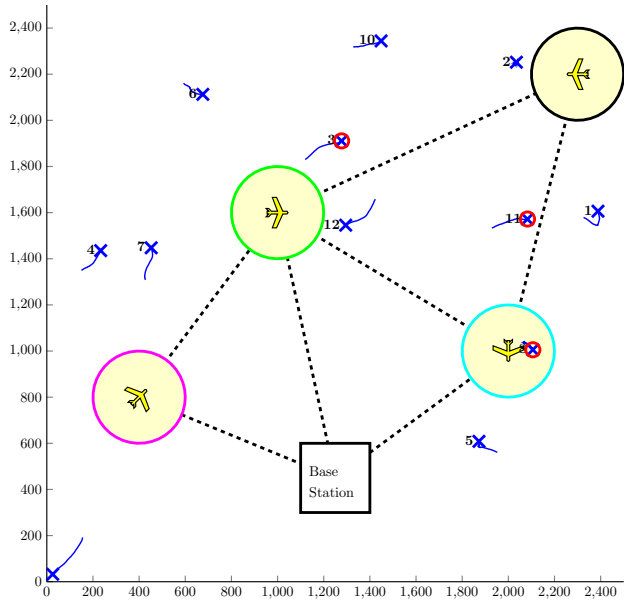


Figure 2.1: Example Target Searching and Tracking scenario with four sensors and twelve targets. Each sensor is a plane-shaped yellow polygon with a light-yellow circle around to indicate its limited field of view (FOV). The color of the border of the FOV differentiates sensors from each other. The targets are marked with blue numbered X's and a short tail for recent movement. Three targets also have a red circle around them, indicating their estimated position. The base station is a black square, and the dotted black lines indicate the communication links between the sensors and base station.

2.1.4 Contribution and literature appraisal

There are many options and challenges in using MSNs for target searching and tracking problems. Selecting an appropriate sensor, communication protocols, autopilot for each sensor platform are some of these. In most papers dealing with TSTs, the focus is usually on either high-level trajectory planning, the target state observer, or both. These are both algorithm designs and will be the focus of this chapter. *Trajectory planning* is a high-level control algorithm which can be executed in a centralized or decentralized fashion, determining the path for each sensor platform. It is sometimes referred to as *path planning*. However, this term is also used for the problem of finding the path from A to B for an agent while avoiding obstacles. Another commonly used term is *Motion Planning*, but this term is even more general, which is why we will use the term trajectory planning. The *target state observer* is the estimation algorithm that uses the raw data from a sensor to estimate the state, usually the position and possibly velocity, of the targets. This is often referred to as a filter because it takes in raw measurements

and aggregates them into, for example, location estimates. As far as the authors know, there have been no surveys on mobile sensor networks for the application of TST. We have limited the literature search to the combination of the phrases "mobile sensor network" and "target" in the title, keyword, or abstract of a paper. We have reviewed more than 300 papers, whereof approximately 100-150 have been included in this chapter.

2.1.5 Previous Surveys

Several surveys have appeared with a focus on MSNs. The survey of Amundson and Koutsoukos (2009) is centered around localization, which is the problem of estimating each sensor's position. Another survey is Zhu et al. (2014), which focuses on communication and data management issues. The authors also discuss different applications, including target tracking. However, the discussion is limited to *single* target tracking. In this chapter, we will also include multi-target tracking, as well as the issue of searching. In Ma and Tan (2013), the authors discuss MSNs in an application context, and also include a section on TST. However, they mainly focus on the type of measurement device applied and separate searching and tracking into different categories. In contrast, we are also interested in the challenges that arise in the combination of searching and tracking. Another distinction is that we focus on high-level trajectory planning, and not the type of measurement devices. The survey conducted by Dagdeviren et al. (2011) is on wireless sensor networks (WSN), but it also contains a section on MSN, identified as *mobile hardware agents*. Here the focus is on localization, communication, and energy harvesting.

In these surveys, the authors present taxonomies for distinct types of MSNs. In this chapter, we are interested in MSNs in which we can actively control each sensor. In the terminology of the above papers, this is called *mobile actuated sensor networks* (Amundson and Koutsoukos 2009), *controllable movement* (Zhu et al. 2014), and *controllable mobility for active sensing* (Ma and Tan 2013).

The similar problem of target tracking has been well studied within the field of wireless sensor networks (WSN), and several surveys have been published (Akyildiz et al. 2002, Naderan et al. 2009; 2012, Souza et al. 2016, Yick et al. 2008). Although target tracking with WSN's might have similar applications to target tracking using MSNs, this can be a very different approach. WSNs are usually decentralized, and the focus is on the communication protocol. Properties that are emphasized are energy consumption, communication range, bandwidth, limited storage and processing power. For structured WSNs sensor deployment is also a central task. All of these properties can be considered in MSN research. However, the sensor used for these networks is often expected to have more energy storage,

processing power, and so on. The focus is typically on the mobility of the sensor and how to exploit this in target searching and tracking.

In addition to surveys on MSNs and WSNs there are several papers that deal with similar problems from other perspectives. From the robotics community, Robin and Lacroix (2016) discuss the problem of target searching and tracking, which they call target management. The taxonomy the authors present starts by dividing searching and tracking into two different categories, as Ma and Tan has done (Ma and Tan 2013). Compared to our survey, we also consider target searching and tracking combined. In addition, Robin and Lacroix (2016) is centered around the different ways TST problems are formulated, while our survey focuses on trajectory planning and target estimation. Another survey from the robotics community is Portugal and Rocha (2011), in which the authors discuss the application of patrolling and take a graph theory approach. The survey is centered around control algorithms, and, in contrast to this survey, it focuses only on solving target search using graph theory. Here we will also include other approaches, such as continuous optimal control, gradient-based field, flocking control, and so on. A similar problem, pursuit-evasion, which is also from robotics, is studied by Chung et al. (2011). These are situations in which one or multiple searchers pursue one or multiple targets trying to avoid detection. The authors also discuss applications in which the targets are unaware of the sensors. For more information on literature dealing with just searching, see Stone et al. (2016).

2.1.6 Organization

This chapter is organized as follows. In Section 2.2, we discuss the overall tasks of the observer and trajectory planner in search and track problems. We go into detail about the distributed and centralized architecture in Section 2.3. Section 2.4 contains the different filters for target state estimation, with Section 2.5 discussing trajectory planning algorithms. We classify the papers presented in the two previous sections in Section 2.6 based on problem formulation and solutions characteristics. In Section 2.7, we discuss the current state of the literature and future work. A short conclusion is given in Section 2.8.

2.1.7 Terminology

In the literature, there are many different terms used in the description of target searching and tracking. This partially stems from the many different approaches that are applied to these problems. Each row of Table 2.1 lists different terms used for the same concept, with the term used in this chapter listed first and in bold. In addition, we will commonly use sensor to mean both the physical vehicle as well as its onboard sensor, processor, communication unit, and so on. However, in some

contexts, such as in Section 2.3, we will use sensor platform only in reference to the physical vehicle. Then, sensor refers to the unit producing measurements.

Terms
target, object
search, detection
tracking, monitoring, coverage
trajectory planning, path planning, motion planning, high-level control algorithm

Table 2.1: Terms used to describe similar concepts in the literature. The term used in this chapter is first and in bold.

2.2 Main components of Mobile Sensor Network for Target Searching and Tracking

We can divide a target search and track problem into two main tasks: observer and trajectory planning. The observer's task is to use sensor data to produce estimates of each target's state, which typically consists of position and velocity. It can also involve synchronizing estimates between multiple observers in a distributed approach, see Section 2.3. The architecture of an observer depends on the number of targets. The most common approach is to use a filter to estimate the state of each target separately. This is illustrated in Figure 2.2. We will be discussing the different filters in Section 2.4. A problem that arises in multiple target tracking is to match measurements with the correct filter. This is called the data association problem and we will discuss it in Section 2.4.7. Another possibility is to combine all filters into one large filter, which is usually decoupled. This approach is rarely used, except in cases where the target's behaviors are actually coupled. Finally, the observer can also include a filter for estimating the state of the environment, called a map, which is also illustrated in Figure 2.2. This can include information about which areas have been explored, obstacle locations, and so on.

Trajectory planning is the task of utilizing the movement of an MSN for searching for and tracking targets. There are multiple factors that impact the design of a trajectory planner. First, it can be distributed or centralized. This will be discussed in more detail in Section 2.3. Another factor is the number of targets versus the number of sensors. Here, there are typically three scenarios (we do not discuss single sensor and single target), which are all listed in Table 2.2. When there are more sensors than targets, Scenarios 1 and 2, at least one sensor can be assigned

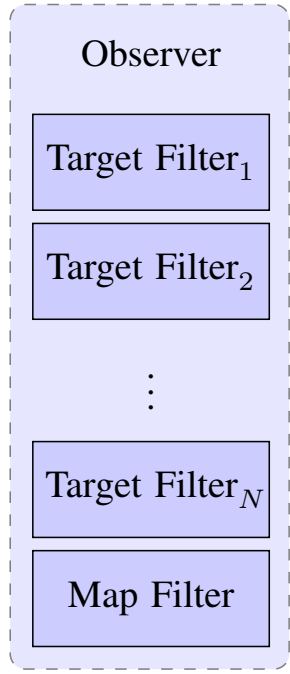


Figure 2.2: Observer architecture. Typically, one filter is used for each target. In addition, sometimes a filter is used to track environmental information.

to exclusively track each target. If there is available initial information about the location of the targets, the task is often exclusively to track the targets with the available sensors. If the positions of the targets are not known initially, but they are considered stationary, then the task is reduced to a search-only problem. However, in the case where there are fewer sensors than targets, Scenario 3, a trajectory planner has to balance the search for versus tracking of targets. In Section 2.6, we classify all the papers presented in this chapter based on distributed/centralized approaches, the number of targets, as well as whether or not the trajectory planner considers tracking, searching, or a combination of both.

In addition, a trajectory planner must consider different types of constraints. For example, a fixed-wing, unmanned aerial vehicle cannot move sideways, which leads to sensor dynamic constraints. Furthermore, there might be obstacles in the environment, and the sensors must avoid collision, both of which are typically implemented as constraints.

Scenario	Number of targets		Number of sensors
1	One	$<$	Multiple
2	Multiple	$<$	More
3	Multiple	$>$	Fewer

Table 2.2: Different scenarios with respect to the number of targets vs sensors.

2.3 Control Architecture

We separate between centralized and distributed approaches, which the control architecture mirrors. In a centralized approach, the sensor platforms are expected to be connected to the base station at all times. It is not necessary for each sensor platform to have much processing power as both the trajectory planning and data filtering can be done by the base station. However, in most cases the sensors platform might do preliminary aggregation of the measurements. For example, if the sensor is a camera, an onboard processor can do the image processing and send only the estimated position of the targets to the base station. Figure 2.3 illustrates a centralized approach.

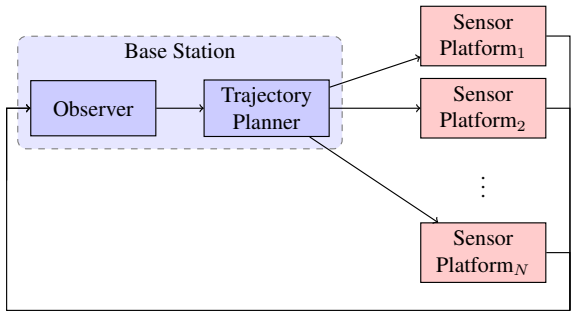


Figure 2.3: Centralized architecture. The solid lines indicate constant connections. The arrows illustrate the way data travel.

In a decentralized approach, each sensor platform must perform both trajectory planning and measurement filtering onboard. In addition, they must coordinate with other platforms, while the base station tries to combine all available target observations from the sensor platforms. The focus of decentralized approaches is typically to have simple control algorithms, which do not exceed the computational power of the processing unit onboard each sensor platform. This approach is illustrated in Figure 2.4.

Whether to choose a centralized or decentralized architecture depends on the ap-

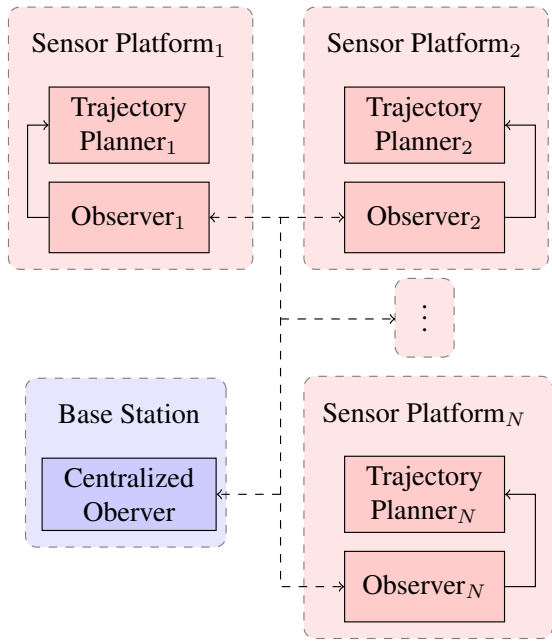


Figure 2.4: Decentralized architecture. The solid lines indicate constant connections, while the dashed lines indicate partial connections. This means that sometimes they are connected and other times not. The arrows illustrate the way data travel. Notice that data travels both ways between all the observers.

plication. Often it is desirable to have a centralized architecture, as a decentralized architecture usually gives suboptimal solutions for the trajectory planning and makes it difficult to coordinate target measurements. In that case a decentralized might seem desirable only when no other option is available. However, there might be multiple reasons to choose a decentralized architecture. First, in the case of unreliable communication between sensors, a decentralized can take this into account and thus be more robust. Second, if there is a lag in communication between the base station and sensors, a decentralized architecture will react faster to changes in the environment. Third, coordinating multiple sensors can become an intractable problem, depending on the planning algorithm, and a decentralized architecture can be the only tractable option. Finally, the choice of architecture can come down to cost. Sensors with better communication range and more computing power is generally more expensive and require more power.

2.4 Target State Filters

The two most popular algorithms for estimating the state of targets in mobile sensor networks are Kalman and particle filters. In this section, we will first present the basic idea for each filter. Then, we will discuss the distributed version of these filters, which are often applied in decentralized approaches. In addition, we will also briefly discuss some of the other filters applied in the literature.

2.4.1 Linear Kalman Filter

The Kalman filter is named after Rudolf E. Kálmán (Kalman et al. 1960) and it is the optimal filter for linear systems with Gaussian noise (Gelb 1974). The Kalman filter produces two outputs, a state estimate and an associated covariance matrix, which quantifies the uncertainty of the estimate. The filtering process contains two steps, executed recursively in time: First, in the a priori step, the previous estimate and covariance matrix propagate forward in time. In the second step, the a posteriori step, the new measurement is processed by combining it with the a priori estimate in a weighted average, depending on the uncertainty description of the new measurement and covariance matrix of the priori step. In addition, the covariance matrix is also updated.

We present the discrete version of the Kalman filter. Let x_k be the true state of a target at time k. Furthermore, let A be the transition matrix, and $w_k \sim \mathcal{N}(0, Q)$ be the process noise with zero mean and covariance $Q = Q^T \geq 0$. We combine this with the measurement, y, and its model, H, with noise $v_k \sim \mathcal{N}(0, R)$ also with zero mean and covariance $R = R^T \geq 0$, and get the following model and measurement equations

$$x_{k+1} = Ax_k + w_k \tag{2.1a}$$
$$y_k = Hx_k + v_k \tag{2.1b}$$

We use $\hat{x}$ to describe the state estimate and P the covariance matrix. To separate the a priori and posteriori step, we write $\hat{x}_{k|pri}$ and $\hat{x}_{k|post}$, and do the same for the covariance. The Kalman update equations are

$$\hat{x}_{k+1|pri} = A\hat{x}_{k|post} \tag{2.2a}$$
$$P_{k+1|pri} = AP_{k|post}A^T + Q \tag{2.2b}$$
$$\hat{x}_{k+1|post} = \hat{x}_{k+1|pri} + K_k(y_k - H\hat{x}_{k+1|pri}) \tag{2.2c}$$
$$P_{k+1|post} = P_{k+1|pri} - K_k H P_{k+1|pri} \tag{2.2d}$$

where

$$K_k = P_{k+1|pri}H^T(HP_{k+1|pri}H^T + R)^{-1}$$

Notice here that we do not include the initial conditions as these will vary for each case.

There are some centralized approaches which use the linear Kalman filter in target searching and tracking applications. Typically, one Kalman filter is used for each tracked target. For example, Cheng et al. (2012), Prabhavathi and Rajeshwari (2011) and Bai et al. (2012) all utilize linear Kalman filters to estimate the target states. In addition, both Haugen and Imsland (2016) and Albert et al. (2017) use Kalman filters to estimate the state of moving icebergs with unmanned aerial vehicles acting as a mobile sensor network.

While the covariance of the measurements, R, can usually be based on characteristics of the measurement device, the process covariance, Q, cannot. In all the above papers, the targets are simulated and the authors themselves selects the covariance Q of the process noise and assume it to be known to the filter. Usually, there is no discussion on how it should be obtained in a real-world example. Another important point is that the Kalman filters assume that the measurement and process noise to be Gaussian distributed, and there is usually no discussion on whether this assumption will be valid either. Finally, a linear Kalman filter also assumes a linear model, which might limit the applicability to a real-world example.

2.4.2 Extended Kalman Filter

The extended Kalman filter is a generalization of the non-linear target and measurement model. These models can be written as

$$x_{k+1} = f(x_k) + w_k \tag{2.3a}$$
$$y_k = h(x_k) + v_k \tag{2.3b}$$

where $f(x_k)$ is the transition function, and $h(x_k)$ is the measurement function.

This leads to a few changes is the Kalman update equations

$$\hat{x}_{k+1|pri} = f(\hat{x}_{k|post}) \tag{2.4a}$$
$$P_{k+1|pri} = A_k P_{k|post} A_k^T + Q \tag{2.4b}$$
$$\hat{x}_{k+1|post} = \hat{x}_{k+1|pri} + K_k(y_k - h(\hat{x}_{k+1|pri})) \tag{2.4c}$$
$$P_{k+1|post} = P_{k+1|pri} - K_k H_k P_{k+1|pri} \tag{2.4d}$$

where

$$K_k = P_{k+1|pri} H_k^T (H_k P_{k+1|pri} H_k^T + R)^{-1}$$
$$A_k = \frac{\partial f}{\partial x}\Big|_{\hat{x}_{k|post}} \qquad H_k = \frac{\partial h}{\partial x}\Big|_{\hat{x}_{k+1|pri}}$$

A well-known problem with the extended Kalman filter is its dependence on linearization, which can lead it to diverge if the initial state estimate is incorrect.

The extended Kalman filter is mainly used by mobile sensor networks to handle non-linear measurement models, (2.3b), while the target model, (2.3a), is usually linear. This is the case for both Ren et al. (2016) (distance-to-target measurement model) and Wu et al. (2014) (range-bearing sensor model). In Ren et al. (2016) the authors also exploit the covariance matrix, P_k, in their control law. Martínez and Bullo (2006) use a general target model in their derivation, and in simulation, use an 8-shaped movement for the target. All these papers use a centralized implementation of the extended Kalman filter.

The same critique as for the linear Kalman filter applies to the extended Kalman filter, except that extended Kalman filer is able to utilize nonlinear models. In addition, the extended Kalman filter has several challenges. First, the convergence of the estimation error to zero depends on the initial state estimate. In Ren et al. (2016), the authors use a sparsity decomposition scheme to initialize the targets positions to ensure that the estimation error converges to zero. InWu et al. (2014) and Martínez and Bullo (2006) there is less discussion on how to initialize the target positions. Second, in several of the mentioned papers the covariance matrix is used in the high-level trajectory planning as a measure of uncertainty of the estimate. Unfortunately, as the covariance is propagated linearly, equation (2.4b), in a nonlinear system this might be a poor quantification of measurement quality. Finally, in the extended Kalman filter the current state estimate is used to linearize the model. This means that if the current estimate is off, it can make the filter diverge.

2.4.3 Distributed Kalman Filter

In mobile sensor networks, it is often desirable to not have a centralized implementation of the observer, as this requires the network to always be fully connected. Olfati-Saber (2009) has developed two distributed versions of the linear Kalman filter, which he calls the Kalman consensus filter and Kalman Information consensus filter. The first is optimal, but scales with $O(n^2)$, while the second is suboptimal and scales with $O(n)$, where n is the number of sensors.

Compared to the Kalman filter from equation (2.2), the a priori step stays the same, while the posteriori step changes. Here, we only include the state update for a single sensor

$$\hat{x}_{i|post} = \hat{x}_{i|pri} + K_i(y_i - H\hat{x}_{i|pri}) + C_i \sum_{j \in N_i} (x_{j|pri} - x_{i|pri}) \tag{2.5}$$

We drop the subscript k, which indicates time, and instead use the subscripts i

and j to note the current sensor and its neighboring sensors. This means that, for sensor i we have $\hat{x}_{i|post} = \hat{x}_{k+1|post}$ and $\hat{x}_{i|pri} = \hat{x}_{k+1|pri}$. In addition, the Kalman gain K_i of sensor i will also depend on the sensor's neighbors. Finally, we have the gain C_i which controls the trade-off between consensus of the sensors and stability of the filter. In Olfati-Saber (2009), a suitable value is suggested, which provides global asymptotically properties.

Olfati-Saber applies the Kalman consensus filter to a mobile sensor network of 20 sensors in a single-target tracking problem (Olfati-Saber and Jalalkamali 2012). He uses the estimation, together with a distributed flocking algorithm. This result was later extended to include multi-target tracking with a coupled estimation and flocking control algorithm (Jalalkamali and Olfati-Saber 2012). Another similar distributed filter, also with a flocking algorithm, is Su et al. (2017). Here, the authors study the problem of tracking two coupled targets. The work on the distributed linear Kalman filter of Olfati-Saber has also been extended to include cubature Kalman filters and applied to mobile sensor networks for single-target tracking (Tan et al. 2017). A cubature Kalman filter is a nonlinear filter for high-dimensional state estimation that exploits cubature points to numerically calculate multivariate moment integrals (Arasaratnam and Haykin 2009). Another extension of Olfati-Saber's work is coupled estimation and flocking control for a single target with limited bandwidth in which only the position of the target is shared among the sensors (Jin et al. 2017).

Two other approaches to the distributed Kalman filter are by Giannini et al. (2012) and Rigatos (2011). In Giannini et al. (2012), the distributed Kalman filters use the inverse of the trace of the covariance to decide which local estimate to propagate though the sensor network. The authors of Rigatos (2011) present a distributed version of both the extended and unscented Kalman filter (see Section 2.4.6 for more about unscented Kalman filters).

The distributed versions of the Kalman and extended Kalman filters suffer from the same weaknesses as their centralized counterparts. An additional challenge is to make the local estimates to converge. As mentioned above it is possible to set the gain controlling the convergence, C_i in equation (2.5), such that filters converge globally asymptotically as proved by Olfati-Saber (2009). Unfortunately, this assumes an accurate knowledge of the process noise Q, which in general will not be available as discussed in Section IV-A. In the papers applying the Kalman consensus filters there is little or no discussion of how to obtain the process noise Q.

2.4.4 Particle Filter

Another popular filter in target tracking applications for mobile sensor networks is the particle filter. Unlike the Kalman filter, the particle filter does not require the process and measurement noise to be Gaussian distributed. This comes at the cost of increased computational complexity.

We will present a general version of the particle filter for a single target which will be based on Arulampalam et al. (2002). Both Kalman and particles filters are a form of Baysian filtering, but, while the Kalman filter represents the probability density function of the target state as a normal distribution, the particle filter approximates it using M number of particles. This means that the particle filter can use any form of distribution. If we let each particle be written as $\{\hat{z}^i, w^i\}$, where $\hat{z}$ and w are the state and weight, respectively, with i representing the particle number. We write the posterior probability density function for the linear Kalman filter and the particle filter together for easy comparison

$$p_{\text{kalman}}(z|y_k) = \mathcal{N}(z; \hat{x}_{k+1|post}, P_{k+1|post}) \tag{2.6a}$$

$$p_{\text{particle}}(z|y_k) \approx \sum_i^M w_k^i \delta(z - \hat{z}_{k+1}^i) \tag{2.6b}$$

$$\text{where} \quad \sum_i^M w_k^i = 1$$

where $\delta(\cdot)$ is the Dirac delta function, and z is the argument of the probability density functions for the Kalman and particle filters, which are noted as $p_{\text{kalman}}(z)$ and $p_{\text{particle}}(z)$. The notation for a normal distribution $\mathcal{N}(z; x, P)$ has argument z, mean x, and covariance matrix P.

The linear Kalman filter is updated with equation (2.2). When using a particle filter, each particle is sampled from an importance density function, $q(z)$, at each iteration. Each particle's weight is updated by

$$w_k^i \propto \frac{p(x_k)}{q(x_k)} \tag{2.7}$$

where $p(x_k) \propto p_{\text{particle}}(z|y_k)$. The choice of importance density function is important for the performance of the algorithm. A good choice of importance density function is minimize the variance of the weights, w_k^i. However, this choice suffers from some drawbacks. For example, it requires the ability to sample from $p(x_k)$, see Section V-1 in Arulampalam et al. (2002) for more details.

A well-known problem with particle filters is the degeneracy problem. Eventually, all the particles will be very unlikely, which in practice means they have a very

small weight. This can be measured through the variance of the weights. A normal measure of degeneracy is to calculate

$$N_{\text{eff}} = \frac{1}{\sum_i^M (w_k^i)^2} \tag{2.8}$$

A commonly implemented way to counteract the degeneracy problem is to resample the particles. Often a threshold is used in relation to equation (2.8) to decide when to resample the particles. In practice, this often means discarding the less likely and multiplying the more likely particles.

The particle filter is used both in single- and multi-target tracking by mobile sensor networks. A centralized approach for single-target tracking is implemented in Li and Djuric (2007). Here, the authors combine the particle filter with a Cramér-Rao Lower Bound to deploy mobile senors and compare it to using stationary sensors. The same authors also formulate a particle filter that does not require an assumed probability distribution for the process noise (Li and Djuric 2008) for single-target tracking. This is based on the work by Míguez et al. (2004), which substitutes the probability density function with a user-defined cost function that measures the quality of the state signal estimates according to the available observations. In Hoffmann and Tomlin (2010), the authors also study single-target tracking. They develop a trajectory planning algorithm for minimizing the expected future uncertainty of the target state, in which they utilize the posterior probability available from the particle filter. The authors of Lu et al. (2014) create a modified version of the particle filter for multi-target tracking, implemented as centralized and combined gradient-based approach for motion plans and control inputs. They use two modifications on the particle filter. First, instead of approximating the post probability distribution with a weighted sum of Dirac delta functions, they use a weighted sum of normal distributions:

$$p_{\text{particle}}(z|y_k) \approx \sum_i^M w_k^i \mathcal{N}(z; \mu_i, \sigma_i^2) \tag{2.9}$$

$$\text{where} \quad \sum_i^M w_k^i = 1$$

Second, they incorporate the newest measurement into the importance density function. This is done through using a target-state likelihood function, instead of a target-state transition function, equation (2.3a). Another similar approach for modifying the particle filter with a modified importance density function and resampling method is Juan-Yi (2011). In Hu and Tu (2017), the authors use a combination of stationary and mobile sensors to track a single target. For target

estimation they use a particle filter, which they modify for low energy consumption through parallel processing and better anti-noise capability.

The particle filter overcomes some of the limitations of the Kalman filters, like the assumption of Gaussian process and measurement noise, in addition to a linear model (compared to a linear Kalman filter). A draw-back is that it does not have an obvious covariance equivalent, which can be used for designing a tracking algorithm. There are a few alternatives to quanity the uncertainty of a target estimates. In Li and Djuric (2007), the authors use a Cramér-Rao Lower Bound. Another option is to use the posterior probability directly like Hoffmann and Tomlin (2010). A third option is to construct a potential feedback function based on the particle filter for the controller Lu et al. (2014). Even though the particle filter can use any probability distribution, this distribution still has to be known. The authors of Li and Djuric (2008) suggest a particle filter where the process noise does not have to be known. However, this requires a user-defined risk function.

2.4.5 Distributed Particle Filter

As with the Kalman filter, it is sometimes necessary to a have a decentralized approach for the particle filter. One way to do this to organize the sensors in clusters and introduce some additional steps. After calculating the local posterior probability distribution, equation (2.6b), the weights, equation (2.7), are updated by the clusterhead (selected leader of the cluster of sensors) with

$$w_{k+1}^i = w_k^i \prod_{j \in \mathcal{N}_c} p(y_j | \hat{z}^i) \tag{2.10}$$

where w_{k+1}^i is the weight of particle i at time $k+1$. The set of neighbors to the clusterhead is denoted $\mathcal{N}_c$, which each have a measurement of the target y_j. The state estimate of particle i is denoted $\hat{z}^i$. Then, after normalization of the weights, the clusterheads resample and find the current estimate of the state. The result is then combined with the other clusterheads through diffusion. This is the approach used by Chen and Sezaki (2011) for single-target tracking using a stationary sensor network. In Gu and Hu (2011), the authors combine a flocking controller with a distributed particle filter for single-target tracking. Transporting all the particles between the nodes requires much bandwidth. Instead the authors use a Gaussian mixture model (GMM) learned from the weighted particles of all sensors through a distributed expectation maximization algorithm. This only requires that a few parameters be exchanged between the nodes. An expectation maximization (EM) is an iterative algorithm in two steps. First, a GMM is assumed and the likelihood for each particle is calculated based on the current GMM model. Second, the GMM model is updated based on the calculated particles. The first step is then repeated with the new GMM model. The algorithm stops when

the GMM model has converged sufficiently. The global GMM model is found through an average consensus filter similar to the one presented in equation (2.5) by Olfati-Saber and Murray (2004). The algorithm by Olfati-Saber and his collaborators has also inspired the authors of Kan et al. (2012) to handle single-target tracking using a mobile sensor network. They further develop the consensus algorithm for particle filters to also consider that some sensors do not observe the target. They demonstrate their algorithm using five sensors for a single target. An additional challenge is to combine distributed particle filters and multiple-target tracking. Particularly, this suffers from the curse of dimensionality, with the number of necessary particles getting multiplied by the number of targets. To meet this challenge, the authors of Beaudeau et al. (2015) assume a linear state model for the targets, equation (2.1a), which enables them to approximate the posterior distributions with normal distributions, while they maintain a nonlinear measurement model, equation (2.3b). This enables them to reduce the required communication between the sensors, since each target state can be approximated by a mean vector and covariance matrix. For each time step, each sensor generates particles from a normal distribution from each of the other sensors. The newest measurement is then used to calculate each particle's weight according to equation (2.7). From the new posterior distribution, a new mean and covariance matrix is calculated based on the weights and particle states, which is then broadcasted to the other sensors.

The distributed particle filter have similar problems to distributed Kalman filters. In addition, a challenge with the particle filters is the exchange of estimates. Since each sensor approximates a probability distribution with a set number of particles, all particles must be exchanged between sensors to accurately exchange estimates. In comparison, a Kalman filter can be exchanged through one state estimate with a covariance matrix. It is natural to approximate the estimate of a particle filter like Beaudeau et al. (2015). However, in that paper the authors approximate the estimate with a Gaussian distribution, which partly make applying the particle superfluous since assuming a Gaussian distribution make the Kalman filter applicable.

2.4.6 Other Observers

In addition to Kalman and particle filters, there have been multiple other filters applied by authors using mobile sensor networks for target tracking. In this section, we will briefly present some of them.

An unscented Kalman filter is an attempt to improve the extend Kalman filter for non-linear systems. A problem with the extended Kalman filter is that the covariance matrix is propagated through a linearization, equation (2.4b) (Wan and Van Der Merwe 2000). This is can lead to large estimation errors, especially for highly

non-linear systems. The unscented Kalman filter use a set of sampling points to capture the true mean and covariance of the state estimate. This achieves improved accuracy over the extended Kalman filter, without an increase in computational complexity. Wang et al. (2010a) and (Xie et al. 2016) present a distributed version of the unscented Kalman filter for target tracking by a mobile sensor network. In addition, Wang et al. (2010b) present a distributed filter based on a more general unscented filter algorithm. Even though the unscented Kalman filter have a same order of magnitude as the extended Kalman filter when it comes to complexity, there is an increased complexity for the human implementing it.

Another estimation approach similar to Kalman filtering is H_∞. Here, instead of minimizing the $\mathcal{L}_2$-norm as done in the Kalman filter, the $\mathcal{L}_\infty$-norm is minimized (Shaked and Theodor 1992). In Nelson and Freeman (2009a), the authors present a distributed version of the H_∞ estimator to track a single target using an MSN. A disadvantage of the H_∞ filter, compared to the Kalman filter, is that it has more user set parameters. The same authors have also suggested a set-value estimation algorithm (Nelson and Freeman 2009b). In contrast to the other filters, this estimator makes no assumption on the process and measurement noise except for a maximum value. Another advantage is a distributed version of the filter. It is straight-forward to use a union of sets from neighboring sensors without taking into consideration that a set should not be added multiple times (a union of a set with itself does not change the set). A problem with a set-value estimator is the increased storage capacity required for each iteration. To overcome this, the authors approximate the current set estimate with ellipsoids or parallelotopes. Another disadvantage is that since there is little assumption on the uncertainty of the estimates, there is not a covariance equivalent.

The federation filter is a distributed version of the Kalman filter designed to be computed in parallel (Carlson 1990) by local filters and fused by a master filter. In Jin et al. (2017), each sensor acts as a master filter, with neighbors as local filters, in a single-target tracking application. This is similar to Olfati-Saber (2009) and have the same problem of needing an accurate estimation of the process noise, Q.

In the case of a linear system with no disturbance, the Luenberger observer is well-suited. This works best for theoretical applications such as Wang et al. (2016), in which a feedback controller for single-target tracking utilizes a Luenberger observer. However, it should not be applied in practice since it does not take into account measurement or process noise.

The authors of La and Sheng (2011a) present a consensus algorithm for estimating a scalar field used for navigation by a flocking algorithm. Each sensor keeps a local version of the scalar field, and the authors present an approach on how to update

the field. To estimate the target's position the probability distribution is assumed known.

2.4.7 Multi-Target Tracking - Data Association

A problem which is often ignored in multiple-target tracking for MSNs, is *to match measurements to filter*. This is often referred to as the *data association problem*, but it can also be called the *multiple-target tracking problem* or *multiple sensor (data) fusion*. To avoid addressing this problem, the targets can either be assumed to be sufficiently far from each other, such that each observation can be matched with its closest filter, or that each target has a special characteristic making it trivial to match it to the correct filter. However, in general, this is not a trivial problem, and it becomes especially prevalent when we do not have continuous observations of each target.

Data association is a well-studied subject, and our goal here is not to give a comprehensive survey of the literature. Rather, it is to give a brief introduction through some surveys dealing with the problem of target tracking. The remainder of this section is based on the following surveys: Blackman (1988), Pulford (2005), Mallick et al. (2012), and Qiu et al. (2015).

Classical data association can be divided into recursive and batch approaches. A recursive approach has irreversible state updates and is typically less computationally expensive. The perhaps most straightforward approach is the nearest-neighbor algorithm. Here measurements are paired with the closest estimate, and it works well when the targets are separated in space and process noise is small. Closeness is typically measured in Euclidean distance, but the signal-to-noise ratio can also be used in cases with much clutter (false-positive measurements). An alternative approach is the all-neighbor algorithm, in which all measurements in the vicinity are used to update the target estimate. This is done by calculating the probability for each measurement and using it in a weighted average to update the target estimate. This approach is also referred to as (joint) probabilistic data association. Another algorithm is the global nearest neighbors. This is a minimal pairing with a set of measurements for a set of target estimates. It can be solved efficiently by the Hungarian algorithm (Kuhn 1955).

In a batch approach, the decision of connecting measurement to target estimates is postponed until more measurements are available. This comes at an increased cost of storage and computational complexity. Examples of algorithms are the Viterbi algorithm, which finds the most likely sequence of hidden states given a sequence of measurements (Viterbi 2010). Another is the expectation maximum algorithm described in Section 2.4.5. Other batch approaches can be classified as

Filter	Assumptions	Drawbacks	Advantages
Kalman Filter	Linear model, Gaussian noise	Restricting assumptions	Low computational complexity
Extended Kalman Filter	Nonlinear model, Gaussian noise	Can become unstable	Low computational complexity
Particle Filter	Nonlinear model	High computational complexity	No Gaussian assumption

Table 2.3: Summary of methods for filtering measurements used by observers.

multiple hypothesis tracking, where a measurement can be temporarily associated with multiple target estimates, and it is not until later that it is associated with a single-target estimate. Examples of these methods are integer programming, Lagrangian relaxation, approximate linear programming, and Markov Chain Monte Carlo (MCMC)-based data association. In addition, particle filters described in section 2.4.4 have also been used for solving data association problems.

2.4.8 Summary

Table 2.3 summarizes the well-known properties of the three most common types of filters used by observers in MSN settings for target tracking.

2.5 Trajectory Planning

In this section, we will discuss high-level control algorithms for the trajectory planning of the sensors in MSNs with an application of target searching and tracking. We divide the algorithms into three different categories. First, we present explicit control algorithms. In an explicit control algorithm, the actuator input can be calculated directly from sensor measurement, state estimates, and so on. This include gradient-descent type algorithms, as well as classical algorithms, such as the proportional-integral-derivative (PID) algorithm. Second, we examine optimization techniques. In these algorithms, the actuator is often the decision variable in an optimization problem, and it can, for example, be implemented in a reced-

ing horizon fashion. Other decision variables can be, for example, the location of each sensor. This typically lead to an implicit formulation for the actuator input. Finally, we explore a category called heuristics. Here, each sensor operates in a rule-based fashion. It is similar to explicit control, but has more of an if-else sentence structure. All types of algorithms can be implemented both in a distributed and centralized fashion. However, the trend is that explicit and heuristic algorithms often are distributed, while optimization strategies typically are centralized.

2.5.1 Explicit Control

In explicit control, the actuator input can be calculated as a function of the sensor states, state of other sensors, and the estimated state of the target(s), and, sometimes, the state of the environment. Let the state of sensor i be z_i and the actuator input be u_i. Furthermore, let $\boldsymbol{z} = [z_1, z_2, \ldots, z_n]$ denote the state of all sensors and $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$ denote the state of all targets. Notice here that we use the notation for the actual state x and not the estimate state $\hat{x}$. In practice, we will always have to use an estimated state in the control law, but, for simplicity, we will use the notation for the actual state through this section and the remaining part of this chapter. We can write the general sensor dynamic equation and feedback control law as:

$$\dot{z}_i = f(z_i, u_i) \tag{2.11a}$$

$$u_i = g(\boldsymbol{z}, \boldsymbol{x}) \tag{2.11b}$$

In the remainder of this section we will discuss different types of explicit control used by mobile sensor networks. First is a gradient-based algorithm. In these algorithms a potential function is constructed, and the derivative is used by the trajectory planner. The potential function can be based on distance to neighboring sensors, probability of finding a target, and so forth. Second is flocking control. Here, the goal is to get the sensors to behave as a unit with a distributed control algorithm. It can be thought of as a special case of gradient-based control. Third, we will discuss P-controllers and other controllers applied by the sensors in MSNs.

Gradient-based control

Gradient-based control is well-suited for distributed approaches for MSNs. They usually work by utilizing local information about other sensors, obstacles, and targets to construct a potential, or cost, function. Then, the gradient of the potential function is calculated and used as an input to the actuator function. The sensor decreases the potential function by moving in the direction of the gradient. This

can be written as

$$U(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{o}) = w_s u_s(\boldsymbol{z}) + w_o u_o(z_i, \boldsymbol{o}) + w_n u_n(z_i, \boldsymbol{x}) \tag{2.12a}$$

$$u_i = \nabla_{z_i} U(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{o}) \tag{2.12b}$$

where $U(\boldsymbol{z}, \boldsymbol{x})$ is the potential function based on the state of all sensors $\boldsymbol{z}$, the target state $\boldsymbol{x}$, and obstacles $\boldsymbol{o}$. The three terms $u_s()$, $u_o()$, and $u_n()$ correspond to collision avoidance and connectivity to other sensors, obstacle avoidance, and navigation, respectively. Connectivity is to keep sensors in communication range with each other. The constants w_s, w_o and w_n are tunable constants to weight the different objectives against each other. We use ∇_{z_i} to denote the gradient with respect to the state of sensor i. Usually, the gradient is found with respect to the position of the sensor.

There are some challenges with a gradient-based approach. The potential function should be convex to move the sensor towards a global optimum. The sensor risks getting stuck in a local minimum with a non-convex potential function. Another challenge is to weight different objectives. A gradient function usually has multiple tunable constants for weight collision and obstacle avoidance, target tracking, and connectivity. It is also often necessary to have a constant deciding how fast the sensor should move towards reducing the potential function.

The typical collision and connectivity potential function is based on distance between a sensor and its neighboring sensors. If we let p_i denote the position of sensor i, we can write

$$u_s(\boldsymbol{z}) = \sum_{j \in \mathcal{N}_i} \varphi(||p_i - p_j||) \tag{2.13}$$

where $\mathcal{N}_i$ denotes the neighborhood of sensor i, and $\varphi()$ is a function designed to either repel sensors when they get too close or attract them when they are too far away. Often it does both. A typical choice for $\varphi()$ is an inverse proportional function, such as, for example, Giannini et al. (2012), Hu et al. (2012), Ma et al. (2008), Zhao et al. (2014a) and Yang et al. (2008). Another option is to use an exponential function, as used by (Ferrari et al. 2011, Rigatos 2011). An interesting choice by the authors of Li et al. (2007) is to design $\varphi()$ such that it is zero in the interval $||p_i - p_j|| \in [\delta_1, R_c\delta_2]$, where δ_1 and δ_2 are small constants, and R_c is the communication range of the sensors.

Similarly to the sensor function, an obstacle function can be constructed as

$$u_o(z_i, \boldsymbol{o}) = \sum_{j \in \mathcal{N}_i} \varphi(||p_i - o_j||) \tag{2.14}$$

where $\mathcal{N}_i$ denotes the obstacles in the neighborhood of sensor i. This time $\varphi()$ is designed solely to repel the sensors from the obstacles.

To design the navigation term, the following properties are often utilized: The probability distribution of a target obtained from an observer (see Section 2.4), probability of detection, and distance to the target. It is therefore hard to write a more specific equation for u_n than the one written in (2.12a). Instead, we will discuss some of the approaches taken in the literature.

In single-target tracking applications, there are multiple ways of designing the navigation term. In Rigatos (2011) and Zhao et al. (2014a), the authors take into consideration the heading of the sensors. The goal of Zhao et al. (2014b) is to get the sensors on an ellipse surrounding the target within a certain distance of each other. In Rigatos (2011), a stochastic variable is added to the gradient algorithm to avoid local minimums. A simpler approach is to use a threshold and constant attractive force for the target on the sensors, such as that used by Ma et al. (2008). Both Chattopadhyay et al. (2015) and Giannini et al. (2012) use the distance to the target when designing the navigation term. Finally, the authors of Gusrialdi et al. (2008) and Li et al. (2007) apply probability calculations to design the navigation term. In Gusrialdi et al. (2008), the authors calculate the expected probability of finding the target, while Li et al. (2007) use the probability that the target is in a region. The author also uses a gradient-based approach that utilizes multiple objectives: maximize connectivity, minimize movement, and minimize target escape probability (Li et al. 2008). This approach is later expanded to handle noisy measurements in Li and Liu (2009).

Another approach to dealing with single-target searching is that of Hutchinson and Bretl (2012). Here the authors use the probability of missed detection to design the navigation term. The author Nelson has two papers dealing with evasive targets, in which he uses a gradient approach. In Nelson and Freeman (2009b). he uses the anticipated measurement to design the navigation term, while in Nelson and Freeman (2009a) the authors use the trace of the covariance matrix from a H_∞ filter (see Section 2.4.6).

Hu et al. (2012), Jha et al. (2016), Rout and Roy (2016), and Yang et al. (2008) deal with multiple-target tracking. They all apply distance to target to the design of the navigation term. In Mathew et al. (2010), the authors use a probability distribution for the targets to calculate a gradient-based algorithm.

Finally, we will discuss two papers that use a similar approach to gradient-based control without first constructing a potential function. First, in Sun et al. (2014), the authors work on multiple target searching and tracking. They use two modes,

search and track, each with a separate control law. In the search mode, the control law consists of multiple terms: a term for probability of detecting a target, another term for collision avoidance, a third to keep the sensor within the search area, and, finally, a momentum term to make it difficult for the UAV to change direction rapidly. Second, Yanmaz and Guclu (2010) calculate forces based on distance to other sensors and move the sensor away from the others to achieve full coverage of an area when performing target searching.

In the papers referenced above, the authors must often make many assumptions and set multiple constants for the algorithm to perform well. Usually, the algorithm is demonstrated in a simulation, where the authors can select parameters such that their algorithm performs well and fulfills their assumptions. However, multiple assumptions and user set constants will likely make these algorithms difficult to apply in practice. A problem with designing the navigation term based only on the position of the target(s) is that the algorithm must know the location of the target to find it. This make the application of these algorithms limited when it comes to searching for unknown targets.

Flocking Control

Flocking control can be considered a version of gradient-based control, but we have chosen to discuss it in its own section as it usually deals with a specific problem, namely single-target tracking with a distributed control algorithm. It is inspired by biological systems of birds, fish, and insects.

The behaviors that lead to simulated flocking were first stated by Reynolds (1987) (in decreasing precedence):

1. Collision Avoidance: avoiding collisions with nearby flockmates;
2. Velocity matching: attempting to match velocity with nearby flockmates; and
3. Flock Centering: attempting to stay close to nearby flockmates.

As pointed out by Olfati-Saber (2006), these rules are often referred to as cohesion, separation, and alignment rules in the literature. Olfati-Saber also points out that these rules have a broad interpretation and that it is not trivial to implement them. Additionally, he proves that an algorithm following the above rules does not necessarily lead to uniform flocking behavior. For example, the phenomena where multiple separate flocks are formed, which is called fragmentation, can occur.

Each sensor in a flocking algorithm is typically modeled as a second-order integ-

rator

$$\dot{q}_i = p_i \tag{2.15a}$$

$$\dot{p}_i = u_i \tag{2.15b}$$

where q and p are the position and velocity of sensor number i. Normally, they are of dimension two. The actuator is u_i and typically consists of three terms:

$$u_i = f_i^g + f_i^d + f_i^\gamma \tag{2.16}$$

where f_i^g controls the distance between the sensors, f_i^d is a damping term that ensures velocity matching for the sensors, while f_i^γ is navigational feedback based on the flock's objective.

In the paper (Olfati-Saber 2006), Olfati-Saber introduces three different flocking algorithms, two for free-space and one with obstacle avoidance. It can be viewed as a tutorial paper for flocking algorithms and is one of the most cited papers within the field and it is also commonly used for algorithm comparison. In later work, the same author has applied his algorithm to single-target tracking (Olfati-Saber 2007, Olfati-Saber and Jalalkamali 2012). He also concludes that the practical need for collision avoidance, combined with a moving rendezvous, leads to an emergence of flocking behavior even without explicit communication between sensors. In terms of cooperation, Olfati-Saber has also applied the flocking algorithm to a coverage problem for multi-target tracking (Jalalkamali and Olfati-Saber 2012).

The work of Olfati-Saber and his collaborators has been extended by La and Sheng in multiple aspects. First, in single-target tracking, the actuator input for the sensors, equation (2.16), consists of multiple tunable gains for each term. To select these gains, they introduce the following objective function in La et al. (2009)

$$F = \frac{\sum_i \int_0^T ||q_i(t) - q_t(t)||dt}{T \sum_i ||q_i(t=0) - q_t(t=0)||} \tag{2.17}$$

here q_i is the position of each sensor, with q_t is the position of the target, and T is the simulation time. Minimizing this objective function corresponds to minimizing the time and distance for the sensor network to catch up to the target. This is a non-convex and non-differentiable objective function. The authors apply a genetic algorithm to solve the optimization problem. A genetic algorithm is inspired by Charles Darwin's theory of natural selection. A set of solutions are generated randomly, and the objective function is used for evaluation. Then, the best solutions are merged together, which is called matching. The new solutions are called offspring. Often some of the offspring are also exposed to random mutation, which are small random changes. Solutions that perform poorly are eliminated. The

above steps are performed iteratively until a satisfying solution is produced. In La et al. (2009), the authors use a Gaussian distribution to generate solutions. The solutions that do not satisfy certain constraints are eliminated. Matching is performed, but they do not use mutations. Another solution to the objective function (2.17) is that of Khodayari et al. (2016). They suggest their own algorithm, which they call the gravitational algorithm. This is inspired by Newton's law of gravity.

La and Sheng also extend the work in obstacle management. In Olfati-Saber (2006), the sensors split when they meet obstacles, which can lead the network becoming disconnected. La and Sheng (2009a) introduce what we will cal the squeezing algorithm. This works by manipulating the required distance between sensors when passing obstacles. Instead of going around an obstacle on separate sides, the sensors "squeeze" together such that all can pass on the same side. Another extension of Olfati-Saber by La and Sheng is the center of mass (CoM) algorithm. If we consider a 2D view of an MSN and think of it as a web, the algorithm from Olfati-Saber (2007) has no guarantee that the target will be at the center of the web during the tracking. In La and Sheng (2009b), the authors introduce an algorithm in which each sensor estimates the center of the MSN. This enables the sensor network to act a single unit when tracking a target. In later work, the authors combine the CoM algorithm with a distributed filter for tracking (La and Sheng 2011a). The CoM algorithm is also extended to handle noisy measurements of each sensor's position and velocity (La and Sheng 2011b). A third extension of Olfati-Saber (2006) by La and Sheng is in multiple-target tracking (La and Sheng 2009c). Here, they split a flock into two when encountering a new target. To split the sensors, they use a seeding algorithm. It is initiated by the sensor closest to the new target, which messages it closest neighbors to follow the new target. This continues until the number of sensors following the new target reach a predetermined number. Merging happens when a target disappears, in the same way as a flock is formed. In La and Sheng (2012), the multiple-target tracking algorithm is combined with the "squeezing" algorithm from La and Sheng (2009a).

In recent years, there have been several other aspects using flocking algorithms for single-target tracking by an MSN. Gu and Hu (2010) study tracking an evasive target. They use a distributed minmax filter, which tries to maximize the worst case of the tracking performance. In Gu and Hu (2011), the same authors combine flocking control with a distributed particle filter. They apply the approximation of the posterior distribution from the particle filter in the flocking control algorithm such that all the sensors are driven towards the target. Tu, Wang and their collaborators designed an algorithm for deployment around a slow-moving target, which considers the heading of the sensors (Tu et al. 2012a). The flocking algorithm is

demonstrated in experiments in Wang et al. (2012). The authors also have an algorithm for deployment which requires only 1-hop neighborhood communication between sensors (Tu et al. 2012b). This algorithm also avoids "holes" in the network. A hole in an MSN can be a small area which no sensor is covering, thus making it possible for a target to hide within the network. Another approach that also focuses on avoiding holes is Zhang and Zhu (2015). Most papers concerning flocking consider the control and target state estimators separately. In Jin et al. (2017), the authors combine a flocking algorithm similar to Olfati-Saber with a distributed Kalman filter. They analyze the coupled controller and observer using Lyapunov theory to prove global asymptotic stability. An additional application is coupled-target tracking studied by Su et al. (2017). Here, two targets with coupled behavior are tracked by splitting the sensor network into two. Each part has a sensor acting as leader with the ability to broadcast messages to the other sensors, which only have local communication abilities. By taking advantage of the coupled behavior, the authors manage to get increased performance compared to treating the two targets as separate tasks. The authors of Dang and Horn (2015) study a centralized approach to flocking for multi-target tracking. Here, the authors use a splitting algorithm to divide a flock when a new target is encountered. In addition to the terms in the actuator equation (2.16), the authors use a merging term for the free-sensors that do not belong to a flock. The same authors also work on single-target tracking in a noisy environment (Dang et al. 2016). Here, they develop a distributed algorithm, where one sensor is selected to be the leader and follow the target, while the others avoid collision and follow the leader. The leader is chosen based on the distance to the target. The sensor network uses a V-shape when chasing a target and assumes a circular shape around the target when it is caught. In Jiang et al. (2013), the authors study single-target tracking by combining flocking control and a distributed filter. The authors use only a potential function for distance between sensors to avoid collision and have no velocity matching. Instead, they use a proportional controller (P-controller) to match each sensor to the position, velocity, and acceleration of the target.

For the problem of searching and tracking multiple targets, flocking control has limited applicability. The most common situation for a flocking algorithm is to have many sensors and track a single target. For example, in a searching application, it is intuitively not desirable (depending on sensor characteristics) to have the sensors flock together, but rather spread out to cover as much ground as possible. Often cost is an important factor, which makes it questionable to use multiple sensors if one is sufficient.

P-controller

In this section, we present control strategies which can be classified as, or similar to, a P-controller. A P-controller uses a desired state as input. Then, it calculates the difference between the actual and desired states and applies an actuator signal proportional to the difference. This is typically written as

$$u_i = k_p(z_{i,\text{desired}} - z_i) \tag{2.18}$$

where k_p is a tunable gain with $z_{i,\text{desired}}$ and z_i denoting the desired and actual state of sensor i, respectively. This type of controller is often used in cooperation with a high-level control, which is, for example, used to find the desired state.

In Xie et al. (2016), the authors formulate an optimization problem to obtain the optimal distance and heading towards a target. Then, they use a P-controller to move each sensor towards its optimal distance and heading. The authors of Tan et al. (2004) calculate a Voronoi diagram (a way of partitioning the plane into a number of equal regions) and use a P-controller to move each sensor towards the center of its cell. A more complex control law is:

$$u_i = -k_i h'(C^* - \mathcal{T}_\mathcal{N}), \tag{2.19}$$

which is applied by Wang et al. (2016) to track a single target. Here k_i and h' are a constant and a penalty function, respectively. The desired coverage of the target is C^* while the actual coverage by the mobile sensor network is $\mathcal{T}_\mathcal{N}$. The authors prove asymptotic stability of the controller using the Lyapunov theory. In Kuo et al. (2017), the authors form a control law to trap a target within a polygon formed by the mobile sensors. To move the polygon, each sensor continuously computes its next waypoint and move towards it using a P-controller.

A P-controller is, in general, not recommended for high-level trajectory algorithms. It has limited applicability since it is difficult to incorporate a higher level objectives. In the above examples, it is only used to move the sensors straight for a desired position.

Other explicit controllers

In this last section on explicit control, we present a paper that does fit into the above categories, but which can be considered to cover an explicit controller.

In Xu et al. (2010), the authors design a control law for multiple-target tracking. They assume that the targets move in an acceleration field and use this to design a control law similar to backstepping. Backstepping is a technique for recursively designed stabilizing control for each subsystem. A disadvantage with backstepping is that it requires an accurate model.

2.5.2 Optimization

We define optimization control to minimize (or maximize) any objective function using actuator input or sensor location as manipulative variables. This lead to an implicit formulation for the actuator input. The actuator input can, for example, be applied to each sensor in a finite horizon fashion or as an assignment. A general optimization control problem can be written as

$$\min_{\boldsymbol{u}} f(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{u}) \tag{2.20a}$$

s.t.

$$g_1(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{u}) \geq 0 \tag{2.20b}$$

$$g_2(\boldsymbol{z}, \boldsymbol{x}, \boldsymbol{u}) = 0 \tag{2.20c}$$

where $\boldsymbol{z}$, $\boldsymbol{x}$, $\boldsymbol{u}$ are the aggregated sensor states, target states, and actuator inputs, respectively. The objective function is $f()$ with inequality and equality constraints denoted $g_1()$ and $g_2()$, respectively. First, we will discuss task allocation algorithms. These are typically applied in a centralized fashion and set the new waypoint for each sensor. Then, we will go into optimal control. Here the actuator of the sensors is used as a decision variable, and the optimization is continuous. In the last part of this section, we will discuss various other optimization strategies.

Task allocation

A typical task allocation deals with either single- or multiple-target tracking. There are usually two steps involved. First, a set of new positions is obtained for the sensors based on, for example, target positions, expected measurements, and so on. Second, a combinatorial formulation is used to set up an assignment problem, which, in its most basic form, can be solved by the Hungarian algorithm. A centralized formulation for the assignment problem is

$$\min_{a} \sum_{i \in \mathcal{P}_{\text{current}}} \sum_{j \in \mathcal{P}_{\text{new}}} C(i,j) a_{ij} \tag{2.21a}$$

$s.t.$

$$\sum_{i \in \mathcal{P}_{\text{current}}} a_{ij} = 1 \quad \forall j \in \mathcal{P}_{\text{new}} \tag{2.21b}$$

$$\sum_{j \in \mathcal{P}_{\text{new}}} a_{ij} = 1 \quad \forall i \in \mathcal{P}_{\text{current}} \tag{2.21c}$$

where the sets of current and new sensor positions are $\mathcal{P}_{\text{current}}$ and $\mathcal{P}_{\text{new}}$, respectively. Note that these do not have to be of equal size, as some sensors may not have to move. Each move is associated with a cost stored in the matrix $C(i,j)$.

Here we use a binary matrix (values 0 or 1) with entries denoted a_{ij}, which is 1 if the sensor at position i should move to position j and 0 otherwise.

Usually, the objective is to minimize energy consumption of the network. Normally, this is done by minimizing the movement of the sensors by letting the cost, $C(i, j)$, in equation (2.21) represent distance. This is a different problem than a typical gradient-based control problem, where the sensors are expected to be continuously moving. An advantage over gradient-based approaches is that task allocation typically leads to an optimal solution even with non-convex problems. The assignment problem can be solved in polynomial time with the Hungarian algorithm (Kuhn 1955). Unfortunately, many of the target tracking formulations for MSNs also include non-linear constraints, making the problem NP-hard, with no known solution within polynomial time. Another challenge is to implement assignment problems in a distributed fashion.

There are multiple papers that deal with single tracking, which uses an objective function in which the cost matrix consists of distances between current and new sensor positions. In Bai et al. (2012), the new sensor positions are found by minimizing target uncertainty. The assignment problem, equation (2.21), is solved using the Hungarian method (Kuhn 1955). The authors of Qi et al. (2016) modify the constraint (2.21c) to only require that each sensor visits at most a new position (they switch out the equality for inequality constraints). In addition, they add a non-linear constraint that requires the probability of detection to be above a given threshold. This is also the criteria they use to find new positions. To solve the problem, they divide it into two sub-problems, one of which can be solved online through the Hungarian algorithm. Instead of reducing the minimum distance, the sensors should travel. The authors of Li and Liu (2007) minimize the velocity of each sensor

$$\min \sum_{i \in \mathcal{S}} ||v_i|| \tag{2.22}$$

where $\mathcal{S}$ is the set of all sensors, and the velocity of sensor i is v_i. The constraints they use are related to tracking performance of the target and connectivity (communication between nodes). The authors of Mahboubi et al. (2010) assume that the movement of the sensors is negligible compared to communication with respect to energy consumption. Energy consumption is considered proportional to the distance between the sensors. The authors suggest an algorithm to optimize the movement of the sensors to minimize the communication distance between them while tracking a target. In another paper by the same author (Mahboubi et al. 2012), he converts an MSN to a graph and finds a weight for each edge corresponding to a communication cost. A shortest path algorithm is then used to find the

path which minimizes the energy necessary to message the location of the target to the base station. Similar approaches are applied in Mahboubi et al. (2011; 2016; 2017). A distributed approach is presented in Zou and Chakrabarty (2007). The algorithm works in two steps. First, each sensor applies a Bayesian filter to decide a new position to move to in order to improve sensing of the target. At the same time, the cost associated with the movement is calculated. This is communicated back to a base station, as well as neighboring sensors, before a decision is made as to which sensors should move. Another centralized approach is Mourad et al. (2012), where the authors use both stationary and mobile sensors. The stationary sensors are used to cover the area, while the mobile sensors are used to track a target. The tracking is done by estimating new positions for the sensors based on the target model and then setting up an assignment problem to minimize the moving distance for the mobile sensors, which is solved by ant colony optimization.

Multiple-target tracking by MSNs can also be solved by formulations similar to the assignment problem using distance as the cost function in equation (2.21). In Liao et al. (2015), the authors use Voronoi diagrams to find the new positions of sensors before applying the Hungarian algorithm to solve the assignment problem. The authors of Sharma et al. (2015) estimate the next position of each target and use the ant colony optimization technique to solve the assignment problem. Another similar approach is by Selvaraj and Balaji (2013). Here, the authors find the best positions for the sensor to cover the area before applying particle swarm optimization to assign sensors to those positions. A set of Kalman filters are used for the target's estimates in Fu and Yang (2014). They then use the inverse of the covariance matrix, along with the Cramér Rao Lower Bond (Tichavsky et al. 1998), as an objective function. In addition, they minimize energy consumption and use a constraint to maintain connectivity. They prove that the resulting optimization problem is NP-hard, and they develop an approximate algorithm that runs in polynomial time. In Gao et al. (2017), the authors formulate an assignment problem, which minimizes sensor movement, while requiring coverage of the monitored area. The authors of Kamath et al. (2007) study the problem of assigning two sensors to track each of multiple targets. They prove that the problem is NP-hard, and suggest an approximate algorithm to solve it in polynomial time. A distributed approach is introduced by Zorbas and Razafindralambo (2013) in which the goal is to maximize network lifetime. The algorithm works in two steps. First, connectivity is ensured, and a leader is selected. Second, minimum movement with required coverage of the targets is used to select the movement of sensors.

Besides minimizing movement distance, there are some papers that try to handle energy consumption more directly for single-target tracking. In Marbukh et al. (2010), the authors model battery consumption of the sensors and use an optim-

ization technique called simulated annealing to trade-off sensor movement versus battery consumption. The authors of Liu et al. (2007) allow sensors to be sleeping or listening, in addition to measuring and moving and formulate an integer optimization problem to maximize the lifetime of the MSN

In this paragraph, we present a few approaches that use a task allocation algorithm, but not to minimize energy consumption of the MSN. Low and his collaborators study target searching and tracking inside an indoor space using an MSN (Low et al. 2004a; 2006; 2004b). They use an ant colony optimization to dynamically allocate sensors to different areas, depending on the number of targets in each area. The author of Hung (2014) uses a distributed approach to allocate sensors to multiple stationary targets. He assumes that each sensor knows the number of sensors required to monitor each target and the location of the targets, but not the location of the other sensors. To discover this, each sensor needs to visit a target to see if it is monitored by enough sensors. The strategy is based on each sensor weighing the benefits of visiting targets to decide its visiting sequence. Another allocation problem is studied by Chang et al. (2015). Here, the problem is to patrol a given number of stationary targets. First, the number of sensors is decided and then multiple patrolling paths for the sensors are constructed. The authors develop a task allocation algorithm for matching sensors to patrol paths.

The assignment problem usually assumes that the positions of the targets are known, or at least have a solid estimate. In other words, these types of formulations are poor at searching for targets. Another, challenge is that the assignment problem, equation (2.21), is static. Searching and tracking problems are often highly dynamic, and solving an optimal problem for a certain time instance, might lead to non-optimal behavior on longer horizons. Finally, some of the assumptions applied to get the optimization problem on the form of equation (2.21) might make the solution invalid to the original problem. For example, some sensors might have nonholonomic moving constraints, which are often disregarded in optimization.

Optimal Control

Optimal control problems seek to find the optimal actuator input given some objective function along with state dynamics and other constraints. A general optimal

control problem can be written as

$$\min_{u(\cdot)} \int_{t_0}^{t_f} L(\xi, u)dt \tag{2.23a}$$

s.t.

$$\dot{\xi} = f(\xi, u) \tag{2.23b}$$

$$\xi_{\min} \leq \xi \leq \xi_{\max} \tag{2.23c}$$

$$u_{\min} \leq u \leq u_{\max} \tag{2.23d}$$

where $L(\xi, u)$ is the cost function with state ξ and actuator input u. The state dynamics are given by the differential function $f(\xi, u)$, with limits on the state and actuator given by equations (2.23c) and (2.23d), respectively.

An optimal control problem is typically converted to a large non-linear problem (NLP), which is a set of algebraic equations. Techniques, such as collocation, can be applied for approximating the objective function. Usually an interior-point or active set technique is applied to solve the NLP problem. See Betts (2010) and Biegler (2010) for details.

In target searching and tracking, an optimal control strategy enables the sensor dynamics to be taken into consideration, as well as more complex objective functions. In Wei and Ferrari (2015), the authors study single-target searching. They use an objective function which is a joint probability function of the position, heading, and velocity of the target. Sensor dynamics are included in the constraint (2.23b), and the actuator is limited. To solve the optimal control problem, the authors apply an approximation method called a variational iteration method. Optimal control techniques have also been applied to multiple-target tracking. In Haugen and Imsland (2016), the authors use UAVs to track icebergs. They apply Kalman filters to the tracking and use the trace covariance matrices in the objective function, along with the actuator of the UAVs. The dynamics of both the sensors (UAVs) and targets (icebergs) are included in the formulation, and a constraint is used to handle collision avoidance. To solve the optimal control problem, they use collocation to transform it into an NLP, which is solved with an interior-point solver. Another optimal control formulation is presented by Baumgartner et al. (2009) for multiple-target tracking using underwater vehicles as sensors. The objective function tries to maximize coverage along with minimizing energy consumption of the sensors. The dynamics of the sensors are included along with environmental information such as current as constraints. To solve the optimal control problem, the authors use a direct shooting method.

Optimal control problems are, in general, highly non-convex. This means that the solution we find will depend on the initialization of the problem. In the above

papers, there is often a lack of discussion of how to initialize these problems even though this can be critical to the resulting solution. Another problem is the computational complexity. In a highly dynamic problem, solving equation (2.23) might take too long, when the solution is ready, the original problem is no longer valid. Finally, there might be several user set parameters to make equation (2.23) possible to solve numerically. This often requires expert knowledge to apply. For example, the horizon can be difficult to select, since different time horizons lead to different solutions and selecting the most desired is difficult. In addition, long time horizons might be infeasible to compute.

Other Optimization Controls

There are a few papers which do not fit directly into the above categories. We present them here. First, Cheng et al. (2012) studies single-target tracking and formulates an optimization problem, which tries to maximize the sensing quality of the target while maintaining coverage of the search area. The controller is implemented in a receding horizon fashion. This means that the optimization problem is solved and only the first sequence of the solution is used. Then, the optimization problem is solved again. This approach is typically applied to dynamic problems, where new information frequently becomes readily available. The authors of Park and Hutchinson (2013) study a special formulation for single-target tracking, in which some of the sensors are expected to fail or even send erroneous messages. To solve this problem, the authors models it as an adversarial task, where some of the sensors are trying to sabotage detection of the target. The authors apply multistage decision to model the problem and solve it using dynamic programming over a receding horizon.

As with optimal control, it is challenging to select an appropriate time horizon for receding horizon problems.

2.5.3 Heuristic Control

We mean rule-based control when we discuss heuristic control. This can, for example, take a structure such as if-else sentences or be a grid map with rules for what actions a sensor should take in each cell. Typical tools are partially observable Markov decision processes or a dynamic Bayesian network to model the problem, and strategies from the field of artificial intelligence are often applied.

A popular problem often solved using techniques from artificial intelligence is evasive target searching or tracking. A simple strategy is applied by Chen et al. (2012) to solve this problem. Here, the authors use a grid map to model the probability of capturing a target in each cell. The sensors then move to the adjacent cell with the highest probability of target capture. The authors analyze how many

sensors are necessary to apply to a given area to have a given probability of capturing a target within a given deadline. The work is later expanded in Hsu et al. (2013). A similar approach is taken by Imai and Ushio (2013) to tracking multiple non-evasive targets. Another approach for an evasive single target is presented by Chin et al. (2010). They study the problem both from the pursuer (sensor) and target's perspectives. They utilize game theory to design strategies. In addition, a communication protocol is developed for the sensors to cooperate. The authors of Ferrari et al. (2009) study multiple evasive target capture. The area is modeled as a grid, which is decomposed into a connectivity graph. The search is performed by an A* algorithm utilizing an expression for the probability of detection.

Another set of target tracking problems from the field of artificial intelligence is cluster and Q-learning. In Prabhavathi and Rajeshwari (2011), the target state is estimated using a Kalman filter. Then, a sensor is chosen to be the cluster head. This sensor gathers measurements of the target from the sensors in its area and fuses them together. As the target moves, the cluster head continues to track the target until it is outside of its area. The tracking information is then passed on by the cluster head to an adjacent cluster head. The strategy for choosing cluster heads is to maximize the lifetime of the network. The authors of Ferrari et al. (2011) combine a gradient-based approach for obstacle and collision avoidance with a Q-learning algorithm to decide the action for each sensor. Q-learning is a learning technique where the sensors decide what actions to take based on the information utilities of the available actions (Russell and Norvig 2002).

Finally, we have some approaches that are rule-based with a similar structure such as a set of if-else sentences. The authors of Krishna et al. (2004) study multiple-target searching. The area is modeled using a grid, dividing it into cells. When a target is discovered by a sensor, the sensor uses fuzzy control to decide whether to continue to search or to start to track that target. Another rule-based approach is that of Takahashi et al. (2009), which focuses on single-target tracking with obstacles which prevent communication to a base station. To maintain contact with the base station, each sensor uses a set of rules to decide its actions. The problem of multiple evasive target searching is studied by Rahman et al. (2011). Here, the authors suggest that the sensors scan for targets using different predefined formations.

An advantage with heuristic approaches is that they often will work better in real-world experiments as they often have fewer limiting assumptions. They can often also adapt to the situation they are in like, for example, Q-learning. A challenge is that it can be difficult to design good heuristic algorithms, and it can be hard to prove overall desired behavior.

Centralized Single Target Tracking (Bai et al. 2012, Cheng et al. 2012, Juan-Yi 2011, Li and Djuric 2007) (Hu and Tu 2017, Liu et al. 2007, Mahboubi et al. 2011; 2016; 2017; 2012) (Marbukh et al. 2010, Mourad et al. 2012, Park and Hutchinson 2013) (Qi et al. 2016, Zhao et al. 2014a)
Centralized Single Target Searching (Wei and Ferrari 2015, Wu et al. 2014)
Centralized Single Target Searching and Tracking ()
Centralized Multi Target Tracking (Albert et al. 2017, Baumgartner et al. 2009, Dang and Horn 2015) (Haugen and Imsland 2016, Liao et al. 2015, Lu et al. 2014) (Sharma et al. 2015)
Centralized Multi Target Searching (Ferrari et al. 2009, Rahman et al. 2011, Selvaraj and Balaji 2013)
Centralized Multi Target Searching and Tracking (Mathew et al. 2010)

Table 2.4: Classification of centralized approaches presented in this chapter

2.6 Classification of Trajectory Planning

The papers presented in this chapter (Sections 2.4 and 2.5) have multiple problem formulations. In addition, the solutions have some different characteristics. In this section, we present a classification of all the papers presented in this chapter.

We have chosen to classify each paper based on three criteria. First, we separate the papers based on whether the approach is centralized or distributed/decentralized. Second, does a paper usually focuses on a single target or multiple targets? Third, the categories tracking, searching and a combination of both are used to separate the papers. Tracking papers usually have a priori estimates of each target, and the focus is on tracking those targets. In papers about searching the authors suggest strategies for finding targets in a given area. Finally, the hardest problem is managing both. Here, sensors usually must weigh the trade-off between searching for new targets and tracking detected targets.

2.7 Discussion and Future Work

In this section, we will discuss the research within target searching and tracking, as well as suggest directions for future research efforts.

A challenge with this literature is that there are almost as many problem formulations as there are strategies to solve them. Both problem definition and the performance measurements are not well defined. This makes it hard to compare approaches and decide what the state-of-the-art is for a specific problem. The reason is that there are so many different applications involving targets for MSNs. However, it would be useful to have some clear problem definition, along with a performance score, to compare different solution strategies.

There is a lack of real-world experiments. Even with the immense potential for MSN there has yet to be reported an industrial-sized implementation. Most papers only demonstrate their approaches in simulations, and the few experiments reported are done in lab settings. More real-world implementations will also make the requirements for the MSN clearer.

Most papers focus on single-target tracking. However, using an MSN to track a single target can be too simple for many real-world applications. For example, in a search and rescue operation, it is not likely that there will be a sufficient number of sensors to cover the entire area and there might not even be a sensor per target. The challenge then becomes balancing both target searching and tracking with fewer sensors than targets. In most approaches for multiple-target searching and tracking, there are more sensors than targets, which makes it possible to split the sensors into different groups such that each target can be tracked separately.

Distributed Single Target Tracking (Dang et al. 2016, Giannini et al. 2012, Gu and Hu 2011, Gusrialdi et al. 2008) (Jin et al. 2017, Kan et al. 2012, Khodayari et al. 2016, Kuo et al. 2017) (La et al. 2009, La and Sheng 2009a;b; 2011b, Li and Liu 2007, Li et al. 2007) (Li et al. 2008, Li and Liu 2009, Ma et al. 2008, Mahboubi et al. 2010) (Martínez and Bullo 2006, Nelson and Freeman 2009a;b, Olfati-Saber 2007) (Olfati-Saber and Jalalkamali 2012, Prabhavathi and Rajeshwari 2011) (Su et al. 2017, Takahashi et al. 2009, Tan et al. 2004; 2017, Tu et al. 2012a;b) (Wang et al. 2010a;b; 2016; 2012, Xie et al. 2016, Zhang and Zhu 2015) (Gu and Hu 2010, Jiang et al. 2013, Rigatos 2011)
Distributed Single Target Searching (Chattopadhyay et al. 2015, Chen et al. 2012, Chin et al. 2010, Ferrari et al. 2011) (Hsu et al. 2013, Hutchinson and Bretl 2012)
Distributed Single Target Searching and Tracking (Hoffmann and Tomlin 2010, La and Sheng 2011a, Yanmaz and Guclu 2010) (Zou and Chakrabarty 2007)
Distributed Multi Target Tracking (Beaudeau et al. 2015, Fu and Yang 2014, Hung 2014, Jha et al. 2016) (Kamath et al. 2007, La and Sheng 2009c; 2012, Ren et al. 2016) (Rout and Roy 2016, Xu et al. 2010, Yang et al. 2008) (Zorbas and Razafindralambo 2013)
Distributed Multi Target Searching (Chang et al. 2015, Gao et al. 2017, Imai and Ushio 2013)
Distributed Multi Target Searching and Tracking (Hu et al. 2012, Jalalkamali and Olfati-Saber 2012, Krishna et al. 2004) (Low et al. 2004a; 2006; 2004b, Sun et al. 2014)

Table 2.5: Classification of distributed approaches presented in this chapter

When it comes to the development of efficient filters there is still a gap between research and real-world application. Very few articles focus on the data association problem, Section 2.4.4, which becomes even more difficult when measurements must be coordinated between multiple sensors. Another challenge with both Kalman and particle filters is that they require a model of the target they are estimating, in addition to a probability distribution for the measurement and process noise. It is possible to use general models, but these might be suboptimal as target behavior can be revealed through measurements. One potential future direction for research could be to introduce filter with learning ability. For example, first classify the type of target and then apply an appropriate model and probability density function.

Another related topic to filters is how long a target should be observed, which relates to the quality of the estimate. A common approach when using a Kalman filter is to use the covariance matrix of the estimate error, P, to measure the quality of the current estimate. However, this depends on the covariance of the process noise. If this is inaccurate it is difficult to know how long to stay with a target to obtain sufficiently confidence in an estimate. The quality of the estimate is also central to how different estimates should be merged together between multiple sensors, as discussed in Section 2.4.3 and 2.4.5. More research efforts should be put into quantifying the quality of target estimate. An additional application for this quantification could be to decide the trade-off between searching for new targets and tracking old targets.

All the observers presented in this survey are based on a theoretical formulation. This involves making assumptions that might be violated in an actual application. A more data-driven approach, using e.g. machine learning in combination with observer-based approaches, may be a promising way forward.

The high-level trajectory algorithms presented in this survey are often well suited for either searching or, more often, tracking. There are few strategies that deal well with both and has an efficient way to deal with the trade-off between them. This is reflected in Figure 2.4 and 2.5 where there are few papers that deals with the combination of both searching and tracking. In comparison many of the applications presented in Section 2.1.2 require the combination. Future research should aim at developing algorithms that can do both searching and tracking targets.

The high-level trajectory algorithms often have different weaknesses. For example, the assignment problem simplifies the dynamics of the sensors to suit the problem formulation, while optimal control problems often are challenging to initialize. A possible opportunity for future research could be to combine these two algorithms. For example by using the assignment problem to initialize an optimal control problem. Another possible combination could be to use a simplified problem solved by

either assignment or optimal control and use a heuristic algorithm to guide towards the solution. A third option could be to combine centralized and decentralized approaches. A case could be that the planning is done centrally when possible, but if communication between a sensor and base station is lost the sensor starts operating on its own.

2.8 Conclusion

In this chapter, we have discussed MSNs applied to target searching and/or tracking. The focus of the chapter has been on the state observer, which utilizes filters to estimate the states of one or multiple targets, and the trajectory planner, which uses information from the state observer to decide trajectories for the sensors. We have discussed the two most popular approaches in detail i.e., the Kalman and particle filters. In addition, we have also discussed some other types of filers. For the trajectory planning, we have divided the approaches into three: explicit, optimization, and heuristic. The explicit and optimization strategies have been divided into subgroups, each of which has been discussed in detail. The papers reviewed in this chapter have been classified based on problem and solution characteristics. Finally, we have discussed the current state of the research, as well as possible directions for future research efforts.

Chapter 3

UAV Path Planning using MILP with Experiments

This work is based on the work published in Albert et al. (2017), which is an extension of Albert and Imsland (2015). There are a few changes to make the terminology consistent with the reminding part of the thesis. In addition, it contains a section with background material both for both the algorithm in this chapter as well as part of the algorithm in Chapter 5.

In this chapter, we look at the problem of tracking icebergs (target tracking) using multiple Unmanned Aerial Vehicles (UAVs). Our solutions use combinatorial optimization for UAV trajectory planning by formulating a mixed integer linear programing (MILP) optimization problem. To demonstrate the approach, we present both a simulation and a practical experiment. The simulation demonstrates the possibilities of the MILP algorithm by constructing a case where three UAVs help a boat make a safe passage through an area with icebergs. Furthermore, we compare the performance of three against a single UAV. In the practical experiment, we take the first step towards full-scale experiments. We run the algorithm on a ground station and use it to set the trajectory for a UAV tracking five simulated icebergs.

3.1 Introduction

Offshore operations in ice-infested arctic areas demand ice management. Ice management is all activities that aim to reduce or avoid the impact of any kind of ice features. Several authors have argued that Unmanned Aerial Vehicles (UAVs) are an efficient platform to perform detection and surveillance of ice features for ice

management purposes, e.g. Eik (2008), Lešinskis and Pavlovičs (2011).

Using mobile sensors, like a UAV, for information gathering is a popular research topic. One of the reasons for its popularity is the many applications ranging from inspections of power lines, ships, pipelines etc., monitoring of traffic and environment, to border patrol, police support, surveillance and reconnaissance and filming for the entertainment industry (Valavanis and Vachtsevanos 2015).

In this chapter, we propose to use UAVs for iceberg tracking, specifically our contribution is an optimization-based trajectory-planning framework for this purpose. There are several different ways to approach UAV trajectory planning using mathematical optimization. For example, the problem can be formulated as a continuous time optimal control problem. Then, by exploiting well-known techniques like single or multiple shooting the problem can be transformed into a large scale nonlinear programming problem, for which there exist many commercially available solvers. Two examples of this approach are Haugen and Imsland (2013) and Walton et al. (2014).

The framework we propose in this chapter is based on combinatorial optimization. It is similar to the Traveling Salesperson Problem (TSP), which is to find the shortest path visiting each city in a given list of cities exactly once. A generalization of TSP is mixed integer linear programming (MILP). For information about modern MILP solvers and problems in general see e.g. Jünger et al. (2009), Bixby (2002), and Chen et al. (2010). In this chapter, we use the CPLEX solver from IBM (IBM 2015).

There are multiple applications similar to iceberg tracking with UAVs where TSP formulations have been applied. We have the problem of doing surveillance with unmanned ground vehicles (UGVs) in combination with a UAV. The UGVes have good sensing, but poor communication capabilities compared to the UAV. The UAV collects information by visiting UGVs. In Barton and Kingston (2013), the authors compare a TSP solution to an adaptive feedforward iterative learning control algorithm, based on the region of attraction for each UGV. Another similar problem is the heterogeneous, multiple depot, multiple UAV routing problem (HMDMURP). This is a generalization of TSP, where the salesmen are UAVs with different minimum turning radius and starting locations. The authors of Oberlin et al. (2010) come up with a approximate algorithm based on the transformation by Noon and Bean (1993). A related problem to HMDMURP is studied by Oh et al. (2015), which come up with a solution based on a modified algorithm for the Chinese Postman Problem to make it suitable for a group of Dubins Vehicles. The article Enright et al. (2005) focuses on the repairman problem, which is TSP with targets appearing according to a Poisson process. The authors calculate lower and

upper bounds for a Dubins path and use these to construct a centralized planner to assign areas for a set of UAVs.

Furthermore, there are a number of approaches using MILP that does not use TSP as a basis for problem formulation, which are similar to iceberg tracking with UAVs. The problem of searching and tracking targets in an urban environment using fixed wing UAVs is tackled by Hirsch and Schroeder (2015). The authors formulate the problem mathematically as a MILP, and solve it using an approximate method consisting of a greedy randomized adaptive search procedure and a simulated annealing. The military application of assigning targets of different priority and trajectory planning to combat UAVs is studied by Shetty et al. (2008). They also formulate the problem as a MILP and compare the CPLEX-algorithm (IBM 2015) to an approximate approach consisting of a tabu search algorithm with regard to optimally and computational efficiency. Schouwenaars et al. (2001) and Ma and Miller (2005) use the CPLEX-algorithm to solve a trajectory planning problem formulated as a MILP for a single or multiple UAVs.

Finally, there are many formulations similar to iceberg tracking that are not solved using a TSP framework. In Sinha et al. (2005a) the authors consider tracking hostile targets moving in group using multiple UAVs. Their solution is an adaptive decentralized algorithm. This is similar to the radar resources distribution and combat management problem, which are solved with a multi-level tree algorithm in Asnis and Blackman (2011). The authors of Farmani et al. (2015) track moving targets in an urban environment, where they take into account UAV and gimbal poses. They also solve the problem decentralized by using an auction method. The trajectory planning for each UAV is done with MPC (Model Predictive Control).

We choose to not consider gimbal pose, nonholonomic constraints (like the Dubins vehicle), decentralization, etc. for the benefit of a simpler formulation, similar to TSP, called the target visitation problem (Grundel and Jeffcoat 2004) (each iceberg has a value and high values get prioritized for an earlier visit). The reason is that we expect to have communication link with all UAVs and to be covering a vast area, where dynamics like gimbal pose and non-holistic constraints will be small compared to the Euclidean distance between the icebergs. This enables us to solve the problem fast for a limited number of UAVs and icebergs, which then enables real-time implementation. We take into account the dynamics of the problem by implementing the solution similar to an MPC, meaning that we solve a static optimization problem often and update the dynamics of the UAVs and iceberg between each time we run the optimization.

3.1.1 Contribution

The contribution of this chapter is a practical implementation of target visitation algorithm to track moving targets. The planning is centralized and the optimization formulation allows for the use of multiple UAVs. We compare to approach using three UAVs to one UAV. In addition, we demonstrate the first step towards practical experiments with a test performed in Ny-Ålesund at Svalbard.

3.1.2 Organization

This chapter is arranged as follows. In Section 3.2, we introduce the problem formulation through a scenario. In Section 3.3 we explain the setup, the modeling of the icebergs and UAVs, and the observer we use. Section 3.4 contains background material for both the formulation in this chapter and the MILP formulation used in Chapter 5. Then, we introduce some assumptions in order to use the target visitation formulation on our problem, and describe how we formulate the problem using mixed integer linear programming in Section 3.5.

To demonstrate the use of multiple UAVs and the possibilities of the algorithm, we introduce a scenario with a boat moving through an iceberg infested area. We perform a simulation of the scenario, compare it to a single UAV, and illustrate the results in Section 3.6. In Section 3.7, we present the experimental results from Ny-Ålesund at Svalbard. Finally, in Section 3.8 and 3.9 we discuss the results and conclude the work of this chapter.

3.2 Problem Formulation

To motivate the problem formulation we consider a concrete case: A boat traveling in an arctic area, where there are drifting icebergs. The boat must avoid collisions with icebergs. Monitoring the surrounding icebergs is necessary for safe operation. One might consider using satellite images for this task. However, in arctic areas the update frequency is often too slow and image resolution is too low to provide sufficient warning for the operations (Eik 2008). Another solution might be to use marine radar, but we assume that the range of a marine radar is insufficient for detecting icebergs early enough for this boat to be able to maneuver safely and efficiently. An iceberg on collision course requires time to take appropriate action. In addition, marine radars suffer from performance degradation due to poor weather conditions such as rain and high waves (Eik 2008). An unmanned aerial vehicle (UAV) with a sensor capable of automatic (or manual) detection of icebergs can be flexible, cheap (compared to manned flights), efficient, and with better coverage than marine radar and has increased spatial and temporal resolution compared to satellites.

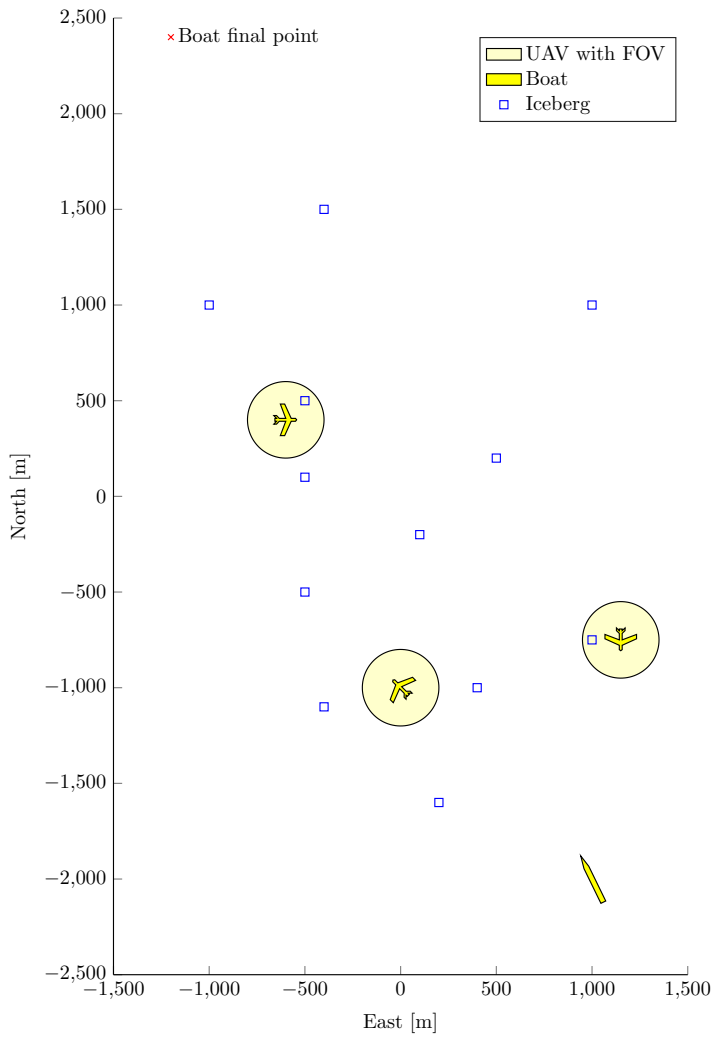


Figure 3.1: Illustrated scenario

The UAV can, for example, be equipped with an optical camera to detect icebergs. This camera will have a limited field of view (FOV), which only makes it possible to observe icebergs in a limited area, at the same time. To monitor a larger area the UAV must move around. A fixed wing UAV is best suited for this purpose, since it can cover relatively large distances.

We want to use multiple UAVs, so we can increase the area coverage. Our task is to make a trajectory planner for each UAV to do continuous tracking of all icebergs in a surrounding area.

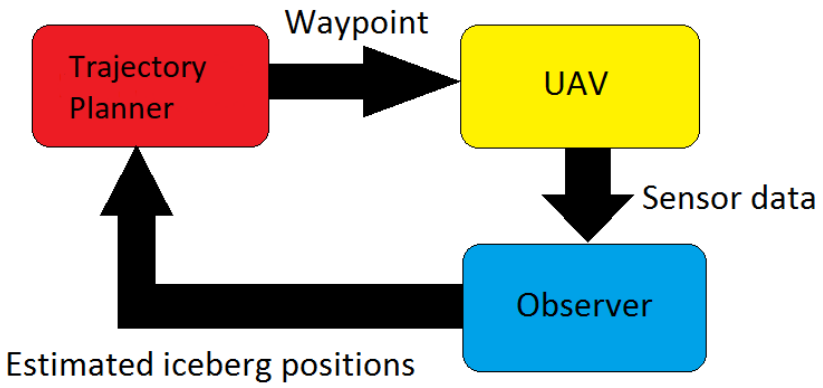


Figure 3.2: System components.

The scenario is illustrated in Figure 3.1. In the scenario, the boat (the yellow polygon) needs to move through an area with icebergs (blue squares). It has three UAV available with a limited field of view, which is illustrated by a light yellow circle. In the drawing, two of the UAVs are currently observing an iceberg each.

3.3 System Overview and Modeling

The system will consist of three components. The setup is illustrated in Figure 3.2. First, a set of physical UAVs, each with an autopilot capable of following waypoints. In addition, each UAV has a sensor able to detect icebergs. Second, the sensor data is sent to an observer to estimate iceberg position and velocity. Finally, the trajectory planner uses the iceberg positions and velocities to find a trajectory to track icebergs. The trajectory planner then sends waypoints to each UAV. The focus of this chapter is the trajectory planner.

We model the UAVs as Dubins vehicles,

$$\dot{z}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\psi}_i \end{bmatrix} = \begin{bmatrix} U\cos(\psi_i) \\ U\sin(\psi_i) \\ u_i \end{bmatrix} \quad \forall i \in [1, \ldots, n_{\text{UAVs}}] \tag{3.1}$$

$$u_{\min} \leq u_i \leq u_{\max} \tag{3.2}$$

where ψ_i is the heading, u_i is the bank angle, x_i and y_i are the Cartesian position, and U is the velocity of each vehicle (all the vehicles are assumed to move with the same velocity). The bank angel has to stay within the limits $u_{\min}$ and $u_{\max}$. Note that we assume constant altitude with this model.

We assume that the real position and velocity of each iceberg is governed by:

$$\dot{\xi}_i = \begin{bmatrix} \dot{s}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \xi_i + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_i(t) \quad \forall i \in [1, \ldots, n_{\text{icebergs}}] \tag{3.3}$$

where ξ_i is the four dimensional iceberg state consisting of position s_i and velocity v_i, both of dimension $\mathbb{R}^2$. The process noise is $w_i(t) \sim (0, q_i)$, which we assume has a Gaussian distribution with a mean of zero and variance of q_i.

To estimate the positions of the icebergs for the trajectory planner we use a discrete Kalman filter. The discrete equation and measurement model for each iceberg using Euler integration, is (for clarity we lose the subscript i for each iceberg)

$$\hat{\xi}_{k+1} = \begin{bmatrix} I & \Delta T \\ 0 & I \end{bmatrix} \xi_k + \begin{bmatrix} 0 \\ \Delta T \end{bmatrix} w_k = A\xi_k + \begin{bmatrix} 0 \\ \Delta T \end{bmatrix} w_k \tag{3.4}$$

$$y_k = \begin{bmatrix} I & 0 \end{bmatrix} \xi_k + v_k = C\xi_k + v_k \tag{3.5}$$

here $\hat{\xi}_{k+1}$ is the estimated state for the next time step, ξ_k is the discretized version of the state estimate at the current time step, y_k is the measured position of the iceberg and ΔT is the time step. The measurement noise, v_k, is also assumed to have a Gaussian distribution with a mean of zero and variance of R.

A Kalman filter tracks the estimated state of the system and updates an associated error covariance matrix. The update process of the filter consists of two steps. The first step is known as the a priori step, where the state is estimated based on the model and the error covariance matrix is updated with the model and process uncertainty. In the second step, known as the posteriori step, the measurement is incorporated to the estimated state and the error covariance matrix is reduced. For our case, the posteriori step will only be included when a UAV is observing the relevant iceberg, while the priori step will be updated each time step.

The Kalman equation for the a priori step will be

$$\hat{\xi}_{k+1}^{\text{priori}} = A\hat{\xi}_k^{\text{post}} \qquad \hat{\xi}_0 = \hat{\xi}_0 \tag{3.6}$$

$$P_{k+1}^{\text{priori}} = AP_k^{\text{post}}A^T + Q \quad P_0 = Q \tag{3.7}$$

here P is the error covariance matrix.

When a measurement is available, the Kalman filter performs the posteriori step

$$K = P_{k+1}^{\text{priori}} C^T / (CP_{k+1}^{\text{priori}} C^T + R) \tag{3.8}$$

$$\xi_{k+1}^{\text{post}} = \hat{\xi}_{k+1}^{\text{priori}} + K(y_k - C\hat{\xi}_{k+1}^{\text{priori}}) \tag{3.9}$$

$$P_{k+1}^{\text{post}} = P_{k+1}^{\text{priori}} - KCP_{k+1}^{\text{priori}} \tag{3.10}$$

here K is known as the Kalman gain. If no measurement is available $K = 0$ (the posteriori value equals the priori value)

We can now define the position uncertainty using the error covariance matrix

$$\sigma = \text{tr}(p_{11}) \tag{3.11}$$

where

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \tag{3.12}$$

Remark: If we select q_i in equation (3.3) different from the process noise, it will function as a heuristic to assign/manipulate priorities to the icebergs. Then, $\sigma(t)$ will represent priority instead of uncertainty. This heuristic can be set different for each iceberg, and can vary with time, $q(t)$. For example, the trajectory planner can set the initial priority, σ_0, from satellite imagery and the rate of change, $q(t)$, based on distance from a moving boat.

3.4 Background on MILP formulations for TSP

This section contains background material, which is the basis for both the algorithm presented in this chapter as well as the combinatorial part of the algorithm in Chapter 5. Readers familiar with MILP formulations can safely skip this section. The material is based upon Chen et al. (2010).

In the introduction, we mentioned the TSP (Traveling Salesperson Problem). We restate the formulation here. *TSP is the problem of finding the shortest tour visiting each city in a given list of cities exactly once.* We should add that the tour must start and end at the same city and there can be no subtours. We will discuss what a subtour is later in this section. As an example, Figure 3.3 illustrates the shortest path between the 104 cities of Norway. Note that when the distances between the cities were calculated, the Earth shape was taken into consideration by using the haversine formula.

There are, in general, two ways to formulate TSP as MILP problem. The first is the integer formulation, which we will use in this chapter. The second, is the binary formulation which we will use in Chapter 5. The difference between these approaches is how they handle subtour elimination.

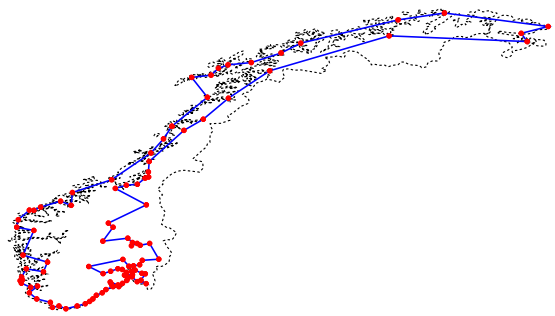


Figure 3.3: Shortest path between the cities of Norway. The Norwegian border is drawn with a black dotted line, each of the cities with a red dot, and the shortest path as a blue line.

The common part of the formulation for both these approaches can be written as

$$\min_{y} \sum_{i}^{N} \sum_{i}^{N} y(i,j)d(i,j) \tag{3.13a}$$

s.t.

$$\sum_{i}^{N} y(i,j) = 1 \qquad \forall j \tag{3.13b}$$

$$\sum_{j}^{N} y(i,j) = 1 \qquad \forall i \tag{3.13c}$$

where N is the number of cities and $Y \in \mathbb{Z}_{\{0,1\}}^{N \times N}$ is a binary matrix. Each entry in the matrix represent the path between two cities. A entry is denoted $y(i,j)$ and it is one if the path from city i to city j is included in the tour. Otherwise it is zero. The distances between all the cities are stored in $D \in \mathbb{R}^{N \times N}$, where each entry is denoted $d(i,j)$. The constraint (3.13b) makes sure each city is entered once, while (3.13c) makes sure that each city is exited once.

The optimal path illustrated in Figure 3.3 satisfies the constraints (3.13b) and (3.13c). However, there are many other solutions that also satisfy these constraint, but that consists of multiple subtours. A subtour is a tour that starts and stops at the same city, but does not include all the cities. We illustrate subtours by solving the instance form Figure 3.3 without subtour elimination. The result is illustrated in Figure 3.4. Instead one tour including all the cities, we get 21 different subtours.

To eliminate the subtours and get a single tour we need an additional set of con-

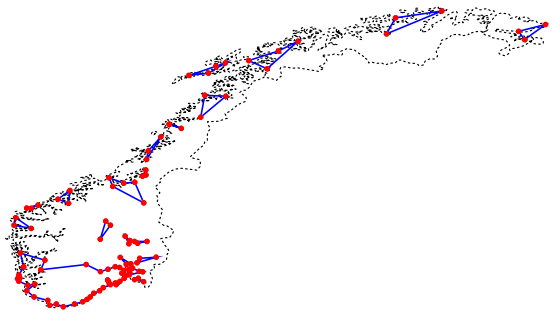


Figure 3.4: Solution when solving shortest path using the formulation in (3.13) without subtour elimination*. The Norwegian border is drawn with a black dotted line, each of the cities with a red dot, and the subtours with blue lines. *We have eliminated subtours of size 2 to make the figure more illustrative.

straints. For the integer formulation, we introduce an additional decision variable, an integer vector $t \in \mathbb{Z}^{N\times 1}$ with entries denoted $t(i)$. We then add the constraints

$$t(j) - (t(i) + 1) \geq -N(1 - y(i, j)) \quad \forall i \neq j, i \neq 1. \tag{3.13d}$$

These constraints are most commonly known as the Big-M or Miller-Tucker-Zemlin constraints. The integer vector represents the sequence each city should be visited. When we include a path from city i to j the constraint becomes active. This means that the right side becomes zero and the following city must have a value at least one higher than the preceding city. Note that the city indexed at one does not have this constraint. This enable us to return to this city. Which city is chosen as index one does not have effect on the algorithm or solution.

Instead of the Big-M constraints we can use binary constraints. Let S be every subset that consists of 2 to $N - 1$ cities. Note that this set grows exponentially with the number of cities. Then, we can formulate the following constraint

$$\sum_{i\in S}\sum_{j\in S} y(i, j) \leq |S| - 1 \quad \forall |S| = 2, 3, \ldots, N - 1 \tag{3.13e}$$

As mentioned above, this set of constraints grows exponentially with the number of cities in our problem. At first glance it might seem like a useless formulation for practical purposes. However, in an actual implementation not all these will be implemented, only the once we need. We find the once we need simply by solving the problem, check for subtours, and add the constraints that eliminates the subtours we got in the solution. For example in the case illustrated in Figure

Table 3.1: Comparison of computational complexity of integer and binary formulation for TSP. For each number of cities each formulation was used to solve the same 20 randomly initiated instances. The average computational time is displayed in the table. When the computational time was above 30 minutes, it is marked with "-" in the table. *For the case of 60 cities using the integer formulation, some of the instances had a computational time above 30 minutes. Here, we show the average of the instances with a computational time below 30 minutes. This presents the integer formulation favorably for the case of 60 cities.

Number of cities.	Integer Formulation	Binary Formulation
5	0.31s	0.24s
10	0.41s	0.38s
20	1.75s	0.78s
30	3.23s	1.56s
45	20.55s	3.53s
60	149.98s*	5.72s
75	-	9.01s
90	-	14.10s
120	-	27.40s

3.3 there are $2^{104} - 106 \approx 10^{31}$ constraints of type (3.13e). However, in practice, our implementation only required 5422 of those constraints before we obtained the solution with a single tour. For comparison, using the integer formulation we would need 10, 609 constraints of the type (3.13d) for the same problem.

The two subtour elimination constraints have different advantages. For a standard TSP formulation the binary formulation is, in general, computationally faster. Table 3.1 contains a comparison of the two formulation for randomly initiated instance of the TSP problem of increasing size implemented in Matlab with the CPLEX solver from IBM (IBM 2015). However, the integer formulation has other advantages. For some versions of the TSP problem, for example the one we use in this chapter, we need the decision variable t in the objective function. In addition, the integer formulation has a graceful performance degradation. This means that if we stop the solver before it finishes, we will get a feasible suboptimal solution that might be close to the optimum. In comparison, if we stop the binary solution prematurely we have a set of subtours that are often not useful for anything.

In this thesis, we use the integer formulation in this chapter, because we use a version of the TSP problem that requires the integer vector in the objective function and our problem size is limited. In Chapter 5, the problem size is bigger and we do not need the integer vector, so we use the binary formulation as a basis for the formulation we use in that chapter.

3.5 Problem Setup and MILP Formulation

We assume we have multiple UAVs to track icebergs in an area. Each UAV is capable of following a given sequence of waypoints. This motivates us to exploit mixed integer linear programming (MILP) to find an optimal sequence of icebergs to visit for each UAV. MILP problems are optimization problems that can contain integer variables in objective function and/or constraints.

3.5.1 Assumptions

We want a trajectory planner capable of real-time implementation and thus want to use a mixed-integer framework where the constraints and objective function are linear. To avoid nonlinearities, we make the following two assumptions:

1. The UAVs can follow any trajectory.
2. Icebergs are stationary within the horizon considered in the optimization.

Assumption 1 means that we do not take the UAV equation (3.1) into consideration in the trajectory planning, except for the distance from the initial position of the UAV to the first iceberg. This will inevitably lead to a suboptimal solution, since if we also consider the movement constraints of the UAV another visitation sequence might be better. However, the difference between the two visitation sequences will be small if the area is large compared to the turning radius of the UAV. In addition, the autopilot will manage to pilot the UAV to any waypoint even though it is not selected according to its dynamics. The second assumption enables us to solve the problem more efficiently. We can make this assumption since UAVs in general move much faster than icebergs (typically 0.1 m/s for icebergs against 22-25 m/s for UAVs). To take into account the slow movement of the icebergs we plan to rerun the optimization either at fixed intervals (sample-based), or every time a UAV observes an iceberg (event-based). Before we rerun the optimization, we update the iceberg positions and position uncertainties with the model from equation (3.6) and (3.11), and if available UAV observations and satellite imagery.

We also make some additional assumptions. First, we assume a constant number of icebergs that are known a priori. Icebergs can easily be added or subtracted in between optimization runs. Second, we assume perfect communication between the UAVs and a ground station. The ground station can perform the optimization and instantly communicate the result to each UAV. Finally, we do not consider the size of the icebergs, as we are only concerned with their location.

We consider anti-collision or fuel constraints for the optimization to be out of scope for this thesis. Anti-collision can be solved by a lower level controller like

the waypoint following controller on each UAV or the UAVs can just fly at different altitude. If we have fuel constraints this can be implemented through keeping track of the distance from base and compare it to remaining fuel. When the fuel gets low the UAV can be set to return to base and removed from the optimization problem.

3.5.2 Optimization variables

Now, we can formulate our problem using MILP. Our approach to formulating the optimization problem in a MILP framework uses the formulation for TSP in Miller et al. (1960) as a basis. Furthermore, we only use the subclass ILP (integer linear programming) of MILP, since we will only use integer and binary variables. The problem will have N nodes, which is the sum of UAVs (n_{UAV}) and icebergs (n_{iceberg}), $N = n_{\text{UAV}} + n_{\text{iceberg}}$. The optimization variables are organized in a matrix and a vector. First, we have a matrix of binary variables $y_{\text{path}} \in \mathbb{Z}_{\{0,1\}}^{N \times N}$. The entry $y_{\text{path}}(i, j)$ represents the path from node i to j. A node is an iceberg or a UAV. It is one if a UAV moves between the nodes and zero otherwise. Second, each UAV has an appurtenant sequence. In each sequence, the UAV is the first entry with the remaining part of the sequence consisting of icebergs. An integer vector, $t \in \mathbb{Z}^{N \times 1}$, represents the number each node has in its sequence. For example, if we have two UAVs, named UAV_1 and UAV_2, and five icebergs, named ice_1 to ice_5, then $t = [\text{UAV}_1, \text{UAV}_2, \text{ice}_1, \text{ice}_2, \text{ice}_3, \text{ice}_4, \text{ice}_5]^T$. Suppose the optimal solution is that the first UAV, UAV_1, visits iceberg ice_2, ice_5 and then ice_1, while the second UAV, UAV_2, visits ice_3 before ice_4. Then we will get the following $t = [1, 1, 4, 2, 2, 3, 3]^T$. Notice that the sequences for each UAV are mixed together in the vector t. We use the binary matrix y_{path} to assign icebergs to each UAV. The t-vector is used to avoid Hamiltonian subpaths, and to include position uncertainty in the objective function.

3.5.3 Constraints

Here, we will find a set of constraints that describe the set of feasible paths in terms of the optimization variables. First, each node cannot be visited and left more than once

$$\sum_{i}^{N} y_{\text{path}}(i, j) \leq 1 \quad \forall j \quad \text{and} \tag{3.14a}$$

$$\sum_{j}^{N} y_{\text{path}}(i, j) \leq 1 \quad \forall i. \tag{3.14b}$$

Next, the number of total paths between the nodes is equal to the number of nodes minus the number of UAVs,

$$N - n_{\text{UAV}} = \sum_{i=1}^{N} \sum_{j=1}^{N} y_{\text{path}}(i,j). \tag{3.15}$$

The t-variable is an integer vector that decides the sequence each UAV will visit icebergs, as explained above. The values of the vector must be within the number of nodes,

$$1 \leq t(i) \leq N \quad \forall i \in [1, 2, \ldots, n_{\text{UAV}}]. \tag{3.16}$$

The same integer vector t is used to represent the visitation sequence for each UAV. For example, if we have two UAVs in our problem, their sequence will be mixed together in the same vector. This is unproblematic since we use the binary matrix y_{path} to set the paths between UAVs and icebergs. We use the t-vector to prioritize high uncertainty icebergs in the objective function and to avoid subcycles. To make sure each UAV is the first of its sequence the first n_{UAV} elements in the t-vector representing the UAVs must be one,

$$t(i) = 1 \quad \forall i \in [1, 2, \ldots, n_{\text{UAV}}]. \tag{3.17}$$

Finally, each path must be connected. To avoid subcycles we need an additional constraint. The following constraint, called the Miller-Tucker-Zemlin constraint (Chen et al. 2010), makes sure that all the paths start with a UAV and are connected:

$$t(j) - (t(i) + 1) \geq -N(1 - y_{\text{path}}(i,j)) \quad \forall i \neq j. \tag{3.18}$$

3.5.4 Optimization Problem

We can now formulate the following optimization problem:

$$\min F(y_{\text{path}}, t) = -(1-\tau)F_1 + \tau F_2 \tag{3.19a}$$

$$\text{s.t. } (3.14), (3.15), (3.16), (3.17), \text{ and } (3.18)$$

where

$$F_1 = \sum_{i=1}^{N} \sigma(i)(N - t(i)) \tag{3.19b}$$

$$F_2 = \mu \sum_{i}^{N} \sum_{j}^{N} y_{\text{path}}(i,j) d(i,j). \tag{3.19c}$$

Here, μ is a scaling variable to make F_1 and F_2 of comparable size, see Section 3.5.5, and τ is a tuning constant used to weight between the two objectives. Setting $\tau = 1$ puts all weight on shortest distance and $\tau = 0$ puts all weight on position uncertainty. In the simulation and practical experiments of this chapter we found through experience the value $\tau = 0.5$ to be appropriate. In the objective function, we have two competing objectives. First, minimizing $-F_1$ equals sorting the icebergs in sequences based on their position uncertainty. Here, $\sigma(i)$ is the position uncertainty of each iceberg and UAV. The UAVs will have a position uncertainty of zero. The position uncertainty is constant when running the optimization. In between optimization runs it is updated with equation (3.11), which means it increases linearly with time for unobserved icebergs and decreases towards zero for observed icebergs. Second, F_2 contains the distance traveled by the UAVs. The matrix d contains the distances between all the nodes, which is recalculated for each iteration of the optimization problem. The distances are Euclidean distances except the distance from the UAVs to the icebergs. To calculate the distance from a UAV with a given heading to each iceberg we use Dubins paths without fixed final heading. A Dubins path is a curve with a minimum turning radius and a fixed initial and final heading. This way we take the nonholonomic dynamics from equation (3.1) of the UAVs into consideration for the path from each UAV to the first iceberg in each sequence.

3.5.5 Scaling

To get a proper trade-off of the objectives F_1 and F_2 in equation (3.19a) we need to scale them by choosing an appropriate value for μ in equation 3.19c. If we knew the optimal values of F_1 and F_2 in advance of the optimization, a natural choice for μ would be the ratio between them. Instead we approximate this ratio by calculating the maximum value for F_1 with decision variables y_{path} and t, in addition to an estimated average value for F_2. The maximum value for F_1 is found by optimizing without constraints, which is easy and efficient to do. We cannot do the same for F_2, since the minimal F_2 will always be zero. Instead we take the d-matrix and calculate the average distance and multiply it by the number of paths we need in our solution, $F_{2,\text{avg}} = d_{\text{avg}}(N - n_{\text{UAV}})$. We can now choose:

$$\mu = \frac{F_{1,\max}}{F_{2,\text{avg}}}. \tag{3.20}$$

If we compare the μ from equation (3.20) with the ratio of F_1 and F_2 after we run the optimization in 1000 simulations, the guessed ratio from equation (3.20) has a mean of 89% and a standard deviation of 13%.

3.5.6 Dynamic Implementation Consideration

When using the static optimization formulation in equation (3.19) on what is in reality a dynamic problem, one of the UAVs can get into a loop where it flies between only one or two icebergs. The reason is that when resolving the problem, the formulation from equation (3.19) does not consider that an iceberg has already been visited and the optimal solution might be to visit the nearest iceberg first even though it has a very small position uncertainty. To remedy this, we calculate the average position uncertainty of all icebergs and exclude all icebergs from the optimization with a position uncertainty below 10 % of the average. This is typically the newly visited icebergs.

3.5.7 Practical Implementation Consideration

In the optimization formulation in equation (3.19) we do not require that each UAV must be assigned to an iceberg. This can lead to a UAV not being assigned to track any iceberg. In this case, the UAV is set to track the closest iceberg (calculated in Dubins distance without the final heading).

3.6 Simulation

To implement the optimization algorithm from the previous section we use MATLAB R2014b with the toolbox YALMIP (Löfberg 2004). YALMIP enables easy implementation of optimization problems. Furthermore, we use the CPLEX solver from IBM (IBM 2015).

To illustrate the possibilities of the optimization algorithm we construct a simulation case with a boat moving through an area with icebergs. We compare using three UAVs to a single UAV for monitoring icebergs. When the boat enters the area we know the position and velocity of the 10 icebergs. This is an unrealistic assumption. However, it is necessary since we have chosen to set the iceberg velocities higher than normal (see next paragraph). The UAV(s) launch from the boat. In our case, the boat is only included for illustrative purposes, so its trajectory is hard-coded.

In Table 3.2, the simulation parameters are given. The optimization of the UAV-trajectories is implemented sample-based, meaning that the optimizations are run using fixed time intervals. As mentioned in Section 3.5.1, icebergs typically move at a velocity of 0.1 m/s. In the simulation, we have chosen to set the iceberg velocity up till about 3 m/s. The reason is that we want to better illustrate the tracking capabilities of the algorithm, since it is also suitable to other tracking applications than icebergs.

Two snapshots from each simulation are shown in Figure 3.5. Here the boat is illus-

trated by a yellow polygon. The UAVs have a solid line for their recent movement and a dotted line for the sequence they plan to visit the icebergs. Each iceberg has a solid blue line for their recent movement and a blue "x" for their current position. The icebergs are also enumerated. A red circle indicates the observers estimated position of each iceberg.

In Figure 3.6, we see the average position uncertainty during the simulations compared for one and three UAVs. The reason the average position uncertainty falls somewhat at the end of the simulation for the single UAV, is that it stops tracking some of the icebergs it cannot find.

3.7 Towards practical experiments

To conduct the practical experiment in Ny-Ålesund at Svalbard there were multiple components involved. In this section, we first describe the components, including software involved in the experiments. Then, we give a description of the practical experiments conducted.

3.7.1 Setup

Figure 3.7 illustrates an overview of the components used in the experiment.

The UAV platform used for the practical experiments was an X8 from Sky Walker Technology (SkyWalkerTechnology 2016). This is a small light-weight off-the-shelf platform with an electric motor. It can carry a light payload of 1-2 kg and is able to stay in the air for about 60 minutes. We used a catapult to launch it into the air, as illustrated in Figure 3.8.

The components onboard the X8 are a single board computer (Odroid 2016) for processing, an autopilot (ArduPilot 2016), a communication link (Ubiquiti 2016) and a switch which enables communication between all of the components.

The single board computer runs DUNE (Pinto et al. 2012). DUNE, short for DUNE Unified Navigational Environment, is an open-source “runtime environment for unmanned systems onboard software”. This enables simple implementation of different tasks, such as reading sensor values or control of actuators, onboard the X8. In this experiment, we specifically use DUNE to receive waypoints from the ground station (i.e, the results from solving the MILP algorithm presented in Section 3.5), and translating and forwarding them to the onboard autopilot. The autpilot is then responsible for guiding the UAV using low level controllers to the given waypoints. Furthermore, DUNE is pulling telemetry data from the autopilot and forwarding it to the ground station, giving it the necessary information about the UAV to initialize the MILP algorithm and find the optimal visitation sequence

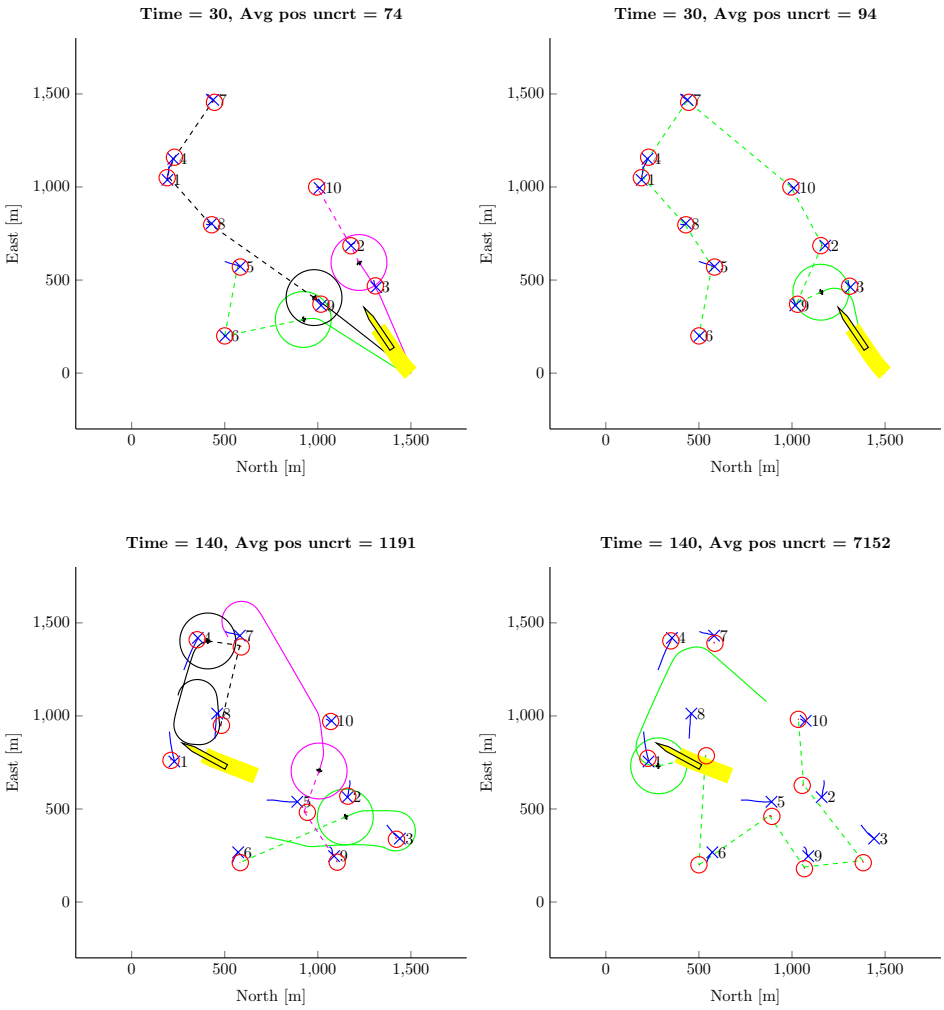


Figure 3.5: Simulations of a boat moving through an area with three or a single UAV(s) to track icebergs. The dotted lines illustrate the planned trajectories for the UAVs (green, black and pink), and the solid lines show their recent movement. In addition, the FOV is marked with a solid circle around each UAV. The iceberg positions are plotted with a blue "x", and have their recent movement in a solid blue line. The observer indicates the current estimate of iceberg positions with a red circles. The boat is drawn as a yellow polygon.

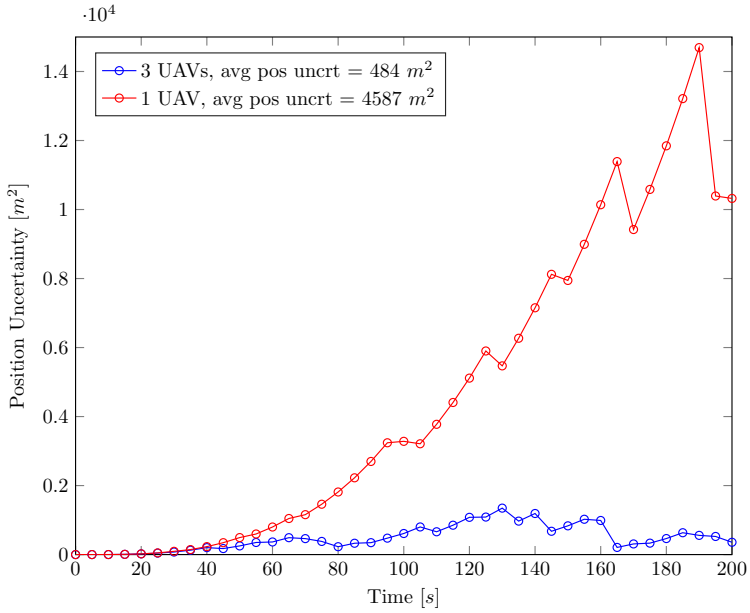


Figure 3.6: The average position uncertainty for the iceberg for the two simulations with 3 and 1 UAV(s).

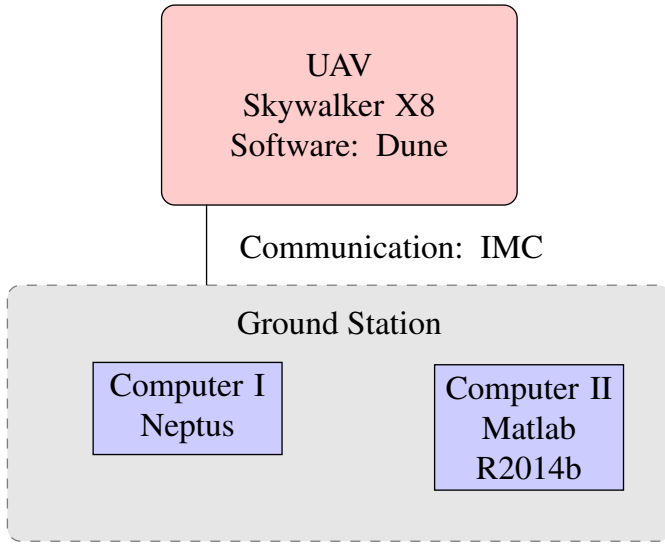


Figure 3.7: Practical Experiment Setup.

Figure 3.8: Launch of X8 using a catapult.

for the simulated icebergs.

The ground station consists of two computers, each running a different software. One of the computers is running MATLAB R2014b, which is where the MILP algorithm is implemented and run. The second computer is running Neptus (Pinto et al. 2006). Neptus is an open-source Command & Control Center which can be used for a variety of tasks. Examples are world representation and modeling, mission planning, simulation, execution control and supervision, and post-mission analysis. For the work presented in this chapter, Neptus is used to illustrate the location of the (simulated) icebergs, as well as the X8's position, telemetry data and current waypoint to the operator.

The communication between the UAV and the ground station is done using a protocol named Inter-Module-Communication (IMC), which is a set of predefined messages made for operations of (multiple) unmanned vehicles and real-time efficiency. The reader is referred to Martins et al. (2009) for a detailed description of the IMC protocol.

3.7.2 Experiment

On the way to full-scale practical experiments we did what can be considered a pilot experiment as a proof of concept. The experiment was conducted as follows:

1. Pilot launches the UAV in the air and stabilizes it at an altitude of about 120 meters.

2. The MILP-algorithm is initiated with four simulated icebergs and returns the optimal sequence for visiting them.

3. The UAV is sent a waypoint to the first iceberg from the resulting visiting sequence of the MILP-algorithm.

4. When the UAV reach the simulated iceberg it loiters around it for 30 seconds, while the ground station is running a new optimization.

5. The UAV is sent a waypoint to the first iceberg in the visiting sequence of the new optimization.

6. 100 seconds after the iceberg tracking mission is started, an additional simulated iceberg is added to the list, iceberg number 5.

7. The UAV continues to track the simulated icebergs by repeating step 3-5 for about 800 seconds of total loiter time.

The MILP-algorithm chooses a visiting sequence for the simulated icebergs based on position and an uncertainty value of each iceberg. The initial four icebergs where set with a random initial uncertainty value. After initialization, the position uncertainty increased constantly with a value of 1 m^2 per each second. In other words, $tr(p_{11})$ in equation (3.11) is 1 m^2/s. The fifth simulated iceberg was added with an uncertainty value of zero.

In Table 3.3 we see the coordinates of the simulated icebergs and their initial uncertainty value. Figure 3.9 illustrates the UAV's first visit of the five icebergs, with Figure 3.10 plots the position uncertainty of each simulated iceberg.

Unlike in Section 3.6, in the practical experiment we use an event-based approach for when to rerun the MILP algorithm. The image processing algorithm used to obtain position and velocity estimate from an iceberg might require the UAV to circle around the iceberg to get more measurements than just a flyby. In our experiment, an event was the 30 seconds loitering around a simulated iceberg, which makes it natural to exploit this time to rerun the MILP-algorithm.

3.8 Discussion

From the simulation in Section 3.6, a single UAV is not sufficient to track the ten icebergs. In the snapshot after 140 seconds the distance between iceberg number 8 and its estimated is larger than the FOV. In addition, we see that iceberg 2, 3, 5, 6 and 9 also are about to get outside FOV. This is not the case when in the simulation with three UAVs, which manage to keep track of the icebergs. When we look at Figure 3.6 the single UAV is unable to keep the position uncertainty within a limit, while three UAVs are sufficient.

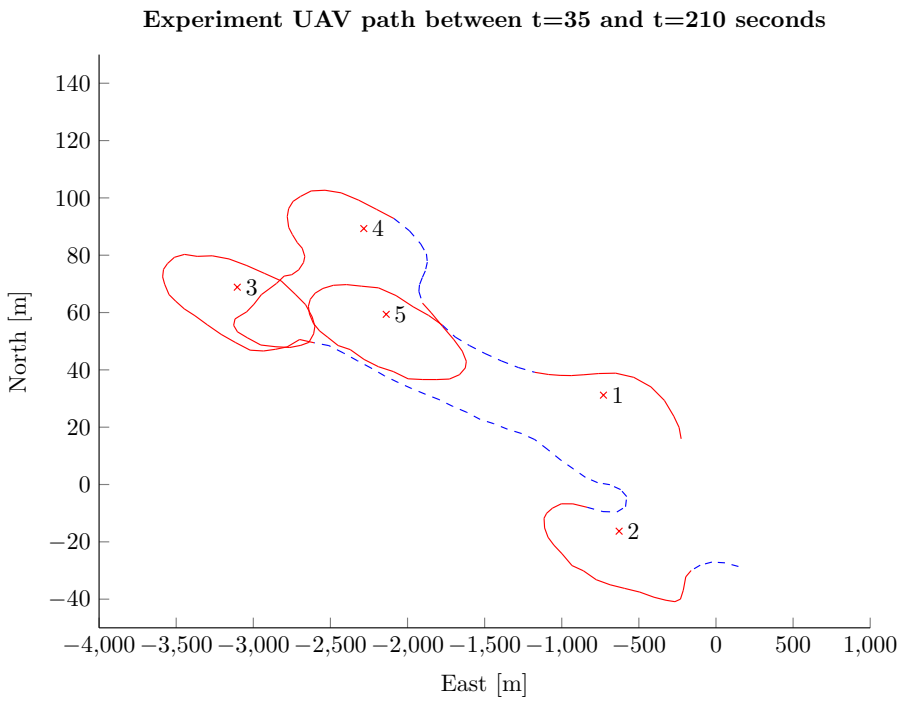


Figure 3.9: UAV path in practical experiments. Both the solid-red and dotted blue line illustrate the UAV path. The solid-red line indicates when the UAV is observing a point, while the dotted blue line is when the UAV is moving towards a new simulated iceberg.

The calculation of Dubins path reduces the sub-optimality originating from not taking UAV dynamics into account (cf. assumptions in Section 3.5.1). Especially, since we only use the first point of each sequence before we rerun the optimization. The way we reduce sub-optimality can best be explained by a simple example. Consider the case with one UAV and two icebergs, where the first iceberg is closer to the UAV in a metric distance. However, the UAV heading is turned towards the second iceberg, which makes the optimal sequence to visit the second before the first iceberg. When we use the Dubins instead of the metric distance, the algorithm manages to take the heading of the UAV into consideration. Notice that we are only able to use Dubins distance from the initial position of the UAV, since we do not know what heading the UAV will have at subsequent points.

The practical experiment serves as a proof of concept. It demonstrates how it is possible to implement the algorithm in practice. However, the practical experiment has several weaknesses. First, the simulated icebergs are too close to each other, for iceberg number 3, 4 and 5 the UAV goes directly from observing one iceberg

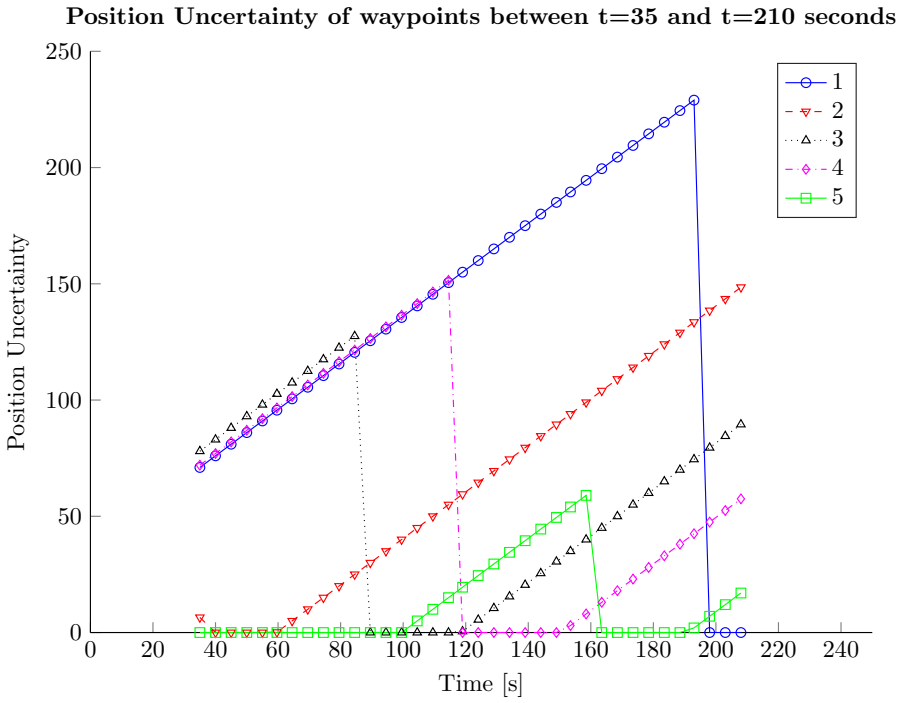


Figure 3.10: The position uncertainty value for each simulated iceberg during the experiment illustrated in Figure 3.9.

to another. In addition, when the icebergs are so close to each other the UAV will inevitably visit each iceberg often independent of the sequence chosen by the MILP algorithm. Unfortunately, during the day of the experiment there was a lack of real icebergs. This lead to not getting image processing tested together with the MILP-algorithm.

In future experiments, a platform with greater reach and greater durability is desired. Longer reach means that the UAV can track icebergs in a larger area, where selecting a more optimal visiting sequence for each UAV becomes essential. We believe the benefits of the proposed algorithm will be more substantial in such a case. Furthermore, in future experiments it will also be interesting to test tracking icebergs moving in open sea. The icebergs in Kongsfjorden close to Ny-Ålesund, where we did our experiments, do not move much and this makes them very easy to track.

Furthermore, for experiments in open sea the observer should be improved by considering clutter and the data association problem. Cluttering is false positive measurements. This can deteriorate the estimate of the icebergs. Data association is the

problem of matching measurement with estimate. For example, if two icebergs are close to each other at the time of observation, the observer might mislabel them. This will also deteriorate state estimate for both of them.

3.9 Conclusion

In this chapter, we have studied the problem of tracking moving icebergs using a set of moving sensor platforms. To solve the problem we have used mathematical optimization, or more specifically, combinatorial optimization. To demonstrate the algorithm we have constructed a case with 10 icebergs and performed a simulation with one UAV and another with three UAVs. The simulations show that a single UAV is unable to track all ten icebergs, while 3 UAVs manage it.

We have also taken the first step towards practical implementation with a practical experiment conducted at Ny Ålesund in Svalbard during the fall of 2015. Here, we have made a successful first practical implementation of the algorithm with one UAV.

Future Work

Possibilities for future work include:

1. Integration of the of MILP-algorithm with an image processing algorithm like Leira et al. (2015).
2. Practical experiments with real icebergs and the use of image processing for iceberg detection.
3. Practical experiments with long distance UAV platform.
4. Use of multiple UAVs in practical experiments.
5. Extend observer to handle false positive and negative measurements, in addition to outliers.

Table 3.2: Simulation Parameters

Parameter	Value	Unit
Boat	1 unit	
(x_0,y_0,heading)	(1500,0,2.0246)	(m,m,rad)
Velocity	10	m/s
Min turning radius	587	m
UAVs	3 units	
(x_0,y_0,heading)	(1500,0,0) (1500,0,$\frac{\pi}{2}$) (1500,0,π)	(m,m,rad)
Velocity	22	m/s
Minimum turning radius	105	m
FOV	150	m
Icebergs	10 units	
x_0	$\in$[0,1500]	m
y_0	$\in$[0,1500]	m
v_x	$\in$[-3,3]	m/s
v_y	$\in$[-3,3]	m/s
Observer		
Time step, ΔT	0.1	s
Process noise, Q	$\begin{bmatrix} 0_{2\times2} & 0_{2\times2} \\ 0_{2\times2} & 0.01I_{2\times2} \end{bmatrix}$	$\begin{bmatrix} m^2 \\ \frac{m}{s}2 \end{bmatrix}$
Measurement noise, R	$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$	m^2
Measurement frequency	2	s^{-1}
Simulation		
Simulation length, T	200	s
Optimization sample time	5	s

Table 3.3: Table with simulated iceberg coordinates and initial uncertainty

Nr.	Iceberg		Initial Value
1	(730	31)	1
2	(628	-16)	5
3	(3104	69)	8
4	(2284	89)	2
5	(2138	59)	1

Chapter 4

Numerical Optimal Control Mixing Collocation with Single Shooting: A Case Study

This work is based on the work published in Albert et al. (2016). There are a few changes to make the terminology consistent with the reminding part of the thesis.

This chapter looks into implementation of numerical optimal control problems of systems with a cascade structure, in which only one part of the dynamic equality constraints has path constraints. We consider two different direct strategies for numerical implementation using direct methods: 1. Collocation for both parts of the cascade. 2. Direct collocation for one part and single shooting for the other. To compare the methods we study the case of iceberg monitoring using a single unmanned aerial vehicle. The study reveals that the second method, under some conditions can be more computationally efficient than the first method.

4.1 Introduction

In this chapter, we study implementation of different numerical methods for continuous time optimal control problems (OCPs) formulated as autonomous cas-

caded nonlinear systems:

$$\min_{u(\cdot)} \int_{t_0}^{t_F} L(p, z, u)dt + E[p(t_F), z(t_F)] \tag{4.1a}$$

s.t.

$$\dot{p} = f_1(p, z, u), \quad p(t_0) = p_0 \tag{4.1b}$$

$$\dot{z} = f_2(z, u), \quad z(t_0) = z_0 \tag{4.1c}$$

$$z_{\min} \leq z \leq z_{\max} \tag{4.1d}$$

where $p \in \mathbb{R}^{n_p}$ and $z \in \mathbb{R}^{n_z}$ are state variables. The $u(t) : [t_0, t_F] \to \mathbb{R}^{n_u}$ is the control input. The objective function consist of the *Lagrange term*, $L(p, z, u) : \mathbb{R}^{n_p} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}$ and the *Mayer term*, $E[p(t_F), z(t_F)] : \mathbb{R}^{n_p} \times \mathbb{R}^{n_z} \to \mathbb{R}$. It is solved over a time interval from $[t_0, t_F]$. In addition, we have dynamic equality constraints for both state variables: $f_1(p, z, u) : \mathbb{R}^{n_p} \times \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_p}$ and $f_2(z, u) : \mathbb{R}^{n_z} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_z}$. Finally, $z_{\min}$ and $z_{\max}$ are lower and upper limits for the z-state variable. Notice that we only have inequality constraints for one of the state variables.

The dynamic systems we study has a cascaded structure, see e.g. Loria and Panteley (2005) for examples. We choose to call the "outer state" p the system state, and the "inner state" z the actuator state. This naming is for convenience, and need not be consistent with all problems of this form.

Our problem belongs to the field of optimal control theory. Mathematicians like Bellman and Pontryagin developed this field of mathematics during the 1950s (Pesch et al. 2009). A breakthrough in the research of optimal control theory came with the *Pontryagin's maximum principle* (Pontryagin 1957). This principle states necessary conditions for optimal control problems in continuous time. We can use these conditions to eliminate the controls, u, from the problem and get a boundary value problem, which we can solve numerically. This is referred to as an indirect approach to optimal control. However, an indirect approach suffers from drawbacks like difficulty in initializing the problem (Betts 2010, Binder et al. 2001). Another approach for solving optimal control problems, which we focus on in this chapter, is the direct approach.

In a direct approach, the optimal control problem is first discretized, before the discretized problem is solved. This enables us to transform the optimal control problem to a nonlinear programming problem (NLP). NLPs have well developed solvers, which are efficient even for large problems, at least when they have structure.

4.1.1 Contribution

In this chapter, we investigate whether merging the objective function and the system state (the state without path constraints) into a new objective function, can increase computational efficiency. This is based on the premise that reverse algorithmic differentiation is efficient for scalar functions of many variables (Griewank and Walther 2008). This enables us to exploit the structure and compare different numerical implementation strategies for the new objective and the actuator state.

4.1.2 Previous Work

The general field of numerical optimal control is a large field, in which single shooting and collocation are standard methods. We recommend Betts (2010) and Biegler (2010) as a starting points.

The case we study in this chapter is trajectory planning using a mobile sensor for target tracking, where we directly build on the approach used in Haugen and Imsland (2013). Another interesting paper studying the same problem is Walton et al. (2014). Both these papers use collocation in implementing a nonlinear problem for path planning formulated similar as OCP (4.1). A difference between the two is the solver used; where Haugen and Imsland (2013) uses an interior-point solver as in this chapter, while Walton et al. (2014) uses a SQP algorithm.

This chapter starts with a formulation of the problem and implementation strategies in Section 4.2. In Section 4.3 we go into details for the implementation for the different approaches we use. We present the case we are using for simulation in Section 4.4. In Section 4.5 we explain the setup for the simulation. We run and discuss the results in Section 4.6 before we come with concluding remarks in the final Section 4.7.

4.2 Problem Formulation and Implementation Strategies

We want to explore discretization strategies in direct approaches for solving problems on the form of OCP (4.1) in a computationally efficient manner. There are broadly three discretization approaches: Single shooting, multiple shooting and collocation (Binder et al. 2001). Each of the approaches have their own advantages and disadvantages.

We apply two different strategies for implementation. First, we use collocation for both the system and actuator state. This is the same strategy as used by Haugen and Imsland (2013) and Walton et al. (2014). However, we use a different number of integration steps and degree of the collocation polynomial for the two states. We call this the pure collocation approach. Second, we want to exploit the structure of our problem. We can merge the objective function with the system state

into a scalar function using single shooting, for which evaluating the gradient has approximately the same complexity as evaluating the function itself using reverse algorithmic differentiation (Griewank and Walther 2008). For the actuator state, which contains both inequality and dynamic equality constraints, we apply collocation for easy handling of the inequality constraints. We term this the combined approach with exact Hessian. In addition, we extend the second approach into two additional approaches. Third, we use limited-memory BFGS-update for the Hessian, we term this approach BFGS. This will generally lead to more iterations, but avoid calculating the computationally expensive Hessian. Fourth, we use the Hessian and increase the convergence tolerance for the NLP-solver. We term this approach BFGS−. With this approach we avoid more iterations, but we might get suboptimal solutions.

4.3 Implementation

For implementation we use Python with CasADi (Andersson 2013a), which is "a symbolic framework for algorithmic differentiation and numeric optimization". CasADi is open-source and implemented in C++ with Python wrappers. We exploit the CasADi framework to use the NLP-solver IPOPT (Wächter and Biegler 2006). IPOPT is a primal-dual interior-point NLP-solver. We compile it with the linear algebra sparse direct solver MA57 (HSL 2015). We chose a interior-point solver over a SQP -solver. The single-shooting approach may fit a SQP-solver better, however we will exploit collocation for all our approaches that leads to huge problems with a sparse structure, for which an interior-point solver in general is a good match. Therefore, we do not include a SQP-solver in our simulations.

We use different strategies to approximate the integral in equation (4.1a) from problem (4.1) depending on our chosen implementation.

4.3.1 Collocation Approach

For the pure collocation approach we approximate the integral (4.1a) as a sum of states. When using only collocation we have all the states of the state variables available.

$$\min_{u(\cdot),p(\cdot),z(\cdot)} \sum_{n=1}^{N} \Delta t L(p_n, z_n, z_n) + E[p(t_F), z(t_F)] \tag{4.2a}$$

s.t.

$$\dot{p} = f_1(p, z, u), \quad p(t_0) = p_0 \tag{4.2b}$$

$$\dot{z} = f_2(z, u), \quad z(t_0) = z_0 \tag{4.2c}$$

$$z_{\min} \leq z \leq z_{\max} \tag{4.2d}$$

Here N is the number of integration steps in approximating the integral. We use collocation for both the state and actuator state.

4.3.2 Combined Approach

In the combined approaches we embed the system dynamics (the p-dynamics) into the objective, by solving it by means of single shooting. For this type of cascade systems this will always be feasible. To illustrate this, we formulate the optimization problem in the following manner

$$\min_{u(\cdot),z(\cdot)} \quad c(z(\cdot),u(\cdot);p_0,t_f) \tag{4.3a}$$

$$\text{s. t.}$$

$$\dot{z} = f_2(z,u); \quad z(t_0) = z_0 \tag{4.3b}$$

$$z_{\min} \leq z \leq z_{\max} \tag{4.3c}$$

where the only dynamic constraint is the actuator dynamics (z-dynamics). The function c is a a scalar function obtained by solving the system dynamics (e.g. by single shooting),

$$p(t;z(\cdot),u(\cdot),p_0) = p_0 + \int_{t_0}^{t_f} f_1(p,z,u)dt,$$

and inserting this solution into the objective:

$$c(z(\cdot),u(\cdot);p_0,t_f) = \int_{t_0}^{t_f} L(p,z,u)dt + E[p(t_f),z(t_f)].$$

The actuator dynamics is still discretized using collocation (Biegler 2010). In our implementation, we use a simple Euler method for the single-shooting discretization of p, and correspondingly a rectangle method for approximating the integral.

As previously mentioned, the objective of doing this is to "get rid off" all the equality constraints that the collocation of the system dynamics generates, by introducing a single, albeit more complex, objective function. The rationale, as mentioned above, is the effectiveness of calculating gradients of scalar functions using reverse algorithmic differentiation. Note, however, that calculating the Hessian of this objective function will not enjoy particular implementation efficiency.

4.3.3 Combined Approaches with BFGS-update

We use a third and a fourth approach, which are both equal to the previous approach, except we use a BFGS-update function instead of calculating the Hessian. This means that we use equation (4.3) for implementation. As previously mentioned calculating the Hessian for the objective function will not enjoy particular

efficiency, approximating it can therefore make the computations faster. However, it will in general lead to the NLP-solver using more iterations before finding a solutions. In the fourth approach we therefore increase the convergence tolerance to avoid many iterations, while trying to avoid getting a suboptimal solution. The default convergence tolerance in the IPOPT-solver is 10^{-8}, we use 10^{-3}.

4.4 Numerical Example

In this chapter, we will use the application of iceberg monitoring using an unmanned aerial vehicle (UAV) for comparing computational efficiency between the approaches. The target for iceberg monitoring is to use an UAV to verify the positions of icebergs. We assume we have an estimate of the position of the icebergs with an appurtenant uncertainty.

We can consider the actuator for this system the entire UAV. The UAV has an ODE describing its motions in addition to some limitations on the actuator for the UAV. To model the UAV we use a *Dubins Vehicle:*

$$\dot{z} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix} = \begin{bmatrix} U\cos(\psi) \\ U\sin(\psi) \\ u \end{bmatrix} \tag{4.4a}$$

$$u_{\min} \leq u \leq u_{\max} \tag{4.4b}$$

$$x_{\min} \leq x \leq x_{\max} \tag{4.4c}$$

$$y_{\min} \leq y \leq y_{\max} \tag{4.4d}$$

where U is the velocity, u is the actuator input, and z is a vector containing position (x, y) and heading (ψ) of the vehicle. We have a lower and upper limit u_{min} and u_{max} for the actuator input. In addition, the vehicle also has to stay within the area defined by x_{min}, x_{max}, y_{min}, and y_{max}. We use this area limit to formulate the objective function for the UAV.

To find an optimal path for the UAV we use the position uncertainty of each icebergs. To obtain a model for the uncertainty we use a simple model of each iceberg

and use a continuous Kalman filter to obtain the following equation:

$$\dot{p}_i = -\frac{p_i^2}{r_i} c_{FOV,i}^2(\hat{\xi}_i, z) + q_i \quad \forall i \in \{1, \dots, N\} \tag{4.5a}$$

where for all $i \in \{1, \dots, N\}$

$$c_{FOV,i} = e^{-k_0 d_i} \tag{4.5b}$$

$$d_i = ||z - \hat{\xi}_i||^2 \tag{4.5c}$$

$$\hat{\xi}_i = \hat{\xi}_{i,0} + v_i t \tag{4.5d}$$

$$q_i = \frac{q_{0,i}}{d_{max}} d_i \tag{4.5e}$$

$$d_{\max} = \max(x_{\max} - x_{\min}, y_{\max} - y_{\min})^2 \tag{4.5f}$$

where N is the total number of icebergs. $p_i \in \mathbb{R}$ is the position uncertainty, $\hat{\xi}_i \in \mathbb{R}^2$ is the position estimate, $\hat{\xi}_0 \in \mathbb{R}^2$ initial position estimate, $r_i \in \mathbb{R}$ is the measurement noise variance, $q_{0,i} \in \mathbb{R}$ is the position variance, $q_i \in \mathbb{R}$ is the modified position variance, $c_{FOV,i} \in \mathbb{R}$ is the field of view function, $d_i \in \mathbb{R}^2$ is the quadratic distance to the UAV, and $v_i \in \mathbb{R}^2$ is the velocity of each iceberg. k_0 is a scalar tuning constant for setting the field of view of the UAV and $t \in \mathbb{R}$ is time. d_{max} is the maximum distance of the allowable area for the UAV. We use this constant to modify the position variance, see the final subsection of this section.

To obtain (4.5) we use the following derivation: First, we have a naïve iceberg model and measurement model:

$$\dot{\xi}_i = v_i + \omega_i(t) \tag{4.6a}$$

$$y_i = c_{FOV,i}(\xi_i, z)\xi_i + \eta_i(t) \tag{4.6b}$$

where $\xi_i \in \mathbb{R}^2$ is the position, and $y_i \in \mathbb{R}^2$ is the measurement of each iceberg. $\omega_i \sim (0, Q_{0,i})$ and $\eta_i \sim (0, R_i)$ it the process and measurement noise. We assume that we are able to obtain estimates of the iceberg position and velocity through for example an infrared camera like in Leira et al. (2015).

Furthermore, we assume that the process and measurement noise are the same for both directions in the plane, such that we can write $Q_{0,i} = q_{0,i} I_{2\times 2}$ and $R_i = r_i I_{2\times 2}$. This assumption gives us a scalar function when we set up the covariance equation for a Kalman filter. This equation is the position uncertainty we get in (4.5a).

Note that we have been following the approach of Haugen and Imsland (2013) up to this point. However, the assumptions in the previous paragraph enables scalar covariance dynamics for each iceberg. In Haugen and Imsland (2013), the authors integrate a 2×2 covariance matrix instead of (4.5a).

The UAV can only observe an iceberg when the iceberg is within its field of view. Ideally, therefore the function $c_{FOV}(\hat{\xi}, z)$ should be a step function which is 1 when the UAV is observing the iceberg and 0 when it is not. However, to get a smooth problem for numerical implementation we approximate this function with the Gaussian kernel function (4.5b). The Gaussian kernel function is popular in support vector machine classification (machine learning) to measure similarities between features.

Finally, we modified the position variance of each iceberg for the NLP-solver to get a better-conditioned problem to solve. If we only use the field of view function from (4.5b), we might get a problem with convergence. To improve the convergence, we modified the variance of the position for each iceberg. We do this by calculating the maximum distance the UAV can have to an iceberg by considering the allowed area the vehicle can stay within, cf. (4.5f). We use this distance to make the position variance of each iceberg a sealed function of the quadratic distance between the iceberg and the UAV, resulting in (4.5e). This is similar as the modifications discussed in Haugen and Imsland (2013).

4.4.1 Optimization formulation

Our goal is to obtain smooth paths that reduce the overall position uncertainty of all the icebergs. We formulate this as the following objective function:

$$\min \int_{t_0}^{t_F} \mu_1 \sum_{i=1}^{N} p_i^2 + \mu_2 u^2 dt + \mu_3 \sum_{i=1}^{N} p_i(t_F)^2 \tag{4.7}$$

where μ_1, μ_2 and μ_3 are manually tuned weights. We use this formulation together with the constraints for the vehicle (the actuator states in the parlance of Section 2) and the iceberg uncertainty from (4.4) and (4.5) (the system states).

4.5 Simulation

In total there are four approaches, which we want to compare both with each other and with respect to how they handle increased complexity. In the iceberg monitoring case, we increase the complexity by increasing the number of icebergs. In simulations, we ran multiple cases for each problem of one to ten icebergs. Table 4.1 contains the parameters used in the simulations. To set up each case we used a random number generator for icebergs position, velocity and initial uncertainty. Notice that the time horizon is estimated. We desire a time horizon that enables the UAV to visit all the icebergs. This will depend on both the number of icebergs and position. To estimate the time horizon for each case we used the naive formula: $T_{est} = (\text{icebergs} \times \sqrt{\sigma_{East}^2 + \sigma_{North}^2} + \sqrt{\mu_{East}^2 + \mu_{North}^2})/\text{uavspeed}$,

Table 4.1: Simulation parameters.

Parameters	Value	Unit
T =Time horizon	Estimated	s
μ_1	10^{-4}	-
μ_2	10^{-6}	-
μ_3	10^{-3}	-
E_{max},E_{min}	± 500	m
N_{max},N_{min}	± 500	m
Desired Δt	2.0	s
z_0	$\begin{bmatrix} 0 & -500 & \frac{\pi}{2} \end{bmatrix}$	m/rad
g	9.81	m/s^2
U	22.0	m/s
u_{min}	$-\frac{g0}{U}\tan(\frac{5\pi}{36})$	1/s
u_{max}	$\frac{g0}{U}\tan(\frac{5\pi}{36})$	1/s
ξ_0	$\in [-500, 500]$	m
v	$\in [-0.2, 0.2]$	m/s
p_0	$\in [100, 300]$	m^2
q_0	1.0	m^2
r	$5 \cdot 10^3$	m^2
k_0	$3.3 \cdot 10^{-5}$	-

where σ and μ is the variance and mean, respectively, (from the UAV-position) of the icebergs.

A challenge when comparing the pure collocation approach to the other three approaches is to choose the number of integration steps and degree of the collocation polynomial. The solution we obtain depends to some degree on the choices we make, due to the approximation and the non-convex nature of the problem. To meet this challenge we chose a polynomial degree of 4 for the UAV dynamics in all cases and a degree of 2 for the iceberg dynamics in the pure collocation approach. In all the collocation polynomials we use Gauss-Legendre polynomials. Furthermore, we used an Euler method (1. order method) for the icebergs in the combined and BFGS approach. Initially we use the same number of integration

Table 4.2: Computational time in seconds, when requiring equal objective function value.

Icebergs	1	2	3	4	5
Collocation	0.1	0.58	1.43	3.23	5.57
Combined	0.13	1.21	3.64	9.47	20.52
BFGS	9.43	1.24	2.01	7.76	17.96
BFGS−	4.37	0.59	0.94	2.49	3.21
Icebergs	6	7	8	9	10
Collocation	10.36	19.39	27.24	43.33	55.49
Combined	35.43	89.60	140.37	277.08	366.55
BFGS	21.9	53.74	45.96	71.20	103.03
BFGS−	5.54	7.92	10.61	12.83	18.00

steps for all approaches, calculated by using a desired time step and the estimated time horizon. To compare the solutions we use for simplicity only the first term of the objective function (4.7). After the initial run, we chose the best solutions and called it the winner of that case. For the other three solutions, called the losers, we increased the number of integration steps with 10 % and reran the optimization. We compared the losers to the winner and if the difference was less than a value of $5m^2$ we accepted the solution. If not, we continued to increase the number of integration steps with 10 % up till a maximum of 10 times. In the few (13 of 500) cases where we failed to find a comparable solution for the four approaches, we discarded the results. Note that due to the non-convex nature of the problem we might not get a better solution by increasing the accuracy of the integration in the optimization, it might even be worse. However, the chance of getting an improved solution typically increase with more integrations steps. Figure 4.1 shows a typical case for a problem of 7 icebergs. Note that at the end the solutions deviate. This is an example of the non-uniqueness of the solutions of this problem formulation. In total, we ran 50 cases for each problem of one to ten icebergs.

Table 4.2 contains the computational time for the four cases, when all solutions have equal values for the objective function. Table 4.3 contains the initial solutions when all the cases use the same number of integrations steps (that is, they might have achieved different objective function values). To look further into the details of where time is expended in a typical optimization, we have given the computational distribution for the problem of seven icebergs in Table 4.4.

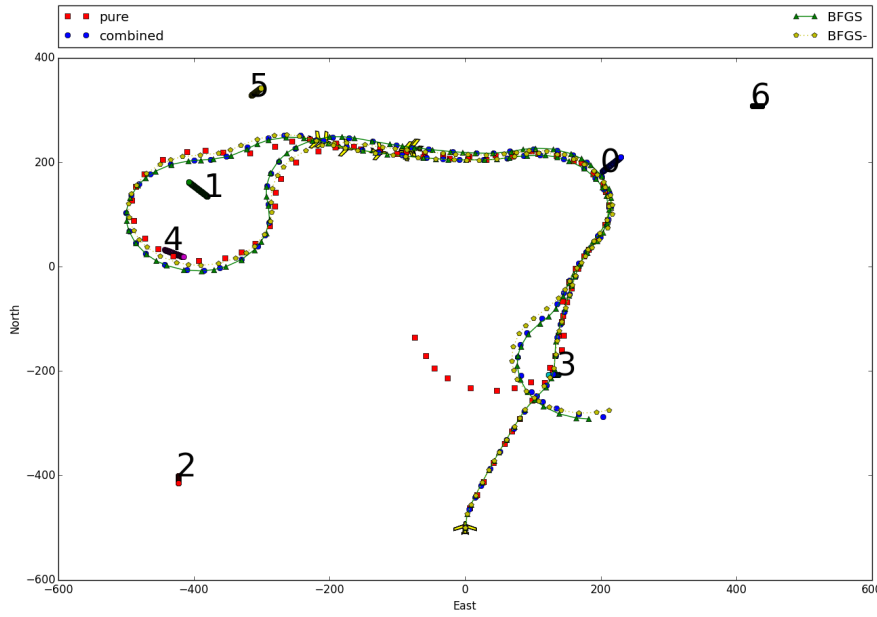


Figure 4.1: Optimal paths case of 7 icebergs. The dotted short lines with a number are the iceberg and their movement. The long curves are the UAV for the different approaches. Notice that the UAV does not have to fly closer than the field of view to observe the icebergs. We can adjust the field of view by changing k_0 in (4.5b).

Table 4.3: Computational time in seconds, with equal number of integrations steps.

Icebergs	1	2	3	4	5
Collocation	0.1	0.6	1.36	3.07	5.08
Combined	0.13	1.23	3.32	9.50	19.24
BFGS	12.39	5.05	1.63	8.07	13.46
BFGS−	4.36	4.65	0.9	2.38	2.79
Icebergs	6	7	8	9	10
Collocation	8.97	14.16	24.16	33.57	52.44
Combined	33.60	62.61	105.56	165.39	248.79
BFGS	23.50	32.55	53.70	59.65	57.71
BFGS−	5.82	6.97	9.38	12.11	14.11

Table 4.4: Average computational distribution and variables for the problem of 7 icebergs over 50 cases (N is the number of integration steps).

Average	Pure	Combined	BFGS	BFGS−
Iterations	410	388	2290	416
Iterationtime	44.43 ms	202.05 ms	20.68 ms	18.47 ms
Function	0.85 ms	0.71 ms	0.71 ms	0.68 ms
Gradient	3.9 ms	1.80 ms	1.81 ms	1.74 ms
Hessian	5.05 ms	128.16 ms	-	-
Variables	$27N + 10$	$13N + 1$	$13N + 1$	$13N + 1$

4.6 Discussion

The combined approach manages to calculate the Gradient faster than the pure collocation approach, as expected. If we look at the computational distribution of 7 icebergs over all 50 cases, from Table 4.4, we see that the combined approaches manages to calculate the gradient about twice as fast, and the number of variables is reduced by the number of icebergs $\times(2N + 1)$, where N is the number of integration steps. However, when we use the exact Hessian for the combined approach whatever we gained on fewer variables and gradient calculation, is lost on the Hes-

sian. With the lost sparsity of the Hessian it becomes about twenty five times as expensive to calculate, even though it is much smaller. By approximating the Hessian with a BFGS-update we get rid of the Hessian calculation at the expense of using more iterations to obtain a solutions. Compared to using the exact Hessian the BFGS-approach uses close to ten times as many iterations. In the last approach when we increase the tolerance and thus reducing the number of iterations. The approach with increased tolerance and BFGS-update for the Hessian in the combined approach is superior when it comes to computational time compared to the other approaches.

Some might consider increasing the convergence tolerance for the BFGS-approach to be cheating. However, doing the same for the combined-approach with exact Hessian and the pure collocation approach does not improve computational time, but in many cases makes it worse.

The objective function values could be the same in the four approaches for the same case, while not having exactly the same optimal path. An example is in Figure 4.1 where the collocation deviate from the other approaches at the final iceberg. For this problem and similar non-convex problems, there might be multiple locally optimal solutions. Only small changes in the problem setup can lead to different solutions.

4.7 Conclusion and Further Work

In this chapter, we compare implementation strategies for continuous-time optimal control problems of systems in a cascaded form (4.1). We implement four different strategies. First, we use collocation for both parts. Second, we use single shooting for the system state combined with collocation for the actuator state. For third and fourth we use the combined approach with BFGS-update for the Hessian and in the only the fourth we increase convergence tolerance of the NLP-solver.

When we apply the combined approach we are able to compute the Gradient more efficiently than with the pure collocation approach, but calculation of the exact Hessian becomes very expensive. This makes the second approach the slowest. Switching to BFGS-update, gives the faster computational time. However, the BFGS-update lead to a huge number of iterations, which also make it slower than the pure collocation approach. Increasing the convergence tolerance, the fourth approach, leads to the combined approach to become more computational efficient than the pure collocation approach. This leads us to the conclusion that cascade systems on the form of (4.1) might benefit computationally to be separated into two parts and apply single shooting on the system state, while using collocation on the actuator state.

Future work include:

- Explore scalability with expansion of the actuator state. For our case, this means increasing the number of UAVs.
- Test the four different approaches on other cases on the form of the OCP in (4.1).

Chapter 5

Combined Optimal Control and Combinatorial Optimization for Searching and Tracking using an Unmanned Aerial Vehicle

This work is based on the work Albert and Imsland (2018a), which is an extension of Albert and Imsland (2017). There are a few changes to make the terminology consistent with the reminding part of the thesis.

Combined searching and tracking of objects using Unmanned Aerial Vehicles (UAVs) is an important task with many applications. One way to approach this task is to formulate path-planning as a continuous optimal control problem. However, such formulations will, in general, be complex and difficult to solve with global optimality. Therefore, we propose a two-layer framework, in which the first layer uses a Traveling-Salesman-type formulation implemented using combinatorial optimization to find a near-globally-optimal path. This path is refined in the second layer using a continuous optimal control formulation that takes UAV dynamics and constraints into consideration. Searching and tracking problems usually trade-off, often in a manual or ad-hoc manner, between searching unexplored areas and keeping track of already known objects. Instead, we derive a result that enables prioritization between searching and tracking based on the probability of finding a new object weighted against the probability of losing tracked objects. Based on this result, we construct a new algorithm for searching and tracking. This algorithm is validated in simulation, where it is compared to multiple base cases as well as

a case utilizing perfect knowledge of the positions of the objects. The simulations demonstrate that the algorithm performs significantly better than the base cases, with an improvement of approximately 5-15%, while it is approximately 20-25% worse than the perfect case.

5.1 Introduction

Searching and tracking moving objects using, for example, a Unmanned Aerial Vehicle (UAV), has a broad range of applications. For arctic areas, ice management is crucial for safe operations, in which detecting and tracking icebergs is a key feature (Eik 2008, Lesinskis and Pavlovics 2011) . Another potential marine application is search and rescue (Goodrich et al. 2008, Tomic et al. 2012). After a shipwreck, one or multiple UAVs can be used to search for and, once found, track people or lifeboats floating in the water. In addition to civilian purposes, there are both police and military applications, for example, combat scenarios (DeSena et al. 2013, Eggers and Draper 2006, Glade 2000) and border patrol (Girard et al. 2004, Isenor et al. 2014).

There are multiple approaches spanning across several fields that deal with the problem of searching and tracking moving objects. In this chapter, we combine two well-known approaches; combinatorial optimization and optimal control.

Combinatorial optimization arose from several practical problems until they were unified in the 1950s by linear programming (Schrijver 2005). One of the most studied problems and the most relevant to this application is the traveling salesperson problem (TSP). TSP can be formulated as *Given a list of n-cities with an appurtenant matrix containing the distances between the cities, find the shortest tour visiting each exactly once.* The origin of TSP is hard pinpoint out, but the first breakthrough came in 1954 when Dantzig et al. (1954) managed to solve a 49-cities problem. Today, instances of 85,900 cities have been solved, see Applegate et al. (2011) for more information about TSP.

In this chapter, we use an alternative version of a TSP formulation, namely the prize collection TSP (PCTSP), which we use without penalties. PCTSP was first formulated by Balas (1989). Using PCTSP, we desire to *find a subcycle of cities amounting to at least a prescribed prize amount, where the salesperson gets a prize for each city that is visited. He pays a penalty for those cities not visited, and the objective is to minimize the travel and penalty cost while fulfilling the given prize amount.* Gutin and Punnen (2006) is a good starting point for the literature on PCTSP.

Optimal control is the study of finding a control law given some objective criteria for a system described by a set of differential equations. It was developed sim-

ultaneously in the U.S. and Soviet Union after WWII by mathematicians such as Bellman and Pontryagin (Pesch et al. 2009). A common way to solve continuous optimal control problems is to first discretize and then transform the equations into a large nonlinear programming problem (NLP). This is called the direct approach. In this chapter, we use collocation for discretization, see Betts (2010) and Biegler (2010) for details.

Search and Track (SaT) problems consist of two parts, each of which have been studied individually. We will discuss both here for their relevance to the two parts of our algorithm. Searching gained attention from the military with the increased use of German submarines during WWII (Stone 1989). Koopman laid the foundation for search theory with his work (Koopman 1956a;b; 1957), and today search problems are usually formulated as optimal control problems which are trying to maximize the probability of detection, and they are solved by transformation to large NLP problems, such as in Walton et al. (2014). For more details about optimal search see Stone et al. (2016).

Tracking can be considered task allocation, since the position of each target is assumed to be known or an a priori estimate is available. The simplest version of this is the aforementioned TSP problem. A generalization of TSP to multiple salespersons is the vehicle routing problem (VRP) (Golden et al. 2008), which has also been applied to UAVs (Oberlin et al. 2010). For more on task allocation for UAVs see Smith (2009).

For a successful application of a SaT problem, multiple tasks must be considered. For example, the trade-off between searching and tracking, decentralized vs centralized, communication constraints, data association problem (matching measurements to filter), safety of the environment, and so on. Here, we will focus on the trade-off between searching and tracking.

An approach to balancing searching and tracking is to separate the two objectives. The traditional way to do this is to use an objective function in which the operator manually weights the different objectives (DeSena et al. 2013, Yao and Zhao 2015). Another way is to separate searching and tracking into two different modes, and then use heuristic rules to switch between the modes (Furukawa et al. 2006, Meng et al. 2017, Zhao et al. 2016). A third option for separating the objectives is to perform the optimization in layers, in which, for example, the security of the vehicle is satisfied first before secondary objectives are considered (Tian et al. 2008).

In this chapter, we combine the two objectives into a single objective function. This can, for example, be done by defining both objectives in terms of information gain,

which is then maximized (Kassas et al. 2015, Peterson et al. 2014, Pitre et al. 2012, Sinha et al. 2005a;b). These approaches use a grid to divide the surveillance area. Mavrommati et al. (2017) develop an approach based on hybrid systems theory, in which they use a receding-horizon ergodic controller. A third approach is to define the problem as a Markov Decision Process (Oispuu et al. 2013, Vanegas Alvarez 2017). We introduce a novel approach in which we calculate the probability of losing the tracked objects and treat it equally with the probability of finding a new object.

Finally, we develop our algorithm based on a theoretical bound for the quality of the filter estimate for each target position. The problem of developing such bounds has been studied to some extent. The most common performance bound is the Posterior Cramér-Rao Lower Bound (Tichavsky et al. 1998) due to of its low computational complexity (Hernandez 2012). It has been applied to target tracking by UAVs (Esmailifar and Saghafi 2017, Koohifar et al. 2017)

5.1.1 Contribution

The contributions of this chapter are twofold. First, we derive a result for how often an object must be visited to keep the error in position estimate within a given confidence interval. We call this result the necessary visitation period. Second, it presents a new search and track (SaT) algorithm, which combines integer linear programming (ILP) with numerical optimization utilizing nonlinear programming (NLP). This has the advantage of obtaining a global optimum from the ILP, while considering the movement constraints of the UAV through the NLP. The resulting SaT algorithm has been validated in simulations, in which it has been compared to multiple base cases and a case utilizing perfect information about the movement of the objects. The simulations show that the new SaT algorithm outperforms the base cases.

5.1.2 Organization

We introduce the problem and the control architecture we use for the UAV and objects in Section 5.2. The control architecture consists of two types of observers and a path planning algorithm. We present the first type of observer, Kalman filters to estimate the state of each object, in Section 5.3. The second type of observer, a probability map to track the movement of the UAV, is found in Section 5.4. To quantify the deterioration of each tracked object, we derive a result in Section 5.5 which relates the covariance matrix of a Kalman filter to a probability. We call this result the necessary visitation period (NVP). Using the observers and NVP, we suggest a search and track algorithm, which combines combinatorial and optimal control in Section 5.6. To validate the new algorithm, we introduce a

simulation framework with multiple base cases and an approximately best case for comparison. Then, we compare our algorithm to this set of approaches in Monte Carlo simulations in Section 5.7 and discuss the results in Section 5.8. Finally, we conclude and discuss further work in Section 5.9.

5.1.3 Notation

Throughout this chapter the notation $\boldsymbol{I}$ and $\mathbf{0}$ will mean the 2×2 identity and zero matrices. We distinguish between different variables as follows: scalars are represented by lowercase letters, vectors be bold lowercase letters, and matrices by bold uppercase letters. Bold upper case letters are also used for sets, and the meaning will be clear from the context.

5.2 Problem formulation and Control Architecture

We consider an open area defined by a 2D Cartesian coordinate system containing an unknown number of moving objects. Our task is to find and, once found, track the location of each object. We have a mobile sensor, for example, a UAV, with a limited sensing capability available. Figure 5.1 illustrates the problem. We will make some simplifying assumption to this problem in the next Section to focus our scope of this chapter to the path planning algorithm for the UAV.

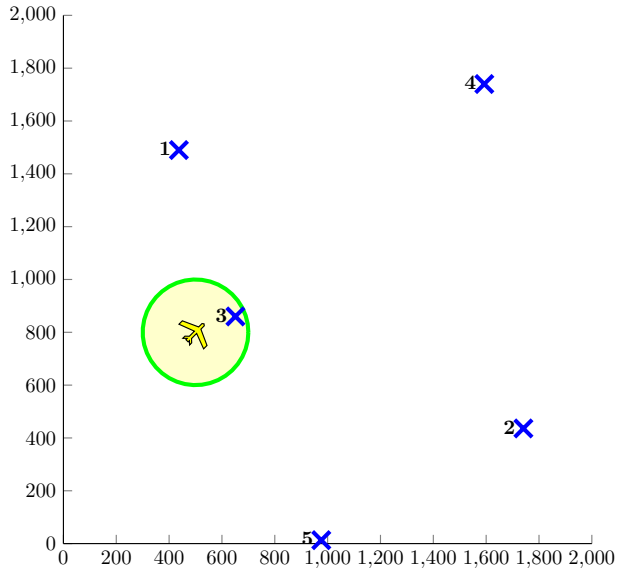


Figure 5.1: Problem illustration. One UAV (yellow polygon) with limited field of view (circle with green border and light-yellow area) monitoring an area of size 2000 x 2000 m^2 with 5 moving objects (blue X's numbered one through five).

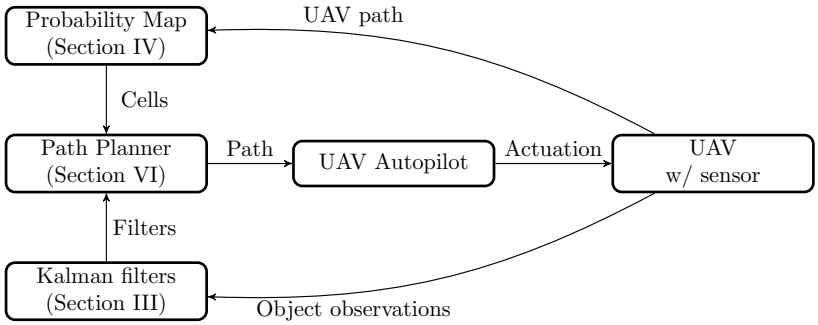


Figure 5.2: Control architecture. Notice that the probability map and the Kalman filters are constructed independently from each other and then combined in the path planner.

The control architecture we have chosen is illustrated in Figure 5.2. It consists of a path planner that utilizes two aggregated observer-type units. The first observer-type unit is a set of Kalman filters that fuse observations of objects with simple velocity models. The second observer-type unit is a probability map that store the UAV's movement in the open area and then calculate where new objects most likely are located. The path planner use the estimated positions of the objects along with the most likely positions of new objects when calculating the path for the UAV. We assume that the UAV has an auto-pilot, such that the path planner does not have to considering actuator inputs directly.

5.3 Kalman filters for Moving Objects

To model the objects, we use a general near-constant velocity model

$$\dot{\boldsymbol{\xi}}_i = \begin{bmatrix} \dot{\boldsymbol{s}}_i \\ \dot{\boldsymbol{v}}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_i + \boldsymbol{w}_i(t) \quad \forall i \in [1, \ldots, n_{\text{object}}] \tag{5.1}$$

where the state of an object, $\boldsymbol{\xi}_i$, contains the position, $\boldsymbol{s}_i$, and velocity, $\boldsymbol{v}_i$. Both are given in 2D Cartesian coordinates. The subscript i denotes object number i. The objects are moving with a constant velocity with the exception of a small process noise, $\boldsymbol{w}_i(t)$, assumed to be Gaussian distributed and characterized as $\boldsymbol{w}_i(t) \sim \mathcal{N}(\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T, \boldsymbol{Q})$.

We use a set of n_{object} discrete Kalman filters to estimate object positions and velocities based on measurements from the sensor onboard the UAV. A Kalman filter is the optimal estimator for linear systems with Gaussian noise (Gelb 1974). Examples of sensors are a camera, radar, and spectral camera (Eik 2008). The sensor will have a limited detecting range, referred to as the field of view (FOV). An advantage of a Kalman filter is that, in addition to the state estimate, it has an

associated covariance matrix to a measure the error of its estimate. The Kalman filter algorithm has two steps. First, the state estimate and covariance matrix are propagated. Then, the newest measurement is used to update the state estimate while considering the certainty of it. The certainty of the measurement is also used to update the covariance matrix.

We will modify the Kalman equations to account for the limited FOV, but first we will make some assumptions.

- Each object has a unique characteristic. This means that we ignore the data association problem, which is the task of linking observation with filter for tracked objects. In the case of iceberg tracking, this is not unrealistic. Each iceberg has a unique geometrical shape that can be used for association using image processing.
- We know the number of objects in the search area. This, together with the above assumption, simplifies and limits the scope of this article to the search and track algorithm. In practice, an estimate of the number of objects will often be sufficient.
- The sensor is capable of detecting multiple objects simultaneously. If the sensors is a camera, there are many algorithms that are able to detect multiple objects simultaneously from an image (Leira et al. 2015).
- Perfect sensing. This means that, when an object is within the FOV of the UAV, it has a 100% chance of detecting it, and there are no false positive measurements. In the case of detecting objects on the ocean surface with a camera, Leira et al. (2015) reports a 99.6% accuracy for detecting objects.

When modifying the Kalman equation, we use a similar approach to Sinopoli et al. (2004). The measurement equation for a single object is

$$\boldsymbol{y}_{i,k} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_{i,k} + \boldsymbol{v}_{i,k} \tag{5.2}$$

where $\boldsymbol{y}_{i,k}$ and $\boldsymbol{\xi}_{i,k}$ are the position measurement and time discretized state of object i at timestep k, respectively. The measurement noise, $\boldsymbol{v}_{i,k}$, is independent of object and time and modeled as

$$p(\boldsymbol{v}_{i,k}|\mu_{i,k}) = \begin{cases} \mathcal{N}([0 \quad 0]^T, \boldsymbol{R}), & \mu_{i,k} = 1 \\ \mathcal{N}([0 \quad 0]^T, \sigma^2 \boldsymbol{I}), & \mu_{i,k} = 0 \end{cases} \tag{5.3}$$

where $\mu_{i,k}$ is a binary variable equal to 1 if object, i, is within FOV of the UAV at timestep k, and 0 if it is not. As in Sinopoli et al. (2004), we use a “dummy"

observation for the absence of an observation. This is accomplished by noting that the absence of an observation can be modeled by letting $\sigma \to \infty$, which means that there is no information in the measurement, and it will, therefore, not be used to update the state estimate.

We follow the derivation from Sinopoli et al. (2004). The a priori step becomes (here we drop the subscript i for notational simplicity)

$$\hat{\boldsymbol{\xi}}_{k+1}^{\text{priori}} = \boldsymbol{A}\hat{\boldsymbol{\xi}}_k, \qquad \hat{\boldsymbol{\xi}}_0 = \begin{bmatrix} \boldsymbol{y}_0 & 0 & 0 \end{bmatrix}^T, \tag{5.4a}$$

$$\boldsymbol{P}_{k+1}^{\text{priori}} = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^T + \Delta_t\boldsymbol{Q}, \quad \boldsymbol{P}_0 = \begin{bmatrix} \boldsymbol{R} & v_{\text{R}}\Delta_t\boldsymbol{R} \\ v_{\text{R}}\Delta_t\boldsymbol{R} & v_{\text{R}}\boldsymbol{I} \end{bmatrix}, \tag{5.4b}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I} & \Delta_t\boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \quad v_{\text{R}} = \frac{1}{2}(v_{\max} - v_{\min}),$$

where $\hat{\boldsymbol{\xi}}_k = \hat{\boldsymbol{\xi}}_k^{\text{post}}$ is the discretized state estimate of an object with discretization timestep Δ_t. It is equal to the previous posteriori step state estimate. The superscript $priori$ denotes the prediction step, with the subscripts k and $k+1$ used for the current and next state, respectively. The state estimate is initialized with the first measurement, $\boldsymbol{y}_0$, and zero velocity. Associated with the state estimate is a covariance matrix $\boldsymbol{P}$.

The a posteriori step of the Kalman filter for each object is

$$\boldsymbol{K} = \boldsymbol{P}_{k+1}^{\text{priori}}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_{k+1}^{\text{priori}}\boldsymbol{C}^T + \boldsymbol{R})^{-1} \tag{5.5a}$$

$$\hat{\boldsymbol{\xi}}_{k+1}^{\text{post}} = \hat{\boldsymbol{\xi}}_{k+1}^{\text{priori}} + \mu_k\boldsymbol{K}(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{\xi}}_{k+1}^{\text{priori}}) \tag{5.5b}$$

$$\boldsymbol{P}_{k+1}^{\text{post}} = \boldsymbol{P}_{k+1}^{\text{priori}} - \mu_k\boldsymbol{K}\boldsymbol{C}\boldsymbol{P}_{k+1}^{\text{priori}} \tag{5.5c}$$

where

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix},$$

and $\boldsymbol{K}$ is the Kalman gain used to weight the new measurement. The variable μ_k is the same as in equation (5.3). Notice that, if μ_k is zero, i.e. the objects are not within the FOV, the update step does not change the state and covariance matrix.

5.4 Probability Map

To keep track of the movements of the UAV in the open area, we use a probability map. Another terminology for probability map is occupancy grid. This enable us

to calculate the most likely location of undiscovered objects. As with the Kalman filters, we assume the number of objects in the area is known.

First, we divide the open area into a grid, and the cell size is selected to be of a comparable size to the FOV of the UAV (in Section 5.7, we use a circular FOV and a cell side length equal to the FOV radius). The setup is illustrated in Figure 5.3.

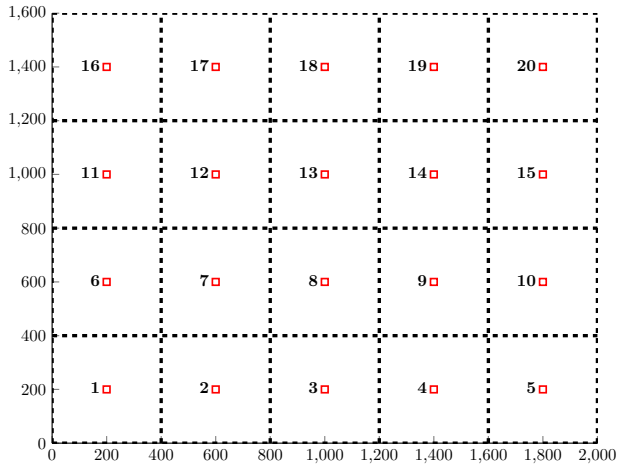


Figure 5.3: Illustration of area divided into a grid. The cells are numbered one to twenty and marked with a red square.

We use probability to describe the most likely location of objects in the area. We define $p_i(t)$ to be the probability of finding at least one unknown object within cell i. This probability will be conditional to the UAV's location, which makes it time-varying as the UAV navigates.

The probability $p_i(t)$ will not only depend on the UAV's proximity, but also on its absence. We introduce a new function called the default probability, $p_0(t)$, that models *the probability of finding an unknown object in a cell in the absence of any observation assuming each object has equal probability of being located in all cells.*. Letting n_{cell} be the number of cells and n_{unknown} be the number of unknown objects, we propose to model the default probability as

$$p_0(t) = 1 - \left(\frac{n_{\text{cell}} - 1}{n_{\text{cell}}}\right)^{n_{\text{unknown}}(t)} \tag{5.6}$$

Notice that we use the assumption that we know the number of unknown objects. In practice, we can either guess or use an estimate for this number. We can, for example, apply this to the area in Figure 5.3, in which there are 20 cells. If we assume that there are five unknown objects in the area, then $p_0 = 1 - (\frac{19}{20})^5 =$

22.62%. As objects are found, the default probability will decrease stepwise. If two of the five objects are found, $p_0 = 14.26\%$.

When the UAV is absent from a cell, $p_i(t)$ is not only affected by $p_0(t)$, but also by its adjacent cells' probabilities. This is best illustrated through an example. When considering Figure 5.3, say that the UAV has made observations in cells 1-15 in such a way that their probabilities are 1%, and let the default probabilities in cells 16-20 all be $p_0(t) = 14.26\%$. If the UAV stops observing, then, after some time, all cells will have a probability equal to $p_0(t)$. However, the cells adjacent to cells 16-20 should approach this probability faster.

We suggest the following function to model the probability of each cell

$$\dot{p}_i(t) = -e^{-k_0 d_i^2} p_i(t) + q g_i(t) \tag{5.7}$$

where

$$g_i(t) = \begin{cases} p_0(t) - p_i(t) & \text{if } p_{\text{avg,i}} < p_i \\ p_0(t) + p_{\text{avg,i}}(t) - 2p_i(t) & \text{else} \end{cases}$$

$$d_i = |\text{uav}_{\text{pos}} - \text{cell}_{\text{pos,i}}|, \quad p_i(0) = p_0, \quad \forall i \in [1, n_{\text{cell}}]$$

The distance between cell i and the UAV is denoted d_i. The positive constants k_0 and q are tunable, and we suggest appropriate values in Section 5.6.6. Finally, $p_{\text{avg,i}}$ is the average probability value of the cells adjacent to cell i. Notice that this differential equation makes $p_i(t)$ always stay within the range $(0, 1)$. When the UAV is observing cell i, the first term dominates and gives fast convergence to zero probability, while, when the UAV is absent, the second term dominates, giving slow convergence to $p_0(t)$, where the convergence rate is influenced by its neighbors.

5.5 Necessary Visitation Period

When a new object is detected, the accuracy of its position estimate will be determined by the process and measurement noise quantified by the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. Given that the sensor and model are reasonably accurate, the estimation error will rapidly converge to a small number if the object is observed by the UAV. However, if the UAV continues to follow the newly discovered object, it will be unable to track previous objects and discover new ones. The state estimates will deteriorate without new measurements, and it is desirable to quantify this deterioration, such that we can select the timespan before the next revisit based on a probability for re-detection of the new object by the UAV.

Given a tracked object and assuming that we want a 90% probability that the estimation position error is less the UAV's FOV, we will (with high probability) re-detect

the object if we measure the object's estimated position. This 90% detection probability corresponds to a specific time. If we measure the object's estimated position before this time, it will have a higher detection probability. The aim of this section is to calculate the time-period of high probability of detection, which we call the necessary visitation period:

Definition 4 (Necessary Visitation Period, NVP)**.** *Given a probability measure, an object, and a sensor, the necessary visitation period (NVP) is the maximal amount of time between two position measurements of the object such that the estimation error in probability is less than detection the range of the sensor.*

The derivation of the NVP has two main steps. First, we propagate the covariance matrix in the absence of observation, see equation (5.4b). Then, we note that the square of the estimated error distance is the sum of two squared normally distributed random variables, making it a second-order chi-squared distribution (Simon 2006). Here, we assume that the x- and y- directions are independent.

Let the true state of an object at timestep k be $\boldsymbol{\xi}_k$. Then, we can define the estimation error

$$\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k - \hat{\boldsymbol{\xi}}_k \tag{5.8}$$

The state consists of the position and velocity such that $\tilde{\boldsymbol{\xi}}_k = \begin{bmatrix} \tilde{\boldsymbol{s}}_k & \tilde{\boldsymbol{v}}_k \end{bmatrix}$. The position error at n timesteps into the future is $\tilde{\boldsymbol{s}}_n$. We can then write the NVP, t_{NVP}, as:

$$\begin{aligned} & t_{\text{NVP}} = \max_n (n\Delta_t) \\ & \text{s.t.} \\ & p(|\tilde{\boldsymbol{s}}_n| \leq \text{FOV}_{radius}) \leq p_{\text{FOV}} \end{aligned} \tag{5.9}$$

where Δ_t is the timestep used in the discretization of the object equations. The detection range of the sensor is FOV_{radius}. The probability p_{FOV} can be set to any value in the interval $p_{\text{FOV}} \in (0, 1)$.

The covariance matrix at timestep k and the process noise, $\boldsymbol{Q}$, are both 4×4

matrices, which we write

$$\boldsymbol{P}_k = \begin{bmatrix} \sigma_{11,k}^2 & \sigma_{12,k} & \sigma_{13,k} & \sigma_{14,k} \\ \sigma_{12,k} & \sigma_{22,k}^2 & \sigma_{23,k} & \sigma_{24,k} \\ \sigma_{13,k} & \sigma_{23,k} & \sigma_{33,k}^2 & \sigma_{34,k} \\ \sigma_{14,k} & \sigma_{24,k} & \sigma_{34,k} & \sigma_{44,k}^2 \end{bmatrix} \tag{5.10}$$

$$\boldsymbol{Q}_k = \Delta_t \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ q_{14} & q_{24} & q_{34} & q_{44} \end{bmatrix} \tag{5.11}$$

where $\boldsymbol{Q}_k$ is the discretized version of the $\boldsymbol{Q}$ matrix from equation (5.1). Notice that both matrices are symmetric.

We can now derive the following result (which is an extension of Theorem 1 from Albert and Imsland (2017)).

Theorem 1 (Necessary Visitation Period)**.** *Given an object characterized by equation* (5.1) *and a Kalman filter for estimating the state of the object given by equations* (5.4) *and* (5.5)*. Let* Δ_t *be the timestep of the filter and* FOV_{radius} *be the range of the sensor. If* χ_2^2 *is the p-value for a chi-squared distribution with two degrees of freedom, then the two equations*

$$\varphi(n; \boldsymbol{\lambda}_a) = 0 \tag{5.12a}$$

$$\varphi(n; \boldsymbol{\lambda}_b) = 0 \tag{5.12b}$$

where:

$$\varphi(n; \boldsymbol{\lambda}) = \frac{1}{3}\lambda_1 \Delta_t^2 n^3 + \left[(-\frac{1}{2}\lambda_1 + \lambda_4)\Delta_t^2 + \lambda_2 \Delta_t\right] n^2$$
$$+ \left[\frac{1}{6}\lambda_1 \Delta_t^2 + (-\lambda_2 + 2\lambda_5)\Delta_t + \lambda_3\right] n + (\lambda_6 - \frac{FOV_{radius}^2}{\chi_2^2})$$
$$\boldsymbol{\lambda}_a = [q_{33}, q_{13}, q_{11}, \sigma_{33,k}^2, \sigma_{13,k}, \sigma_{11,k}]$$
$$\boldsymbol{\lambda}_b = [q_{44}, q_{24}, q_{22}, \sigma_{44,k}^2, \sigma_{24,k}, \sigma_{22,k}]$$

will each have exactly one real solution, which we denote n_1 *and* n_2*, respectively. The parameters in the set* $\boldsymbol{\lambda}_{a,b}$ *comes from the covariance matrix,* $\boldsymbol{P}_k$*, and the process noise,* $\boldsymbol{Q}_k$*, of equations* (5.10) *and* (5.11) *with initial condition given by* (5.4b)*.*

Then, if the sensor takes a measurement at the estimated position of the object

given by equation (5.4a) at the time

$$t_{nvp} = \min(n_1, n_2)\Delta_t, \tag{5.13}$$

it will at least have a probability of measuring the real position of the object within the confidence interval given by the p-value of the χ^2_2 distribution.

As an example of the use of this result, consider a Kalman filter estimating the state of an object with the covariance matrices of the estimate, $\boldsymbol{P}_k$, and the process noise, $\boldsymbol{Q}_k$, and timestep Δ_t. Then, if we which to have a 95% chance of measuring the object at the estimated position after t_{nvp} seconds, we must select $\chi^2_2 = 5.99$.

Proof. Let the covariance matrix of a state estimate of an object using a Kalman filter be $\boldsymbol{P}_k$. Consider the timestep from k to $k+n$ in which we do not receive any measurements, i.e. $\mu_{i,k} = \mu_{i,k+1} = \cdots = \mu_{i,k+n} = 0$. Here, n is the number of timesteps into the future from k. Then, the covariance is completely determined by equation (5.4b), which we apply recursively to arrive at

$$\boldsymbol{P}_{k+n} = \boldsymbol{A}^n \boldsymbol{P}_k (\boldsymbol{A}^n)^T + \sum_{i=0}^{n-1} \boldsymbol{A}^i \boldsymbol{Q} (\boldsymbol{A}^i)^T \tag{5.14}$$

where

$$\boldsymbol{A}^n = \begin{bmatrix} \boldsymbol{I} & n\Delta_t \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}.$$

Furthermore, following the notation from equation (5.10), we calculate the upper corner of the matrix $\boldsymbol{P}_{k+n}$

$$\begin{aligned} \sigma^2_{11,k+n} &= \sigma^2_{11,k} + 2n\Delta_t \sigma_{13,k} + (n\Delta_t)\sigma_{33,k} \\ &+ \sum_{i=0}^{n-1} [q_{11} + 2i\Delta_t q_{13} + (i\Delta_t)^2 q_{33}] \end{aligned} \tag{5.15}$$

We can now apply summation formulas to remove the sum from the second part of this equation.

$$\begin{aligned} \sigma^2_{11,k+n} &= \sigma^2_{11,k} + 2n\Delta_t \sigma_{13,k} + (n\Delta_t)^2 \sigma_{33,k} \\ &+ \frac{1}{3} q_{33} \Delta_t^2 n^3 + (-\frac{1}{2} q_{33} \Delta_t^2 + q_{13}\Delta_t) n^2 \\ &+ (\frac{1}{6} q_{33} \Delta_t^2 - q_{13}\Delta_t + q_{11}) n \end{aligned} \tag{5.16}$$

Next, we can follow the same approach to calculate the variance in the y-direction.

$$\sigma_{22,k+n} = f(\sigma_{22,k}, \sigma_{24,k}, \sigma_{44,k}, q_{44}, q_{24}, q_{22}, \Delta_t) \tag{5.17}$$

The detection range for the sensor onboard the UAV is $\text{FOV}_{\text{radius}}$. We need to keep the position estimate, $\tilde{\boldsymbol{s}}_{k+n}$, less than the range of the sensor

$$|\text{FOV}_{\text{radius}}| \geq |\tilde{\boldsymbol{s}}_{k+n}| \tag{5.18}$$

$$\text{FOV}^2_{\text{radius}} \geq \tilde{x}^2_{k+n} + \tilde{y}^2_{k+n}. \tag{5.19}$$

The variance of error in both directions is given by (5.16). If we let $\sigma^2_{\text{pos},k+n} = \max(\sigma^2_{11,k+n}, \sigma^2_{22,k+n})$ and divide both sides of equation (5.19) by it, we get

$$\frac{\text{FOV}^2_{\text{radius}}}{\sigma^2_{\text{pos},k+n}} \geq \frac{\tilde{x}^2_{k+n}}{\sigma^2_{\text{pos},k+n}} + \frac{\tilde{y}^2_{k+n}}{\sigma^2_{\text{pos},k+n}} \tag{5.20}$$

Now, we have two normally distributed random variables squared, one with variance value one and the other with variance value less than one. This will be less than a chi-squared distribution of second order. Let χ^2_2 be the p-value of a given confidence interval for a chi-squared distribution. Then, we obtain

$$\chi^2_2 \geq \frac{\tilde{x}^2_{k+n}}{\sigma^2_{\text{pos},k+n}} + \frac{\tilde{y}^2_{k+n}}{\sigma^2_{\text{pos},k+n}} \tag{5.21}$$

Finally, we can combine this equation with (5.20) to get an expression for the maximum variance

$$\sigma^2_{\text{pos},k+n} = \frac{\text{FOV}_{\text{radius}}}{\chi^2_2} \tag{5.22}$$

Combine equations (5.16) and (5.17) with (5.22) to arrive at (5.12). If we use the minimum of the solutions to the two equations, we are guaranteed that equation (5.21) will hold.

Furthermore, both equations (5.12) are cubic functions in n. If we let α, β, γ and δ be the constants defining each of these functions, then $\alpha n^3 + \beta n^2 + \gamma n + \delta = 0$. Thus, the discriminant is given by

$$\Delta = 18\alpha\beta\gamma\delta - 4\beta^3\delta + \beta^2\gamma^2 - 4\alpha\gamma^3 - 27\alpha^2\delta^2 \tag{5.23}$$

When the discriminant is negative, the cubic function will have only one real and two complex conjugated solutions (Irving 2003). Since we must choose all elements of $\boldsymbol{Q}_k > 0$ and $\Delta_t > 0$, and we use initial conditions given by equation (5.4b), the discriminant is always negative, and thus the cubic functions in equation (5.12) will each have only one real solution, n_1 and n_2, respectively. □

5.6 Search and Track Algorithm

In this section, we present the path planning algorithm from Figure 5.2, which is a Search and Track (SaT) algorithm for the UAV. First, it utilizes Theorem 1 to select positions for the tracked objects. Then, it uses combinatorial optimization to find a globally optimal visitation sequence. This solution is then used as an initial condition for an optimal control problem, which produces a continuous path that is feasible w.r.t. UAV dynamics and constraints.

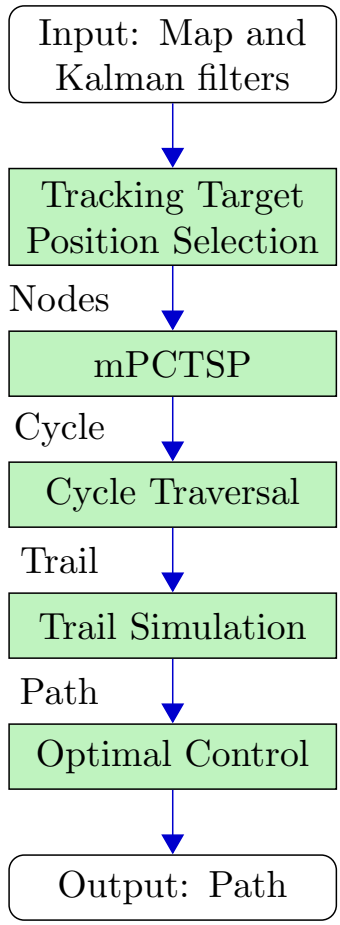


Figure 5.4: Flowchart for the search and track algorithm.

Figure 5.4 illustrates the main components of this algorithm. As input, we have the Kalman filters and probability map. In the first step, we use the Kalman filters to select a position for each tracked object by applying Theorem 1. We call this set of positions for objects nodes, and the cells from the probability map cell nodes. Both

sets of nodes are used as input for the modified prize collection TSP (mPCTSP) algorithm, which produces a cycle. Here, we define a cycle as *an ordered set of nodes starting and stopping at the same node*. The mPCTSP produces a cycle, instead of a trail, to enable it to run an optimization of the UAV's traversal of the cycle, such that the traversal closest to fulfilling the assumed object node positions is chosen. A trail is *an ordered set of nodes starting from the UAV*. The Runge-Kutta method uses the trail to simulate the differential equations from Sections 5.3 and 5.4. Finally, the Runge-Kutta simulation is used as an initial state for an optimal control problem (OCP). Both the Runge-Kutta method and OCP produce a path for the UAV. We define a path as *an ordered set of positions, which takes the nonholnomoic constraint of the UAV into account*. In other words, a path is a flyable set of waypoints for the UAV. Figure 5.6 shows an example of the different algorithm steps.

5.6.1 Tracking Objects Position Selection

The mPCTSP algorithm will calculate an optimal visitation sequence, assuming that the tracked objects are stationary. For this, we need to assign positions to the objects. Theorem 1 enable us to calculate the timespan, t_{nvp}, that each object can be absent measurements given a probability for re-detection. We select a probability p_{ideal} (see Section 5.6.6) and use Theorem 1 to calculate t_{nvp} for each tracked object. The results are stored in the vector $\boldsymbol{t}_{\text{ideal}} \in \mathbb{R}^{n_{\text{object}}}$, such that t_{nvp} for object i is $t_{\text{ideal}}(i)$. The number of tracked objects is n_{object}. When running the mPCTSP algorithm, we assume that we will find a solution such that the UAV can measure each object's position at time $\boldsymbol{t}_{\text{ideal}}$. This assumption will not be feasible, in general, since it requires the UAV to visit each tracked object at a specific time. However, when doing the cycle traversal step (Section 5.6.3), we adjust $\boldsymbol{t}_{\text{ideal}}$ to suit the UAV's traversal. To illustrate, if the estimated position and velocity of object i are $(400, 500)$m and $[1, 1]$m/s, respectively, at the time we run the SaT algorithm, and we selected $p_{\text{ideal}} = 85\%$ and apply Theorem 1 to get $t_{\text{ideal}}(i) = 100$, then, the tracked object position is $(400, 500) \times 100[1, 1] = (500, 600)$m for the mPCTSP algorithm. If, after selecting a cycle in the cycle traversal step, the UAV is estimated to visit object i at 90s, then the position is adjusted to $(400, 500) \times 90[1, 1] = (490, 590)$m for the trail simulation and OCP problem. Note that this also leads to an increase in the probability of re-detection for object i, since the UAV is visiting it earlier than planned.

5.6.2 Modified Prize Collection TSP (mPCTSP)

The modified Prize Collection TSP (mPCTSP) is a combinatorial optimization problem, in which we use the following formulation (modified from Chen et al.

(2010)).

$$\max \sum_{i=n_{\text{object}}+1}^{n} x(i)p(i) \tag{5.24a}$$

s.t.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} y(i,j)d(i,j) \leq d_{\max} \tag{5.24b}$$

$$\sum_{j=1}^{n} y(j,i) = 2, \ \sum_{j=1}^{n} y(i,j) = 2 \quad i \in [1, n_{\text{object}}] \tag{5.24c}$$

$$\sum_{j=1}^{n} y(j,i) = 2x(i), \ \sum_{j=1}^{n} y(i,j) = 2x(i)$$

$$i \in [n_{\text{object}} + 1, n] \tag{5.24d}$$

$$\sum_{i,j \in \boldsymbol{E}_s} y(i,j) = |\boldsymbol{S}| - 1 \quad \forall \quad |\boldsymbol{S}| = 2, 3, ..., n_{\text{cycle}} - 2 \tag{5.24e}$$

where $n = n_{\text{object}} + n_{\text{cell}}$ is the number of nodes. The optimization variables are the binary vector $\boldsymbol{x} \in \mathbb{R}^{1 \times n_{\text{cell}}}$ and the binary matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$. The vector, $\boldsymbol{x}$, represents the cell nodes. Element $x(i)$ is one if the node is included in the cycle and zero if it is not. The matrix $\boldsymbol{Y}$ represents the arcs between the nodes. An arc is included between node i and j if element $y(i,j)$ is one and not included if $y(i,j)$ is zero. The vector, $\boldsymbol{p}$, contains the most recent values for the probability of detection of each cell node with the probability of each element given as $p(i) = p_i(t)$ from equation (5.7). The matrix $\boldsymbol{D}$ contains the distances between the nodes, with a single element given as $d(i,j)$. The maximum distance the UAV can travel is set by the constant $d_{\max}$ (see Section 5.6.6 for details on how to select this constant). Finally, $\boldsymbol{E}_s$ is the set of all proper subsets, $\boldsymbol{S}$ is a proper subset, and n_{cycle} is the length of the cycle.

The objective function, (5.24a), maximizes the probability of detection of objects from the open area. The constraint, (5.24b), makes sure the cycle is less than $d_{\max}$. In a standard PCTSP problem, all nodes would be optional to visit. Our modification is that the object nodes are not. We do this by requiring that these nodes are visited exactly once (5.24c), while cell nodes must be visited only if they are in the cycle (5.24d). The last constraint, (5.24e), is a subcycle elimination. See Chen et al. (2010), Chapter 6.5, for details.

A special case is made when we have only one object available. Then, instead of finding a cycle, which does not consider the UAV's starting position, the mPCTSP

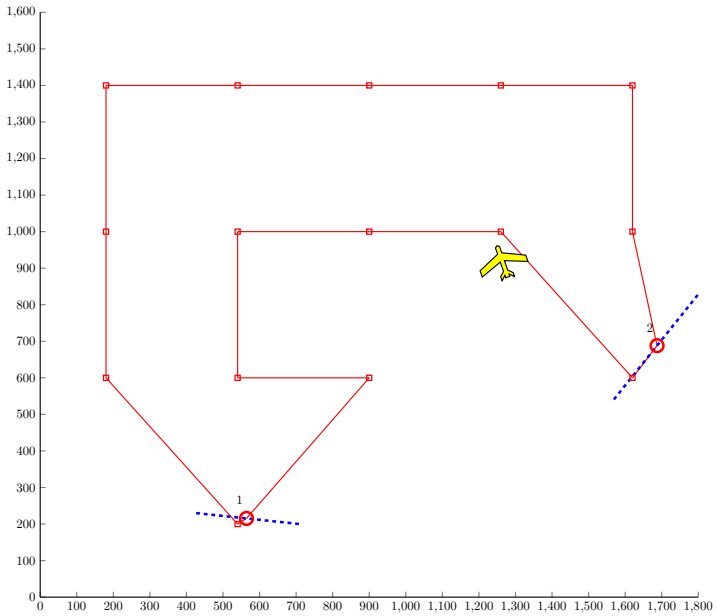


Figure 5.5: Typical resulting cycle from mPCTSP algorithm. The cell nodes are drawn as red squares, while the object node position at $\boldsymbol{t}_{\text{ideal}}$ are drawn as red circles with a blue dotted line representing the time $\boldsymbol{t}_{\text{ideal}} \pm 60$seconds. The UAV is drawn as a yellow polygon.

finds a trail from the UAV's position to the object's position. This is achieved by replacing constraint (5.24c) with

$$\sum_{j=1}^{n} y(j,1) = 0, \sum_{j=1}^{n} y(1,j) = 1 \tag{5.25a}$$

$$\sum_{j=1}^{n} y(j,n) = 1, \sum_{j=1}^{n} y(n,j) = 0 \tag{5.25b}$$

where we assume that node number 1 is the UAV starting position and node number n is the object position.

When we have only one object, we skip the cycle traversal optimization, and go straight to the trail simulation.

5.6.3 Cycle Traversal

A typical cycle from the mPCTSP algorithm is illustrated in Figure 5.5. Here, the object node positions at $\boldsymbol{t}_{\text{ideal}}$ are illustrated with red circles and number 1 and 2, while the cell nodes with red squares. In addition, the object nodes has a dotted blue line through them. This line illustrates the movement of the object

nodes $\boldsymbol{t}_{\text{ideal}} \pm 60$ seconds. Our goal is to find a traversal of the cycle, in which the objects are close to the position at $\boldsymbol{t}_{\text{ideal}}$ when the UAV pass them. For the case in Figure 5.5, we have $t_{\text{ideal}}(1) = 166s$ and $t_{\text{ideal}}(2) = 174s$. Let the start time of the traversal be $t_0 = 0$, then we would like the UAV to be at object node 1 at $t = 166s$, object node 2 at $t = 174s$, and traverse the cycle.

In general, it is not possible to find an exact solution to the above problem. However, we try to find a solution that is close by considering all possible traversals of the cycle, meaning starting with any node in the cycle and considering both directions. A traversal can be characterized by the time each node is visited by the UAV. Note that for the object nodes their position will depend on this time. Let n_{cycle} be the number of the nodes in a cycle and each traversal be numbered $j \in [1, 2n_{\text{cycle}}]$. Furthermore, let each node in a traversal be called an element, such that $t_1^j, t_2^j, \ldots t_{n_{\text{cycle}}}^j$ be the time each element is visited in traversal number j. Note that t_1^1 and t_1^2 reference the first element in traversal 1 and 2, which is not the same node unless both traversals starts at the same node.

We formulate a multi-layered optimization formulation to select traversal as follows

$$c_1 = \min_j \max(0, (|\Delta\psi_1^j| - \frac{\pi}{2})^2) \tag{5.26a}$$

$$c_2 = \min_j \max(0, (|\Delta\psi_2^j| - \frac{\pi}{2})^2) \tag{5.26b}$$

$$c_3 = \min_j \sum_{i=1}^{n_{\text{object}}} \left(t_{\text{ideal}}(i) - t_k^j \right)^2 \qquad \text{s.t. node}(k) = i \tag{5.26c}$$

where $\Delta\psi_1^j$ and $\Delta\psi_2^j$ are the *difference* between the UAV's heading and the angle between the UAV and the first and seconds elements of trail j. The time t_k^j is when the UAV visits element k in traversal j. We use a function node(k) to map from element k to corresponding node.

The first two layers fo the optimization formulation, equation (5.26a) and (5.26b), correspond to regularization. This is necessary to get continuity of the solution during multiple reruns of the SaT algorithm. The final layer, equation (5.26c), selects the traversal closest to fulfilling the UAV passing each node object at time $\boldsymbol{t}_{\text{ideal}}$.

To calculate when the UAV visits each element in a traversal we use the following iterative formula

$$a(t_{k+1}^j)^2 + bt_{k+1}^j + c = 0 \tag{5.27a}$$

where

$$a = |U|^2 - |\hat{\boldsymbol{\xi}}^v(\text{node}(k))|^2$$

$$b = -2(\begin{bmatrix}1 & 1\end{bmatrix} \hat{\boldsymbol{\xi}}^p(\text{node}(k)) - p_k^j \hat{\boldsymbol{\xi}}^v(\text{node}(k)))$$

$$c = -|\hat{\boldsymbol{\xi}}^p(\text{node}(k)) - p_k^j|^2$$

$$p_{k+1}^j = \hat{\boldsymbol{\xi}}^p(\text{node}(k)) + t_k^j \hat{\boldsymbol{\xi}}^v(\text{node}(k)) \tag{5.27b}$$

$$\forall k = [1, \ldots, n_{\text{cycle}}]$$

$$t_0^j = 0 \quad p_0^j = \text{UAV position} \tag{5.27c}$$

where U is the UAV's velocity. The velocity and position of element k are $\hat{\boldsymbol{\xi}}^p(\text{node}(k))$ and $\hat{\boldsymbol{\xi}}^v(\text{node}(k))$. The position of element k in traversal j is denoted p_k^j. Notice that we ignore the dynamics of the UAV, i.e. equation (5.29b). Also note that for all the cell nodes $\hat{\boldsymbol{\xi}}^v(\text{node}(k)) = [0, 0]$

The iterative formula is derived by considering the UAV and a element to be two moving points where we control the UAV without movement restriction and calculate their intersection.

5.6.4 Trail Simulation

To simulate a trail, we need an autopilot for the UAV. We use a P-controller, which calculates the desired heading towards the next node and adjusts the current heading appropriately. Let the next node have the position $(x_{\text{desired}}, y_{\text{desired}})$ and the UAV model be equation (5.29b), then

$$\begin{aligned} \psi_{\text{desired}} &= \text{atan2}(y_{\text{desired}} - y, x_{\text{desired}} - x) \\ u &= \psi_{\text{desired}} - \psi \end{aligned} \tag{5.28}$$

where atan2 is a four-quadrant arctangent function.

We use a Runge-Kutta method (Dormand and Prince 1980) for the simulation in which we simulate the UAV equation, (5.29b), and all the cell equations, (5.7).

5.6.5 Optimal Control Problem

In the last step of the SaT algorithm, we formulate an optimal control problem (OCP) to take the UAV's dynamics into account. In contrast to the trail simulation, the OCP considers not only the next node, but all nodes when creating the UAV path. This leads to a path better suited to the UAV than the trail simulation produced. The difference is best illustrated by comparing Figures 5.6(c) and 5.6(d).

To limit the size of the problem, we include only the nodes used by the trail simulation in Section 5.6.4. In addition, all nodes are treated as simplified cell nodes, meaning that we use equation (5.7) with $q = 0$ and initialize them with $p_0 = 1$. The positions of the object nodes are adjusted to the trail simulation. This gives us the following formulation

$$\min_{u(\cdot)} \int_0^{t_{\text{end}}} \mu \sum_{i=1}^{n_{\text{node,ocp}}} p_i^2(t) + u^2(t) dt \tag{5.29a}$$

s.t.

(5.7) with $q = 0$ and $p_0 = 1$

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} U\cos(\psi) \\ U\sin(\psi) \\ u \end{bmatrix} \tag{5.29b}$$

$$-u_{\text{lim}} \leq u \leq u_{\text{lim}} \tag{5.29c}$$

$$-\eta \leq x \leq X + \eta \tag{5.29d}$$

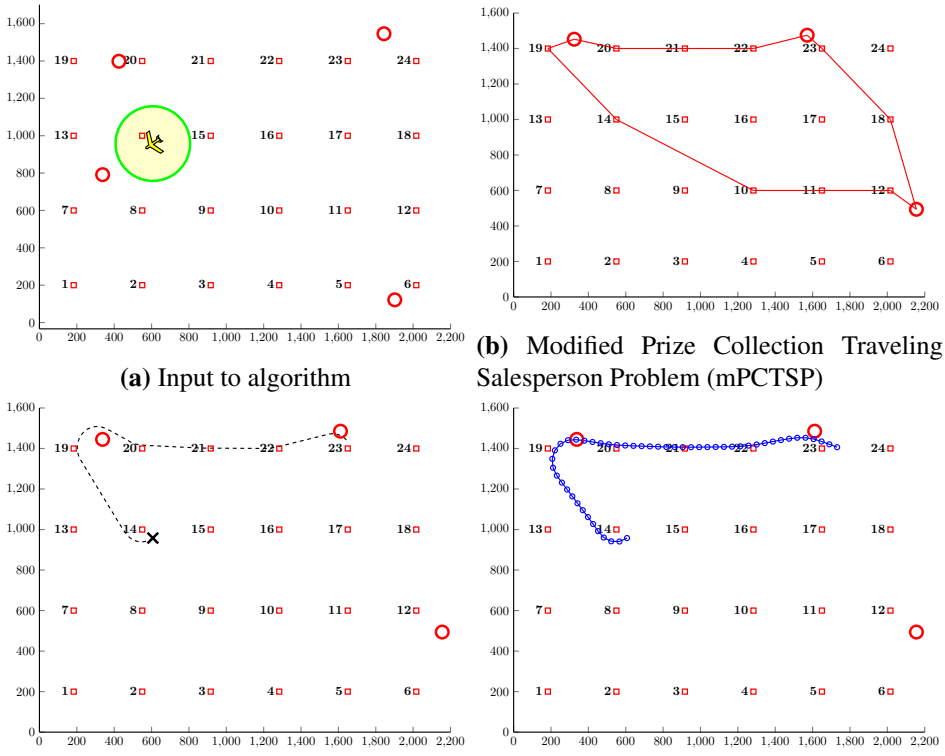
$$-\eta \leq y \leq Y + \eta \tag{5.29e}$$

Here, $p_i(t)$ is used to attract the UAV to cell i instead of representing any probability of detecting an object. It is limited in value to between 0 and 1. The time horizon is given by t_{end} and μ is a tunable constant used to weigh between the two objectives (see Section 5.6.6 for how to select an appropriate value). The number of nodes used by the OCP is $n_{\text{node,ocp}}$. Note that this is not necessarily the same number of nodes used in the mPCTSP problem in Section 5.6.2. The UAV state is $\boldsymbol{z}$, which consists of Cartesian coordinates, $(x, y)^T$, and heading, ψ. The velocity, U, is constant, while the turn rate, u, is bounded. The constraints (5.29d) and (5.29e) make sure that the UAV stays within the open area. Here, we assume the area to be a rectangle given by the coordinates X and Y, and η is a threshold which allows the UAV to move a slightly outside the area.

5.6.6 Tunable parameters

One of the strengths of our algorithm is that an operator does not have to weight some constant between searching and tracking. However, it is necessary to set some parameters. Fortunately, most of these can be set in relation to other parameters or suggested values work in most cases.

In terms of the the area model, there are two parameters that need to be set, k_0 and q. First, k_0's primarily function is within the optimal control problem. It should attract the UAV to the center of each cell, but avoid being set so large such that the



(a) Input to algorithm

(b) Modified Prize Collection Traveling Salesperson Problem (mPCTSP)

(c) Trail simulation for mPCTSP

(d) Optimal Control

Figure 5.6: The main steps of the search and track algorithm. Each cell is drawn as a red square with an appurtenant number. The estimated position of objects are red circles. The UAV is indicated by a yellow polygon, and its FOV is illustrated by a light yellow circle and green border. The mPCTSP solution is drawn with a solid red line, the Runge-Kutta simulation with a dotted black line, and the optimal control problem as a solid blue line with circles for each segment.

UAV does not have to reach the center to reduce the cell value. A value that works in most cases is

$$k_0 = 5 \times 10^{-4} m^{-2} \tag{5.30}$$

The other parameter, q, decides the rate at which any cell reaches the default probability in the absence of observation. It can be set in relation to the expected velocity of the moving objects. We use the following value

$$q = 0.01 \tag{5.31}$$

In preparing the Kalman filters for the mPCTSP algorithm, we need to set a probability of redetection for an object at its estimate position, which we will denote p_{ideal}. The path planner tries to satisfy this constraint. This probability must be set so that we do not visit the objects too often, which would lead to less searching, but not too high so as we would only search instead of also tracking the known objects. Given the performance measure in Section 5.7.1, we weight it equally between finding a new object and reducing the estimated error of a known object that is just about to get lost (estimated position error larger than the UAV's FOV). We use a heuristic to try to reflect this

$$p_{\text{ideal}} = \min\left(0.85, 1 - \frac{1}{n_{\text{cell}}} \sum_{i=0}^{n_{\text{cell}}} p(i)\right) \tag{5.32}$$

where $p(i)$ is cell i's probability of finding an object, and n_{cell} is the number of cells. Notice here that we use an upper threshold of 0.85 to avoid no searching at all in the case where most of the area is well-explored.

The distance limit for the mPCTSP is $d_{\max}$. We would like this constant to be as large as possible, while still making it possible for the resulting cycle to be successfully completed within t_{ideal} (see Section 5.6.1). We use the following heuristic to set it

$$d_{\max} = \begin{cases} \max(U t_{\text{ideal}}(1), |\text{UAV}_{\text{pos}} - \hat{\boldsymbol{\xi}}^p(1)|) & \text{if} \quad n_{\text{object}} = 1 \\ d_{\text{tsp}} + 1.5 \times U \max(0, c_{3,\text{avg}}(\text{TSP})) & \text{else} \end{cases} \tag{5.33}$$

where U is the velocity of the UAV, and its position is UAV_{pos} The position of object 1 at time $t_{\text{ideal}}(1)$ is $\hat{\boldsymbol{\xi}}^p(1)$. The TSP solution distance for the tracked objects using their position at timestamp t_{ideal} is d_{tsp}. Finally, $c_{3,\text{avg}}(\text{TSP})$ is the traverse cost from equation (5.26c) for the TSP solution of the objects divided by the number of objects. Notice that this heuristic for the setting $d_{\max}$ always make the mPCTSP feasible.

The last tuning constant is μ which is used to weight between attracting the UAV to each cell and actuator use in the optimal control problem from Section 5.6.5. Our main objective is to reduce the probability of all the cells, but it is necessary to limit actuator use to obtain a practical solution for the UAV. The following heuristic works well

$$\mu = \frac{6}{n_{\text{cell}}} \tag{5.34}$$

5.6.7 Implementation

The algorithm is implemented in a receding horizon fashion. That is, an objective function over a finite horizon produces a sequence of actuator inputs. Then, we apply the first part of the sequence before rerunning the optimization. There are two ways to decide when to rerun the optimization. The first is sample-based. In this case, the SaT algorithm is rerun at regular intervals that are less than the time horizon of the optimal control problem. The second is event-based. An event can be when a new object is discovered or has just moved outside the FOV. Even with an event-based approach, it is necessary to decide a maximum time length before rerunning the optimization, which, again, must be less than the time horizon of the optimal control problem. In the simulation, we use the sample-based approach.

The software used to implement the algorithm and simulation was Matlab R2015a. The mPCTSP formulation from Section 5.6.2 was written using YALMIP (Löfberg 2004) and solved using IBM's CPLEX (IBM 2015). To simulate the differential equations from section 5.6.4, we used the Matlab integrated method ode45 (Shampine and Reichelt 1997). For the optimal control problem from Section 5.6.5, we wrote the formulation using CasADi (Andersson 2013b) and solved it using interior point solver IPOPT (Wächter and Biegler 2006) with the linear solver mumps.

5.7 Simulation

To validate the algorithm from Section 5.6, we compare it to multiple base cases and an omniscient case which utilizes the actual position and velocity of each object.

5.7.1 Simulation Scenario

To compare the new algorithm to the other algorithms, we run Monte Carlo simulations of the following scenario: We have an open area defined by its X- and Y-coordinates (a rectangle with corners (0,0), (X,0), (X,Y), (0,Y)), a given number of objects moving according to equation (5.1), and an available UAV with a limited FOV to search and track the objects. This is illustrated in Figure 5.1.

In the simulation, there are two parameters we vary: the number of objects and size of the area. To simplify size change, we keep the y-dimension fixed and only vary the x-dimension.

A practical problem with simulating this scenario is that the moving objects will not stay within the defined area throughout the simulation, which lead to no objects for the UAV to monitor. To adjust for this, we give the state space a cylindrical topology. Another term for this concept is periodic boundary. This means that we let the objects behave as the snake from the popular arcade game (Punyawee et al. 2016), meaning that when an object leaves the area on one side, it reappears on the other side. This lead to another practical problem for the UAV. If a Kalman filter estimates an object to be close to the edge of the area, but the object is on the other side, the UAV will miss detection of object, even if the estimate is only slightly off. To compensate for this, we let the UAV's FOV go across the edge of the area. Both the snake property of the objects and the UAV sensing capabilities are illustrated in Figure 5.7.

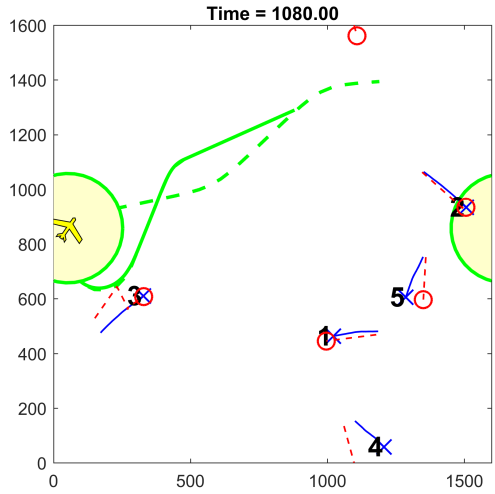


Figure 5.7: Monitoring a 1600 x 1600 m^2 area. The UAV is currently sensing object number 2 across the edge of the area. The estimated position of object 4, red circle, has moved across from one side of the area to the other.

Finally, we need a performance measure to compare the different algorithms. We suggest a simple binary measure in which each object is either observed or not. Observed is defined as *the object estimated position error being less than the FOV detection range of the UAV*. Then, we discretize the simulation and, for each step, count the number of observed objects. This lead to the following performance

measure

$$H = \frac{1}{n_{\text{object}} n_{\text{sim}}} \sum_{i=1}^{n_{\text{object}}} \sum_{k=1}^{n_{\text{sim}}} h_i(k) \tag{5.35a}$$

where

$$h_i(k) = \begin{cases} 1 & \tilde{\boldsymbol{s}}_i(k) \in \text{FOV} \\ 0 & \text{else} \end{cases} \tag{5.35b}$$

where $h_i(k)$ is a binary function returning 1 if object i's position error is within the FOV range at timestep k, and 0 if it is not. The number of simulation steps is $n_{\text{sim}} = \frac{T}{\Delta_t}$, where T is the simulation length. The estimation error of the position of object i at timestep k is $\tilde{\boldsymbol{s}}_i(k)$. Notice that we do not include any specific tracking or searching performance measure since our concern is having an estimate for each object within FOV detection range. The lack of an estimate and an estimate outside the FOV are considered equal. Furthermore, notice that this score is always between zero and 1, $H \in [0, 1]$, where 0 indicates no objects found and 1 indicates all objects have a position estimate within FOV detection range for the entire simulation.

5.7.2 Base Cases

The first base case sets the UAV to follow a simple straight line patrol, as illustrated in Figure 5.8. This does not take into consideration any of the estimates of the object positions. To allow the turns to be flyable for the UAV, we utilize the optimal control algorithm from Section 5.6.5.

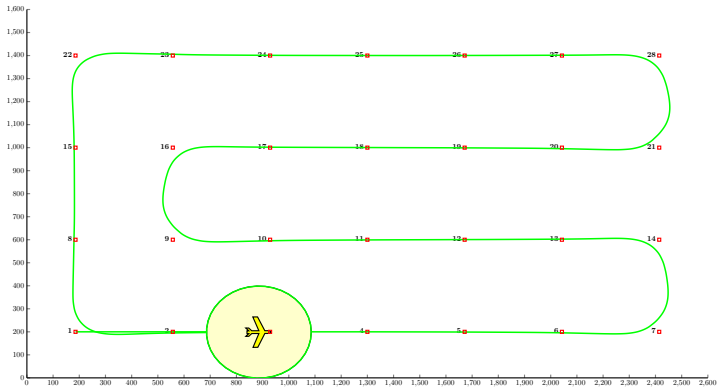


Figure 5.8: Base case 1. Straight line patrol of an example area of size 2600×1600 m^2.

In the second base case, we have the UAV following a looping pattern, which is shown in Figure 5.9. This performs slightly better than the straight line patrol.

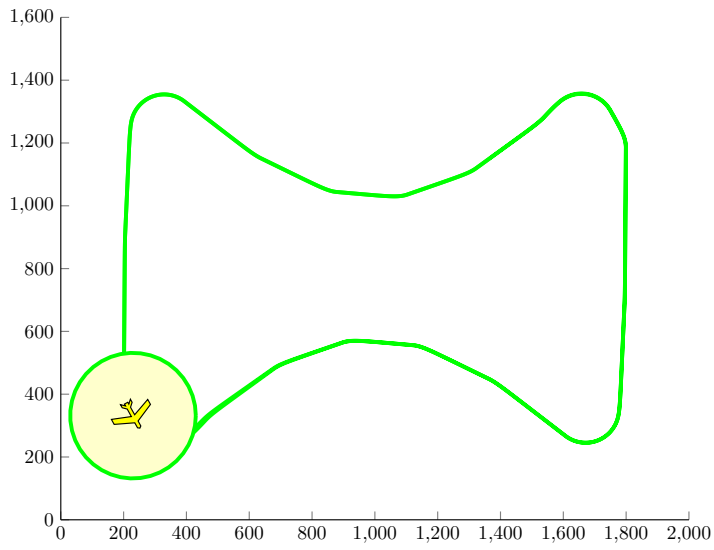


Figure 5.9: Base case 2. Looping patrol of an example area of size 2000×1600 m^2.

Finally, we use a case in which the UAV uses random behavior. This is implemented by setting the UAV to fly towards a random waypoint within the search area. The UAV changes the waypoint either when it reaches it or has attempted to reach it for 50 seconds. The value of 50 seconds was obtained experimentally by comparing multiple values and selecting the one with the best performance.

5.7.3 Best Case

To have a best case for comparison, we introduce an algorithm that utilizes knowledge of the actual positions of the objects. The algorithm is not optimal since it does not consider the nonholonomic constraint of the UAV.

Let $\boldsymbol{t}_{\text{last}} \in \mathbb{R}^{1 \times n_{\text{object}}}$ be a vector containing the timepoint when an object's error in estimate exceeds the FOV of the UAV. If an object does not have an estimate, it is simply set to zero. Then, the algorithm minimizes the following two layered optimization functions

$$A) \quad \min_{j} \sum_{i=1}^{n_{\text{trail}}} (t_{\text{last}}(i) - \text{trail}_t^j(i))^2 \tag{5.36a}$$

$$B) \quad \min_{\text{trail}(\cdot)} \text{Tour Length} \tag{5.36b}$$

The algorithm works by trying all possible permutations of the object visitation sequence calculated using equation (5.27). If more than one sequence has the

same value for objective A, the shortest trail is selected (objective B).

5.7.4 Results

The parameters used in the simulation are given in Table 5.1. Note that both the measurement and process noise, R and Q, are set based on trail and error to make the objects difficult to find and track. The SaT algorithm and perfect information algorithm were implemented using sample-based optimization, in which horizon $t_{\text{end}} = 100$s and the optimization was rerun every $dt_{\text{sim}} = 40$s. For the autopilot, we used a-line-of sight algorithm from Chapter 10 of Fossen (2011).

Figure 5.10 shows the results of running 50 simulations for seven different map sizes and five moving objects. The search and track algorithm scored, on average, about 5-10 % better than the base cases for the larger area. Figure 5.11 illustrates the results of 30 simulations for a constant map size (2000m $\times$ 1600m) while varying the number of objects from two to ten. Depending on the number of objects, the SaT algorithm score was between 5 - 15% better than the base cases.

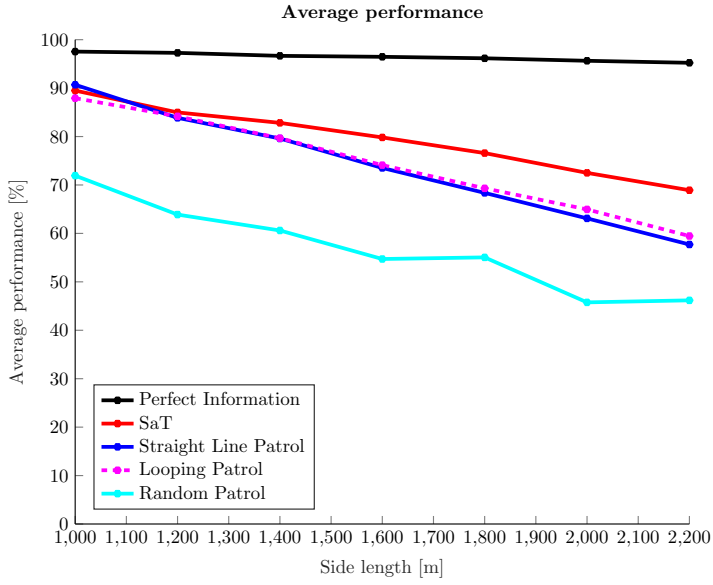


Figure 5.10: Comparison of SaT algorithm to base cases and the case of utilizing perfect information for areas of varying size using 5 objects.

5.8 Discussion

A strength of the proposed SaT algorithm is that little tuning is necessary for the operator. There is no objective function in which the weight of an artificial constant

Table 5.1: Simulation Parameters

Parameter	Value [unit]
UAV	1 unit
(x_0,y_0,heading)	(150,200,0)[(m,m,rad)]
Minimum turning radius	105.8 [m]
FOV$_{radius}$	200 [m]
Velocity	22 [m/s]
Objects	5 units
v_x	[-3,3] [m/s]
v_y	[-3,3] [m/s]
Observer	
Measurement period, ΔT	0.1 [s]
Process noise variance, Q	$10^{-4} \times \begin{bmatrix} 10 & 1 & 1 & 1 \\ 1 & 10 & 1 & 1 \\ 1 & 1 & 50 & 1 \\ 1 & 1 & 1 & 50 \end{bmatrix}$ [m/s^2]
Measurement noise variance, R	$\begin{bmatrix} 5 & 2.5 \\ 2.5 & 5 \end{bmatrix}$ [m^2]
Simulations	50
Simulation length, T	1800 [s]
Area width, Y	1600 [m]
Area length, X	[1000,2200] [m]
Algorithms	
k_0	5×10^{-4} [-]
q	0.01 [-]
η	150 [m]
t_{end}	100 [s]
dt_{sim}	40 [s]

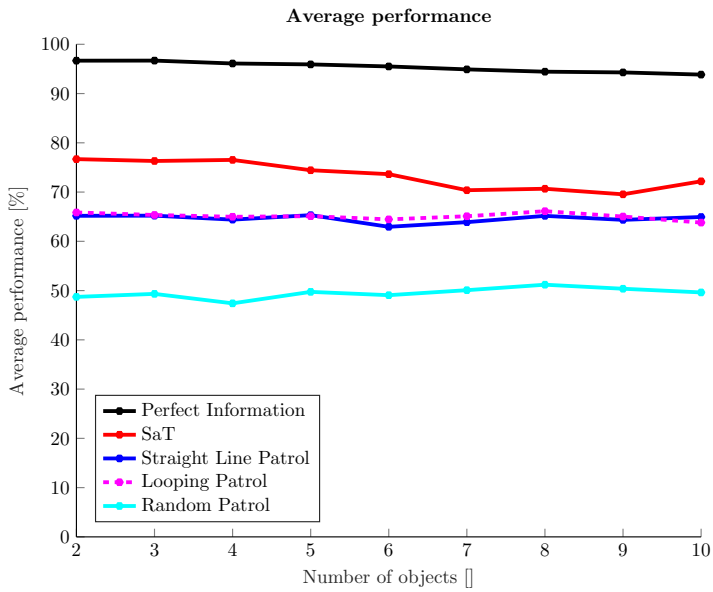


Figure 5.11: Comparison of SaT algorithm to base cases and utilizing perfect information for varying numbers of objects using an area of size 2000×1600 m^2.

decides the trade-off between searching and tracking. Instead, we have a more intuitive constant p_{ideal} which enable us to weight the probability between losing an object and finding a new one. The suggested heuristic in equation (5.32) from Section 5.6.6 makes it unnecessarily to do this trade-off manually.

Notice that we have assumed that the number of objects in the search area is known. This is utilized both when we know the number of necessary Kalman filters as well as in the probability map, see Section 5.4. This is to simplify and limit the scope of this article. In practice, this assumption is not crucial. It is not difficult to add/remove Kalman filters as objects are discovered or move out of the region, and with regards to the probability map, a reasonable estimate works well. For example, an estimate can be formulated based on historical data for iceberg searching or in a rescue operation where it is not unlikely that we know the number of people missing.

Another strength with the suggested SaT algorithm is its ability to combine combinatorial optimization and OCP. A weakness with OCP is the lack of global properties. The solution will always depend on the chosen time horizon. However, combinatorial optimization has global properties, but is unable to account for the nonlinear dynamic of the UAV. By using a combinatorial optimization to initialize the OCP problem, we achieve a globally optimal solution while considering the

nonholonomic constraints of the UAV. Some might argue, based on Figure 5.6, that the OCP does not improve the solution much. However, it always improves the solution, and the computational time to run it is short compared to the time needed to run the combinatorial portion.

The performance of the SaT algorithm is not much better than that of the base cases for a small area. However, as the area grows, its benefits increase. The reason for this is that the base cases perform close to the perfect information case for small areas, meaning that they are close to optimal and there is not much room for improvement. Therefore, for small areas, the added complexity of the SaT algorithm does not pay off and either the straight line or looping patrol are considered sufficient.

When varying the area size and the number of objects, there are two observable trends. Generally, a larger area and more objects lowers the performance score of all algorithms. However, the area size has a larger effect than the number of objects. Furthermore, the difference between the SaT algorithm and the base cases increases with the area size. The number of objects decreases the difference, but the effect is less than that of the area size.

5.9 Conclusion

In this chapter, we have studied the problem of tracking moving objects in an open area using a UAV with a limited FOV. We suggested a control architecture with two types of observers. First, we used Kalman filters for estimating the state of each moving objects. Second, we used a probability map to track the movement of the UAV and calculate the most likely part of the area containing undiscovered objects. To help select positions for the tracked objects to use in the search and track (SaT) algorithm, we developed a theoretical result called the necessary visitation period (NVP) which relates the covariance of each Kalman object to a probability. Then, we introduced a SaT algorithm, which combines combinatorial optimization and optimal control. This addressed the weakness of each individual approach. Optimal control problems are implemented in a receding horizon fashion, and can get stuck in local optima. The initialization using a combinatorial solution gave us a global optimum. It is difficult to incorporate UAV dynamics in a combinatorial formulation, but this is easily incorporated in an optimal control formulation. To validate the SaT algorithm we introduced a scenario consisting of an open area and a set number of objects. To make the objects stay within the given area, they were set to behave as a snake from the arcade game. The SaT algorithm was compared to several base cases and a best case, which utilized perfect information. The scenario was used in Monte Carlo simulations, which demonstrated that the SaT algorithm performed better than the base cases for larger areas with minor

differences for smaller areas. The number of objects in the area had less of an effect on the difference, but an increased number of objects decreased the difference between the SaT-algorithm and the base cases.

Future Work

Future work will include:

1. Expanding the SaT algorithm to include multiple UAVs.
2. Performing full-size experiments to validate the results.

Conclusion

In this thesis, we have presented trajectory planning algorithms for fixed wing UAVs for target tracking as well as target searching and tracking. Iceberg monitoring has been the motivating real-world application.

In Chapter 1, we defined *target tracking* and *target searching and tracking*. In addition, we introduced the main models used both for target as well as sensors, UAVs, throughout this thesis. The chapter also contains a discussion of real-life scenarios applicable to target searching and/or tracking. We also introduced the optimization tools utilized in the trajectory planning algorithms of this thesis.

Chapter 2 is a survey on the literature of mobile sensor networks utilized to solve target searching and tracking problems. We focused on the trajectory planning algorithms as well as the target estimation algorithms, also known as the observer. We classified the literature based on the solutions characteristics and the problem formulation.

A trajectory planning algorithm for target tracking based on mixed integer linear programming was introduced in Chapter 3. This algorithm uses a formulation similar to the traveling salesperson problem, known as the target visitation problem (Grundel and Jeffcoat 2004). It utilized multiple UAVs, which can be considered a mobile sensor network. The trajectory planning algorithm was demonstrated both in simulation and experiments performed at Ny-Ålesund.

In Chapter 4, we present a trajectory planning algorithm for target tracking where we utilize an optimal control formulation. We focus on the implementation, in which we combine single shooting and collocation to increase the computational efficiency compared to collocation alone.

Chapter 5 contains a trajectory planning algorithm for target searching and track-

ing for a single UAV, which combines the ideas from the trajectory planning algorithms from Chapter 3 and 4. The trajectory planning is done in two layers. First, a MILP formulation is used to find an optimal path without considering the dynamic constraints imposed by the UAV. The MILP solution is used to initialize an optimal control problem (OCP). In general, OCP suffers from non-convex solutions and is hard to solve. The initialization enables the OCP optimization to obtain a near optimal solution, which takes the dynamic constraints of the UAV into consideration.

Future Work

In this thesis, our main focus has been algorithm development for a single UAV. However, in many applications for target searching and tracking it could be necessary to utilize multiple UAVs. A natural extension is therefore to make the algorithms from Chapter 4 and 5 to multiple mobile sensors.

Another possible extension of the trajectory planning algorithms presented in this thesis is to make them distributed. All three algorithms suffer from the curse of dimensionality and will scale poorly with increased problem size. In addition, in multiple applications the mobile sensors cannot be expected to be continuously communicating with a base station. In this case, it will be required that the trajectory planning is done in a distributed fashion where each mobile sensor only has contact with its neighbors (sensors in close vicinity).

For practical implementations there are multiple factors to consider. First, the model for the target tracking requires a Guassian distributed process noise, quantified by a covariance matrix. The covariance matrix is used by all the algorithms in this this thesis. In the tracking algorithm in Chapter 3 it is used to prioritize between icebergs, in the algorithm in Chapter 4 it is indirectly used in the objective function, and in the searching and tracking algorithm in Chapter 5 it is used to calculate the probability of redetecting a target when it is left without observation. In practice, it will be necessary to estimate this covariance matrix. Another practical problem is matching observations with estimates. This is particularly relevant for our problem since we only have partial observations of targets. This problem is known as data association. Third, neither the target tracking algorithm of Chapter 3 nor 4 have a way to handle lost icebergs (target not being at its estimated position). The target searching and tracking algorithm in Chapter 5 can handle lost targets, which it does by stop tracking a target if it is not likely to be found. This is a necessary feature for a practical implementation.

Bibliography

.

Aeryon
2007. Aeryon. https://www.aeryon.com/. Accessed: 2018-05-05.

Akyildiz, I. F., W. Su, Y. Sankarasubramaniam, and E. Cayirci
2002. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422.

Albert, A. and L. Imsland
2015. Mobile sensor path planning for iceberg monitoring using a MILP framework. In *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings*, volume 1, Pp. 131–138.

Albert, A. and L. Imsland
2017. Performance bounds for tracking multiple objects using a single UAV. In *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*, Pp. 1539–1546. IEEE.

Albert, A. and L. Imsland
2018a. Combined optimal control and combinatorial optimization for searching and tracking using an unmanned aerial vehicle. *Journal of Intelligent & Robotic Systems*. Accepted for publication.

Albert, A. and L. Imsland
2018b. Survey: Mobile sensor networks for target detection and tracking. *Cyber-Physical Systems*, 0(0):1–42.

Albert, A., L. Imsland, and J. Haugen

2016. Numerical optimal control mixing collocation with single shooting: A case study. *IFAC-PapersOnLine*, 49(7):290–295.

Albert, A., F. S. Leira, and L. Imsland
2017. UAV path planning using MILP with experiments. *Modeling, Identification and Control*, 38(1):21.

Amundson, I. and X. Koutsoukos
2009. A survey on localization for mobile wireless sensor networks. *Mobile Entity Localization and Tracking in GPS-less Environnments*, Pp. 235–254.

Andersson, J.
2013a. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Andersson, J.
2013b. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium.

Applegate, D., R. Bixby, V. Chvátal, and W. Cook
2011. *The traveling salesman problem: A computational study*. Princeton University Press.

Arasaratnam, I. and S. Haykin
2009. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269.

ArduPilot
2016. ArduPilot. http://ardupilot.com/. Accessed: 2016-02-10.

Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp
2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Asnis, G. and S. Blackman
2011. Optimal allocation of multi-platform sensor resources for multiple target tracking. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, Pp. 1–8. IEEE.

Bai, J., P. Cheng, J. Chen, A. Guenard, and Y. Song
2012. Target tracking with limited sensing range in autonomous mobile sensor networks. In *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, Pp. 329–334. IEEE.

Balas, E.
1989. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636.

Barton, K. and D. Kingston
2013. Systematic surveillance for UAVs: A feedforward iterative learning control approach. In *2013 American Control Conference*, Pp. 5917–5922. IEEE.

Baumgartner, K. A., S. Ferrari, and A. V. Rao
2009. Optimal control of an underwater sensor network for cooperative target tracking. *IEEE Journal of Oceanic Engineering*, 34(4):678–697.

Beaudeau, J. P., M. F. Bugallo, and P. M. Djurić
2015. RSSI-based multi-target tracking by cooperative agents using fusion of cross-target information. *IEEE Transactions on Signal Processing*, 63(19):5033–5044.

Betts, J. T.
2010. *Practical methods for optimal control and estimation using nonlinear programming*. SIAM.

Biegler, L. T.
2010. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*, volume 10, 1st edition. SIAM.

Binder, T., L. Blank, H. G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. P. Schlöder, and O. von Stryk
2001. *Introduction to model based optimization of chemical processes on moving horizons*, Pp. 295–339. Springer.

Bixby, R. E.
2002. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15.

Blackman, S. S.
1988. Theoretical approaches to data association and fusion. In *Sensor Fusion*, volume 931, Pp. 50–56. International Society for Optics and Photonics.

Carlson, N. A.
1990. Federated square root filter for decentralized parallel processors. *IEEE Transactions on Aerospace and Electronic Systems*, 26(3):517–525.

Chang, C.-Y., G. Chen, G.-J. Yu, T.-L. Wang, and T.-C. Wang
2015. TCWTP: Time-constrained weighted targets patrolling mechanism in wireless mobile sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(6):901–914.

Chattopadhyay, P., D. K. Jha, S. Sarkar, and A. Ray
2015. Path planning in GPS-denied environments: A collective intelligence approach. In *American Control Conference (ACC), 2015*, Pp. 3082–3087. IEEE.

Chen, D.-S., R. G. Batson, and Y. Dang
2010. *Applied integer programming: modeling and solution*. John Wiley & Sons.

Chen, H. and K. Sezaki
2011. Distributed target tracking algorithm for wireless sensor networks. In *Communications (ICC), 2011 IEEE International Conference on*, Pp. 1–5. IEEE.

Chen, Y.-Y., C.-C. Hsu, C.-F. Chou, and K. Lin
2012. On detecting mobile target with deadline constraint in mobile sensor networks. In *Sensors, 2012 IEEE*, Pp. 1–4. IEEE.

Cheng, P., X. Cao, J. Bai, and Y. Sun
2012. On optimizing sensing quality with guaranteed coverage in autonomous mobile sensor networks. *Computer Communications*, 35(9):1107–1114.

Chin, J.-C., Y. Dong, W.-K. Hon, C. Y.-T. Ma, and D. K. Yau
2010. Detection of intelligent mobile target in a mobile sensor network. *IEEE/ACM Transactions on Networking (TON)*, 18(1):41–52.

Chung, T. H., G. A. Hollinger, and V. Isler
2011. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299.

Dagdeviren, O., I. Korkmaz, F. Tekbacak, and K. Erciyes
2011. A survey of agent technologies for wireless sensor networks. *IETE Technical Review*, 28(2):168–184.

Dang, A. D. and J. Horn
2015. A mobile sensor network tracking moving targets in a dynamic environment. *IFAC-PapersOnLine*, 48(5):1–6.

Dang, A. D., H. M. La, and J. Horn
2016. Distributed formation control for autonomous robots following desired shapes in noisy environment. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2016 IEEE International Conference on*, Pp. 285–290. IEEE.

Dantzig, G., R. Fulkerson, and S. Johnson
1954. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

DeSena, J. T., S. R. Martin, J. C. Clarke, D. A. Dutrow, B. C. Kohan, and I. Kadar
2013. Decentralized closed-loop collaborative surveillance and tracking performance sensitivity to communications connectivity. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 8745.

Dormand, J. R. and P. J. Prince
1980. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26.

Eggers, J. and M. H. Draper
2006. Multi-UAV control for tactical reconnaissance and close air support missions: operator perspectives and design challenges. In *Proc. NATO RTO Human Factors and Medicine Symp. HFM-135. NATO TRO, Neuilly-sur-Siene, CEDEX, Biarritz, France*, Pp. 2006–11.

Eik, K.
2008. Review of experiences within ice and iceberg management. *Journal of Navigation*, 61(4):557–572.

Enright, J., E. Frazzoli, K. Savla, and F. Bullo
2005. On multiple UAV routing with stochastic targets: Performance bounds and algorithms. In *Proc. of the AIAA Conf. on Guidance, Navigation, and Control*.

Esmailifar, S. M. and F. Saghafi
2017. Cooperative localization of marine targets by UAVs. *Mechanical Systems and Signal Processing*, 87:23–42.

Farmani, N., L. Sun, and D. Pack
2015. Tracking multiple mobile targets using cooperative unmanned aerial vehicles. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, Pp. 395–400. IEEE.

Ferrari, S., M. Anderson, R. Fierro, and W. Lu
2011. Cooperative navigation for heterogeneous autonomous vehicles via approximate dynamic programming. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, Pp. 121–127. IEEE.

Ferrari, S., R. Fierro, B. Perteet, C. Cai, and K. Baumgartner
2009. A geometric optimization approach to detecting and intercepting dynamic targets using a mobile sensor network. *SIAM Journal on Control and Optimization*, 48(1):292–320.

Fossen, T. I.
2011. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.

Fu, Y. and L. Yang
2014. Sensor mobility control for multitarget tracking in mobile sensor networks. *International Journal of Distributed Sensor Networks*, 2014.

Furukawa, T., F. Bourgault, B. Lavis, and H. F. Durrant-Whyte
2006. Recursive bayesian search-and-tracking using coordinated UAVs for lost targets. *IEEE International Conference on Robotics and Automation (ICRA)*, Pp. 2521–2526.

Gao, X., Z. Chen, F. Wu, and G. Chen
2017. Energy efficient algorithms for k-sink minimum movement target coverage problem in mobile sensor network. *IEEE/ACM Transactions on Networking*.

Gelb, A.
1974. *Applied optimal estimation*. MIT press.

Giannini, S., D. Di Paola, and A. Rizzo
2012. Coverage-aware distributed target tracking for mobile sensor networks. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, Pp. 1386–1391. IEEE.

Girard, A. R., A. S. Howell, and J. K. Hedrick
2004. Border patrol and surveillance missions using multiple unmanned air vehicles. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, Pp. 620–625. IEEE.

Glade, D.
2000. Unmanned aerial vehicles: Implications for military operations. Technical report, DTIC Document.

Golden, B. L., S. Raghavan, and E. A. Wasil
2008. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media.

Goodrich, M. A., B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey
2008. Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, 25(1-2):89–110.

Griewank, A. and A. Walther
2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM.

Grundel, D. and D. Jeffcoat
2004. Formulation and solution of the target visitation problem. *Collection of Technical Papers - AIAA 1st Intelligent Systems Technical Conference*, 1:1–6.

Gu, D. and H. Hu
2010. Distributed minimax filter for tracking and flocking. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Pp. 3562–3567. IEEE.

Gu, D. and H. Hu
2011. Rényi entropy based target tracking in mobile sensor networks. *IFAC Proceedings Volumes*, 44(1):13558–13563.

Gusrialdi, A., T. Hatanaka, and M. Fujita
2008. Coverage control for mobile networks with limited-range anisotropic sensors. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, Pp. 4263–4268. IEEE.

Gutin, G. and A. P. Punnen
2006. *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media.

Haugen, J.
2014. *Autonomous Aerial Ice Observation*. PhD thesis, Norwegian University of Science and Technology.

Haugen, J. and L. Imsland
2013. Optimization-based autonomous remote sensing of surface objects using an unmanned aerial vehicle. *2013 European Control Conference, ECC 2013*, Pp. 1242–1249.

Haugen, J. and L. Imsland
2016. Monitoring moving objects using aerial mobile sensors. *IEEE Transactions on Control Systems Technology*, 24(2):475–486.

Hernandez, M.
2012. Performance bounds for target tracking: computationally efficient formulations and associated applications. *Integrated Tracking, Classification, and Sensor Management: Theory and Applications*, Pp. 255–310.

Hirsch, M. J. and D. Schroeder
2015. On the decentralized cooperative control of multiple autonomous vehicles. In *Handbook of Unmanned Aerial Vehicles*, Pp. 1577–1600. Springer.

Hoffmann, G. M. and C. J. Tomlin
2010. Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*, 55(1):32–47.

HSL
2015. A collection of fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk. [Online; accessed 19-March-2015].

Hsu, C.-C., Y.-Y. Chen, C.-F. Chou, and L. Golubchik
2013. On design of collaborative mobile sensor networks for deadline-sensitive mobile target detection. *IEEE Sensors Journal*, 13(8):2962–2972.

Hu, F. and C. Tu
2017. An optimization model for target tracking of mobile sensor network based on motion state prediction in emerging sensor networks. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–16.

Hu, J., L. Xie, and C. Zhang
2012. Energy-based multiple target localization and pursuit in mobile sensor networks. *IEEE Transactions on Instrumentation and measurement*, 61(1):212–220.

Hung, L.-L.
2014. Efficient algorithms for sensor detachment in wmsns. *International Journal of Ad Hoc and Ubiquitous Computing*, 16(3):172–182.

Hutchinson, S. and T. Bretl
2012. Robust optimal deployment of mobile sensor networks. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, Pp. 671–676. IEEE.

IBM
2015. IBM ilog CPLEX optimization studio CPLEX. http://www.ibm.com. Accessed: 2015-09-12.

Imai, K. and T. Ushio
2013. Effective combination of search policy based on probability and entropy for heterogeneous mobile sensors. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, Pp. 1981–1986. IEEE.

Irving, R. S.
2003. *Integers, polynomials, and rings: a course in algebra*. Springer Science & Business Media.

Isenor, A. W., Y. Allardb, A.-L. S. Lapinskia, H. Demersb, and D. Radulescub
2014. Coordinating UAV information for executing national security-oriented collaboration. In *Proc. of SPIE Vol*, volume 9248.

Jadaliha, M. and J. Choi
2013. Environmental monitoring using autonomous aquatic robots: Sampling algorithms and experiments. *IEEE Transactions on Control Systems Technology*, 21(3):899–905.

Jalalkamali, P. and R. Olfati-Saber
2012. Information-driven self-deployment and dynamic sensor coverage for mobile sensor networks. In *American Control Conference (ACC), 2012*, Pp. 4933–4938. IEEE.

Jha, D. K., P. Chattopadhyay, S. Sarkar, and A. Ray
2016. Path planning in GPS-denied environments via collective intelligence of distributed sensor networks. *International Journal of Control*, 89(5):984–999.

Jiang, S., L. Dou, and H. Fang
2013. Target tracking based on federated filter for mobile sensor networks. In *Computer Science and Network Technology (ICCSNT), 2013 3rd International Conference on*, Pp. 1263–1268. IEEE.

Jin, D., W. Zeng, H. Su, H. Zhou, and M. Delie
2017. Distributed estimation and control of mobile sensor networks based only on position measurements. *IET Control Theory & Applications*, 11(10):1627–1633.

Juan-Yi, Z.
2011. An improved target tracking accuracy algorithm based on particle filtering

in WMSN. In *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, Pp. 131–134. IEEE.

Juang, P., H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein
2002. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *ACM SIGARCH Computer Architecture News*, 30(5):96–107.

Jünger, M., T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey
2009. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*. Springer Science & Business Media.

Kalman, R. E. et al.
1960. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kamath, S., E. Meisner, and V. Isler
2007. Triangulation based multi target tracking with mobile sensor networks. In *Robotics and Automation, 2007 IEEE International Conference on*, Pp. 3283–3288. IEEE.

Kan, Z., E. L. Pasiliao, J. W. Curtis, and W. E. Dixon
2012. Particle filter based average consensus target tracking with preservation of network connectivity. In *Military Communications Conference, 2012-MILCOM 2012*, Pp. 1–6. IEEE.

Kassas, Z. M., A. Arapostathis, and T. E. Humphreys
2015. Greedy motion planning for simultaneous signal landscape mapping and receiver localization. *IEEE Journal of Selected Topics in Signal Processing*, 9(2):247–258.

Khodayari, E., V. Sattari-Naeini, and M. Mirhosseini
2016. Flocking control with single-com for tracking a moving target in mobile sensor network using gravitational search algorithm. In *Swarm Intelligence and Evolutionary Computation (CSIEC), 2016 1st Conference on*, Pp. 125–130. IEEE.

Koohifar, F., A. Kumbhar, and I. Guvenc
2017. Receding horizon multi-UAV cooperative tracking of moving rf source. *IEEE Communications Letters*, 21(6):1433–1436.

Koopman, B. O.
1956a. The theory of search. i. kinematic bases. *Operations research*, 4(3):324–346.

Koopman, B. O.
1956b. The theory of search. ii. target detection. *Operations research*, 4(5):503–531.

Koopman, B. O.
1957. The theory of search: Iii. the optimum distribution of searching effort. *Operations research*, 5(5):613–626.

Krishna, K. M., H. Hexmoor, S. Pasupuleti, and S. Chellappa
2004. A surveillance system based on multiple mobile sensors. In *FLAIRS Conference*, Pp. 128–133.

Kuhn, H. W.
1955. The Hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

Kuo, C.-H., T.-S. Chen, and S.-C. Syu
2017. Adaptive trap coverage in mobile sensor networks. *Procedia Computer Science*, 110:102–109.

La, H. M., T. H. Nguyen, C. H. Nguyen, and H. N. Nguyen
2009. Optimal flocking control for a mobile sensor network based a moving target tracking. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, Pp. 4801–4806. IEEE.

La, H. M. and W. Sheng
2009a. Adaptive flocking control for dynamic target tracking in mobile sensor networks. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Pp. 4843–4848. IEEE.

La, H. M. and W. Sheng
2009b. Flocking control of a mobile sensor network to track and observe a moving target. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, Pp. 3129–3134. IEEE.

La, H. M. and W. Sheng
2009c. Moving targets tracking and observing in a distributed mobile sensor network. In *American Control Conference, 2009. ACC'09.*, Pp. 3319–3324. IEEE.

La, H. M. and W. Sheng
2011a. Cooperative sensing in mobile sensor networks based on distributed consensus. In *Proc. Signal Data Process. Small Targets Conf*, Pp. 81370Y1–81370Y14.

La, H. M. and W. Sheng
2011b. Flocking control algorithms for multiple agents in cluttered and noisy environments.

La, H. M. and W. Sheng
2012. Dynamic target tracking and observing in a mobile sensor network. *Robotics and Autonomous Systems*, 60(7):996–1009.

Lalooses, F., H. Susanto, and C. H. Chang
2005. Recovery target tracking in wildlife. *Wireless and Optical Communication, Montreal Canada*.

Latif, T., E. Whitmire, T. Novak, and A. Bozkurt
2016. Sound localization sensors for search and rescue biobots. *IEEE Sensors Journal*, 16(10):3444–3453.

Leira, F., T. A. Johansen, and T. I. Fossen
2015. Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera. In *IEEE Aerospace Conference*.

Lešinskis, I. and A. Pavlovičs
2011. The aspects of implementation of unmanned aerial vehicles for ice situation awareness in maritime traffic. In *Proceedings of 15th International Conference Transport Means*, Pp. 65–68.

Lesinskis, I. and A. Pavlovics
2011. The aspects of implementation of unmanned aerial vehicles for ice situation awareness in maritime traffic. *Transport Means - Proceedings of the International Conference*, Pp. 65–68.

Li, Y. and P. M. Djuric
2007. Particle filtering for target tracking with mobile sensors. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, Pp. II–1101. IEEE.

Li, Y. and P. M. Djuric
2008. Target tracking with mobile sensors using cost-reference particle filtering. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, Pp. 2549–2552. IEEE.

Li, Y. and Y.-H. Liu
2007. Energy saving target tracking using mobile sensor networks. In *Robotics and Automation, 2007 IEEE International Conference on*, Pp. 3653–3658. IEEE.

Li, Y., Y.-h. Liu, and X. Cai
2007. Local control strategy for target tracking in mobile sensor networks. In *Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on*, Pp. 674–679. IEEE.

Li, Y., Y.-h. Liu, H. Zhang, H. Wang, X. Cai, and D. Zhou
2008. Distributed target tracking with energy consideration using mobile sensor networks. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, Pp. 3280–3285. IEEE.

Li, Y.-y. and Y.-h. Liu
2009. Tracking point or diffusing targets using mobile sensor networks under sensing noises. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Pp. 564–569. IEEE.

Liao, Z., J. Wang, S. Zhang, J. Cao, and G. Min
2015. Minimizing movement for target coverage and network connectivity in mobile sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(7):1971–1983.

Liu, L., Y. Sun, and Z. Wang
2007. CTCOMSN: Collaborative target coverage optimization in mobile sensor networks. Pp. 636–639.

Löfberg, J.
2004. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan.

Loria, A. and E. Panteley
2005. Cascaded nonlinear time-varying systems: Analysis and design. *Lecture Notes in Control and Information Sciences*, 311:23–64.

Low, K. H., W. K. Leow, and M. H. Ang
2004a. Reactive, distributed layered architecture for resource-bounded multi-robot cooperation: Application to mobile sensor network coverage. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, Pp. 3747–3752. IEEE.

Low, K. H., W. K. Leow, and M. H. Ang
2006. Autonomic mobile sensor network with self-coordinated task allocation and execution. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(3):315–327.

Low, K. H., W. K. Leow, and M. H. Ang Jr
2004b. Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In *Proceedings of the National Conference on Artificial Intelligence*, volume 4, Pp. 28–33.

Lu, W., G. Zhang, S. Ferrari, M. Anderson, and R. Fierro
2014. A particle-filter information potential method for tracking and monitoring maneuvering targets using a mobile sensor agent. *The Journal of Defense Modeling and Simulation*, 11(1):47–58.

Ma, C. S. and R. H. Miller
2005. Mixed integer linear programming trajectory generation for autonomous nap-of-the-earth flight in a threat environment. In *2005 IEEE Aerospace Conference*, Pp. 1–9. IEEE.

Ma, K., Y. Zhang, and W. Trappe
2008. Managing the mobility of a mobile sensor network using network dynamics. *IEEE Transactions on Parallel and Distributed Systems*, 19(1):106–120.

Ma, X. and J. Tan
2013. Active sensing with mobile sensor networks: A survey. *Journal of Communications*, 8(2):110–127.

Mahboubi, H., W. Masoudimansour, A. G. Aghdam, and K. Sayrafian-Pour
2011. Cost-efficient routing with controlled node mobility in sensor networks. In *Control Applications (CCA), 2011 IEEE International Conference on*, Pp. 1238–1243. IEEE.

Mahboubi, H., W. Masoudimansour, A. G. Aghdam, and K. Sayrafian-Pour
2016. Maximum lifetime strategy for target monitoring with controlled node mobility in sensor networks with obstacles. *IEEE Transactions on Automatic Control*, 61(11):3493–3508.

Mahboubi, H., W. Masoudimansour, A. G. Aghdam, and K. Sayrafian-Pour
2017. An energy-efficient target-tracking strategy for mobile sensor networks. *IEEE transactions on cybernetics*, 47(2):511–523.

Mahboubi, H., W. Masoudimansour, A. G. Aghdam, K. Sayrafian-Pour, and V. Marbukh
2012. Maximum life span strategy for target tracking in mobile sensor networks. In *American Control Conference (ACC), 2012*, Pp. 5096–5101. IEEE.

Mahboubi, H., A. Momeni, A. G. Aghdam, K. Sayrafian-Pour, and V. Marbukh
2010. Optimal target tracking strategy with controlled node mobility in mobile sensor networks. In *American Control Conference (ACC), 2010*, Pp. 2921–2928. IEEE.

Mallick, M., S. Coraluppi, and C. Carthel
2012. Multitarget tracking using multiple hypothesis tracking. *Integrated Tracking, Classification, and Sensor Management: Theory and Applications*, Pp. 163–203.

Marbukh, V., K. Sayrafian-Pour, H. Mahboubi, A. Momeni, and A. G. Aghdam
2010. Towards evolutionary-pricing framework for mobile sensor network self-organization. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, Pp. 1–8. IEEE.

Martínez, S. and F. Bullo
2006. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668.

Martins, R., P. S. Dias, E. R. Marques, J. Pinto, J. B. Sousa, and F. L. Pereira
2009. IMC: A communication protocol for networked vehicles and sensors. In *Oceans 2009-Europe*, Pp. 1–6. IEEE.

Mathew, G., A. Surana, and I. Mezić
2010. Uniform coverage control of mobile sensor networks for dynamic target detection. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, Pp. 7292–7299. IEEE.

Mavrommati, A., E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey
2017. Real-time area coverage and target localization using receding-horizon ergodic exploration. *IEEE Transactions on Robotics*.

Meng, W., Z. He, R. Su, P. K. Yadav, R. Teo, and L. Xie
2017. Decentralized multi-UAV flight autonomy for moving convoys search and track. *IEEE Transactions on Control Systems Technology*, 25(4):1480–1487.

Míguez, J., M. F. Bugallo, and P. M. Djurić
2004. A new class of particle filters for random dynamic systems with

unknown statistics. *EURASIP Journal on Advances in Signal Processing*, 2004(15):303619.

Miller, C. E., A. W. Tucker, and R. A. Zemlin
1960. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4):326–329.

Mourad, F., H. Chehade, H. Snoussi, F. Yalaoui, L. Amodeo, and C. Richard
2012. Controlled mobility sensor networks for target tracking using ant colony optimization. *IEEE Transactions on Mobile Computing*, 11(8):1261–1273.

Naderan, M., M. Dehghan, and H. Pedram
2009. Mobile object tracking techniques in wireless sensor networks. In *Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on*, Pp. 1–8. IEEE.

Naderan, M., M. Dehghan, H. Pedram, and V. Hakami
2012. Survey of mobile object tracking protocols in wireless sensor networks: a network–centric perspective. *International Journal of Ad Hoc and Ubiquitous Computing*, 11(1):34–63.

Nelson, T. R. and R. A. Freeman
2009a. Decentralized H ∞ filtering in a multi-agent system. In *American Control Conference, 2009. ACC'09.*, Pp. 5755–5760. IEEE.

Nelson, T. R. and R. A. Freeman
2009b. Set-valued estimation for mobile sensor networks. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, Pp. 2681–2686. IEEE.

Noon, C. E. and J. C. Bean
1993. An efficient transformation of the generalized traveling salesman problem. *INFOR*, 31(1):39.

Oberlin, P., S. Rathinam, and S. Darbha
2010. Today's traveling salesman problem. *IEEE Robotics & Automation Magazine*, 17(4):70–77.

Odroid
2016. Odroid U3. http://www.odroid.com/. Accessed: 2016-03-02.

Oh, H., H.-S. Shin, S. Kim, A. Tsourdos, and B. A. White
2015. Cooperative mission and path planning for a team of UAVs. In *Handbook of Unmanned Aerial Vehicles*, Pp. 1509–1545. Springer.

Oispuu, M., M. Sciotti, and A. Charlish
2013. Air route selection for improved air-to-ground situation assessment. In *SPIE Defense, Security, and Sensing*, Pp. 87420M–87420M. International Society for Optics and Photonics.

Olfati-Saber, R.
2006. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420.

Olfati-Saber, R.
2007. Distributed tracking for mobile sensor networks with information-driven mobility. In *American Control Conference, 2007. ACC'07*, Pp. 4606–4612. IEEE.

Olfati-Saber, R.
2009. Kalman-consensus filter: Optimality, stability, and performance. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, Pp. 7036–7042. IEEE.

Olfati-Saber, R. and P. Jalalkamali
2012. Coupled distributed estimation and control for mobile sensor networks. *IEEE Transactions on Automatic Control*, 57(10):2609–2614.

Olfati-Saber, R. and R. M. Murray
2004. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533.

Park, H. and S. Hutchinson
2013. Worst-case performance of a mobile sensor network under individual sensor failure. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, Pp. 895–900. IEEE.

Pereira, E., R. Bencatel, J. Correia, L. Félix, G. Gonçalves, J. Morgado, and J. Sousa
2009. Unmanned air vehicles for coastal and environmental research. *Journal of Coastal Research*, Pp. 1557–1561.

Pesch, H. J., M. Plail, and D. Munich
2009. The maximum principle of optimal control: A history of ingenious ideas and missed opportunities. *Control and Cybernetics*, 38(4A):973–995.

Peterson, C. K., A. J. Newman, and J. C. Spall
2014. Simulation-based examination of the limits of performance for decentralized multi-agent surveillance and tracking of undersea targets. In *Signal Processing, Sensor/Information Fusion, and Target Recognition XXIII*, volume 9091, P. 90910F. International Society for Optics and Photonics.

Pinto, J., P. Calado, J. Braga, P. Dias, R. Martins, E. Marques, and J. Sousa
2012. Implementation of a control architecture for networked vehicle systems. In *Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*. IFAC.

Pinto, J., P. S. Dias, R. Gonçalves, E. R. B. Marques, G. M. Gonçalves, J. B. Sousa, and F. L. Pereira
2006. Neptus: a framework to support a mission life cycle. In *Proc. IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC)*. IFAC.

Pitre, R. R., X. R. Li, and R. Delbalzo
2012. UAV route planning for joint search and track missions-an information-value approach. *IEEE Transactions on Aerospace and Electronic Systems*, 48(3):2551–2565.

Pontryagin, L. S.
1957. *Mathematical theory of optimal processes*. CRC Press.

Portugal, D. and R. Rocha
2011. A survey on multi-robot patrolling algorithms. *Technological innovation for sustainability*, Pp. 139–146.

Prabhavathi, M. and R. Rajeshwari
2011. Cluster-based mobility management for target tracking in mobile sensor networks. In *Advanced Computing (ICoAC), 2011 Third International Conference on*, Pp. 198–203. IEEE.

ProxDynamics
2008. Prox Dynamics. http://www.proxdynamics.com/home. Accessed: 2018-04-16.

Pulford, G.
2005. Taxonomy of multiple target tracking methods. *IEE Proceedings-Radar, Sonar and Navigation*, 152(5):291–304.

Punyawee, A., C. Panumate, and H. Iida
2016. Finding comfortable settings of snake game using game refinement meas-

urement. *International Conference on Computer Science and its Applications*, Pp. 66–73.

Puri, A.
2005. A survey of unmanned aerial vehicles (UAV) for traffic surveillance. *Department of computer science and engineering, University of South Florida*.

Qi, Y., P. Cheng, J. Bai, J. Chen, A. Guenard, Y.-Q. Song, and Z. Shi
2016. Energy-efficient target tracking by mobile sensors with limited sensing range. *IEEE Transactions on Industrial Electronics*, 63(11):6949–6961.

Qiu, C., Z. Zhang, H. Lu, and H. Luo
2015. A survey of motion-based multitarget tracking methods. *Progress In Electromagnetics Research B*, 62:195–223.

Rahman, M. J. A., A. I. Abu-El-Haija, and H. M. Al-Najjar
2011. On the detection of intelligent mobile targets in a mobile sensor network. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, Pp. 1268–1275. IEEE.

Ren, G., V. Maroulas, and I. D. Schizas
2016. Exploiting sensor mobility and covariance sparsity for distributed tracking of multiple sparse targets. *EURASIP Journal on Advances in Signal Processing*, 2016(1):53.

Reynolds, C. W.
1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics*, 21(4):25–34.

Rigatos, G. G.
2011. A distributed motion planning and distributed filtering approach for target tracking in mobile sensor networks. *IFAC Proceedings Volumes*, 44(1):4771–4778.

Robin, C. and S. Lacroix
2016. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760.

Rout, M. and R. Roy
2016. Self-deployment of mobile sensors to achieve target coverage in the presence of obstacles. *IEEE Sensors Journal*, 16(14):5837–5842.

Russell, S. J. and P. Norvig
2002. *Artificial intelligence: a modern approach (International Edition)*. Pearson US Imports & PHIPEs.

Schouwenaars, T., B. De Moor, E. Feron, and J. How
2001. Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, Pp. 2603–2608. IEEE.

Schrijver, A.
2005. On the history of combinatorial optimization (till 1960). *Handbooks in Operations Research and Management Science*, 12:1–68.

Selvaraj, K. and S. Balaji
2013. Controlled mobility sensor networks for target tracking using particle swarm optimization. In *Current Trends in Engineering and Technology (ICCTET), 2013 International Conference on*, Pp. 388–391. IEEE.

Sensefly
2009. senseFly. https://www.sensefly.com/. Accessed: 2018-05-05.

Shaked, U. and Y. Theodor
1992. H ∞-optimal estimation: a tutorial. In *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, Pp. 2278–2286. IEEE.

Shampine, L. F. and M. W. Reichelt
1997. The matlab ode suite. *SIAM journal on scientific computing*, 18(1):1–22.

Sharma, S. et al.
2015. Target tracking technique in wireless sensor network. In *Computing, Communication & Automation (ICCCA), 2015 International Conference on*, Pp. 486–491. IEEE.

Shetty, V. K., M. Sudit, and R. Nagi
2008. Priority-based assignment and routing of a fleet of unmanned combat aerial vehicles. *Computers & Operations Research*, 35(6):1813–1828.

Simon, M.
2006. *Probability distributions involving gaussian random variables: A handbook for engineers and scientists*, volume 683.

Sinha, A., T. Kirubarajan, and Y. Bar-Shalom
2005a. Autonomous ground target tracking by multiple cooperative UAVs. *2005 IEEE Aerospace Conference*, Pp. 1–9.

Sinha, A., T. Kirubarajan, and Y. Bar-Shalom
2005b. Autonomous surveillance by multiple cooperative UAVs. *Proc. of SPIE Vol*, 5913:59131V–1.

Sinopoli, B., L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry
2004. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464.

SkyWalkerTechnology
2016. Sky walker technology (HK) co., ltd. http://www.skywalker-model.com/. Accessed: 2016-02-10.

Smith, S. L.
2009. *Task allocation and vehicle routing in dynamic environments*. University of California, Santa Barbara.

Souza, É. L., E. F. Nakamura, and R. W. Pazzi
2016. Target tracking for sensor networks: A survey. *ACM Computing Surveys (CSUR)*, 49(2):30.

Stone, L. D.
1989. Or forum-what's happened in search theory since the 1975 lanchester prize? *Operations Research*, 37(3):501–506.

Stone, L. D., J. O. Royset, and A. R. Washburn
2016. *Optimal Search for Moving Targets*, volume 237. Springer.

Su, H., Z. Li, and M. Z. Chen
2017. Distributed estimation and control for two-target tracking mobile sensor networks. *Journal of the Franklin Institute*, 354(7):2994–3007.

Sun, L., S. Baek, and D. Pack
2014. Distributed probabilistic search and tracking of agile mobile ground targets using a network of unmanned aerial vehicles. In *Human Behavior Understanding in Networked Sensing*, Pp. 301–319. Springer.

Takahashi, J., K. Sekiyama, and T. Fukuda
2009. Cooperative object tracking with mobile robotic sensor network. *Distributed Autonomous Robotic Systems. Springer, Berlin*, Pp. 51–62.

Tan, J., N. Xi, W. Sheng, and J. Xiao
2004. Modeling multiple robot systems for area coverage and cooperation. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, Pp. 2568–2573. IEEE.

Tan, Q., X. Dong, Q. Li, and Z. Ren
2017. Weighted average consensus-based cubature Kalman filtering for mobile

sensor networks with switching topologies. In *Control & Automation (ICCA), 2017 13th IEEE International Conference on*, Pp. 271–276. IEEE.

Tian, X., Y. Bar-Shalom, and K. R. Pattipati
2008. Multi-step look-ahead policy for autonomous cooperative surveillance by UAVs in hostile environments. *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, Pp. 2438–2443.

Tichavsky, P., C. H. Muravchik, and A. Nehorai
1998. Posterior Cramér-Rao bounds for discrete-time nonlinear filtering. *IEEE Transactions on Signal Processing*, 46(5):1386–1396.

Tomic, T., K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka
2012. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics & Automation Magazine*, 19(3):46–56.

Tu, Z., Q. Wang, H. Qi, and Y. Shen
2012a. Flocking based distributed self-deployment algorithms in mobile sensor networks. *Journal of Parallel and Distributed Computing*, 72(3):437–449.

Tu, Z., Q. Wang, H. Qi, and Y. Shen
2012b. Flocking based sensor deployment in mobile sensor networks. *Computer Communications*, 35(7):849–860.

Ubiquiti
2016. Ubiquiti rocket M5. https://www.ubnt.com/airmax/rocketm/. Accessed: 2016-02-10.

Valavanis, K. and G. Vachtsevanos
2015. UAV applications: Introduction. *Handbook of Unmanned Aerial Vehicles*, Pp. 2639–2641.

Vanegas Alvarez, F.
2017. *Uncertainty based online planning for UAV missions in GPS-denied and cluttered environments*. PhD thesis, Queensland University of Technology.

Viterbi, A. J.
2010. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *The Foundations Of The Digital Wireless World: Selected Works of AJ Viterbi*, Pp. 41–50. World Scientific.

Wächter, A. and L. T. Biegler
2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57.

Walton, C., Q. Gong, I. Kaminer, and J. Royset
2014. Optimal motion planning for searching for uncertain targets. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 19(3):8977–8982.

Wan, E. A. and R. Van Der Merwe
2000. The unscented Kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, Pp. 153–158. IEEE.

Wang, L., N. Wang, and H. Zhu
2010a. Consensus based distributed unscented information filtering for air mobile sensor networks. In *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, volume 2, Pp. 492–495. IEEE.

Wang, L., Q. Zhang, H. Zhu, and L. Shen
2010b. Adaptive consensus fusion estimation for msn with communication delays and switching network topologies. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, Pp. 2087–2092. IEEE.

Wang, X., T. Song, and Y. Wu
2016. Covering a mobile target using mobile sensor networks. In *Control and Decision Conference (CCDC), 2016 Chinese*, Pp. 1433–1437. IEEE.

Wang, Y., Z. Tu, Q. Wang, Y. Shen, and J. Li
2012. Flocking based distributed deployment for target monitoring in mobile sensor networks: Algorithm and implementation. In *Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International*, Pp. 2472–2477. IEEE.

Wei, H. and S. Ferrari
2015. A geometric transversals approach to sensor motion planning for tracking maneuvering targets. *IEEE Transactions on Automatic Control*, 60(10):2773–2778.

Wu, W., F. Zhang, and Y. Wardi
2014. Target localization: Energy-information trade-offs using mobile sensor networks. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, Pp. 2944–2949. IEEE.

Xie, F., H. Xiao, and Y. Wang
2016. Coordinated target estimation and tracking control by mobile sensor networks. In *Advanced Robotics and Mechatronics (ICARM), International Conference on*, Pp. 399–404. IEEE.

Xu, Y., S. Salapaka, and C. L. Beck
2010. Dynamic maximum entropy algorithms for clustering and coverage control. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, Pp. 1836–1841. IEEE.

Yang, M., Y. Cao, L. Tan, and J. Yu
2008. An enhanced precise self-deployment algorithm in mobile sensor network. In *Information Science and Engineering, 2008. ISISE'08. International Symposium on*, volume 2, Pp. 786–789. IEEE.

Yanmaz, E. and H. Guclu
2010. Stationary and mobile target detection using mobile wireless sensor networks. In *INFOCOM IEEE Conference on Computer Communications Workshops, 2010*, Pp. 1–5. IEEE.

Yao, M. and M. Zhao
2015. Unmanned aerial vehicle dynamic path planning in an uncertain environment. *Robotica*, 33(3):611–621.

Yick, J., B. Mukherjee, and D. Ghosal
2008. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330.

Zhang, J., L. Jia, S. Niu, F. Zhang, L. Tong, and X. Zhou
2015. A space-time network-based modeling framework for dynamic unmanned aerial vehicle routing in traffic incident monitoring applications. *Sensors*, 15(6):13874–13898.

Zhang, L. and Y. Zhu
2015. Mobile sensor deployment based on distributed flocking algorithm. In *Signal Processing, Communications and Computing (ICSPCC), 2015 IEEE International Conference on*, Pp. 1–5. IEEE.

Zhao, C., M. Zhu, H. Liang, and Z. Wu
2016. The sustainable tracking strategy of moving target by UAV in an uncertain environment. *Control Conference (CCC), 2016 35th Chinese*, Pp. 5641–5647.

Zhao, S., B. M. Chen, and T. H. Lee
2014a. Optimal deployment of mobile sensors for target tracking in 2D and 3D spaces. *IEEE/CAA Journal of Automatica Sinica*, 1(1):24–30.

Zhao, W., Z. Tang, Y. Yang, L. Wang, and S. Lan
2014b. Cooperative search and rescue with artificial fishes based on fish-swarm algorithm for underwater wireless sensor networks. *The Scientific World Journal*, 2014.

Zhu, C., L. Shu, T. Hara, L. Wang, S. Nishio, and L. T. Yang
2014. A survey on communication and data management issues in mobile sensor networks. *Wireless Communications and Mobile Computing*, 14(1):19–36.

Zorbas, D. and T. Razafindralambo
2013. Prolonging network lifetime under probabilistic target coverage in wireless mobile sensor networks. *Computer Communications*, 36(9):1039–1053.

Zou, Y. and K. Chakrabarty
2007. Distributed mobility management for target tracking in mobile sensor networks. *IEEE Transactions on Mobile Computing*, 6(8).