Journal of Intelligent & Robotic Systems (2019) 95:691–706
https://doi.org/10.1007/s10846-018-0915-4

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

Anders Albert[1] · **Lars Imsland**[1]

Received: 9 November 2017 / Accepted: 3 August 2018 / Published online: 8 August 2018

Abstract

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.

Keywords UAV · Motion planning · Mathematical optimization · Combinatorial optimization · Path planning · Target tracking

1 Introduction

Searching and tracking moving objects using, for example, an 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 [1, 2]. Another potential marine application is search and rescue [3, 4]. 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 [5–7] and border patrol [8, 9].

✉ Anders Albert
anders.albert@ntnu.no

Lars Imsland
lars.imsland@ntnu.no

[1] Department of Engineering Cybernetics Norwegian University of Science Technology, Trondheim, Norway

There are multiple approaches spanning across several fields that deal with the problem of searching and tracking moving objects. In this paper, 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 [10]. 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 [11] managed to solve a 49-cities problem. Today, instances of 85,900 cities have been solved, see [12] for more information about TSP.

In this paper, 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 [13]. 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 [14] 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 simultaneously in the U.S. and Soviet Union after WWII by mathematicians such as Bellman and Pontryagin [15]. 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 paper, we use collocation for discretization, see [16] and [17] 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 [18]. Koopman laid the foundation for search theory with his work [19–21], 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 [22]. For more details about optimal search see [23].

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) [24], which has also been applied to UAVs [25]. For more on task allocation for UAVs see [26].

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 [7, 27]. Another way is to separate searching and tracking into two different modes, and then use heuristic rules to switch between the modes [28–30]. 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 [31].

In this paper, 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 [32–36]. These approaches use a grid to divide the surveillance area. [37] 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 [38, 39]. 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 [40] due to of its low computational complexity [41]. It has been applied to target tracking by UAVs [42, 43]

1.1 Contribution

The contributions of this paper are twofold. First, it is an extension of the results from [44] due to removing some of their restricting assumptions. 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.

1.2 Organization

We introduce the problem and the control architecture we use for the UAV and objects in Section 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 3. The second type of observer, a probability map to track the movement of the UAV, is found in Section 4. To quantify the deterioration of each tracked object, we derive a result in Section 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 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 7 and discuss the results in Section 8. Finally, we conclude and discuss further work in Sections 9 and 10.

1.3 Notation

Throughout this paper the notation $\boldsymbol{I}$ and 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.

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 1 illustrates the problem. We will make some simplifying assumption to this problem in the next Section to focus our scope of this paper to the path planning algorithm for the UAV.

The control architecture we have chosen is illustrated in Fig. 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

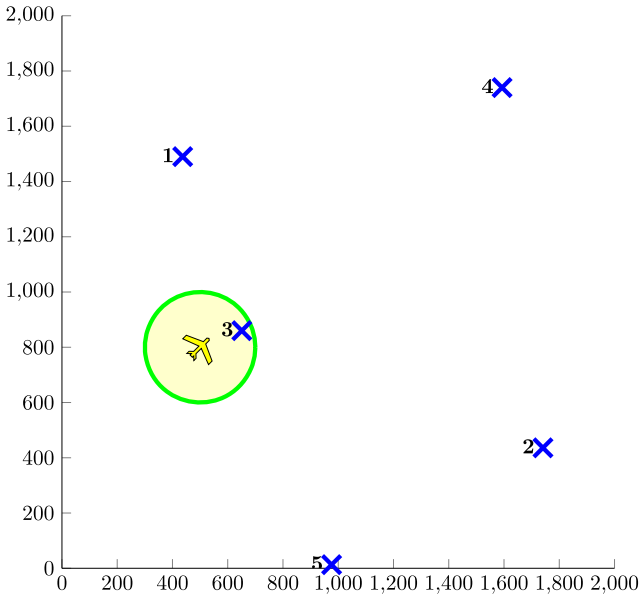


Fig. 1 Problem illustration. One UAV (yellow polygon) with limited field of view (circle with green border and light-yellow area) monitoring an area of size 2000 x 2000 m^2 with 5 moving objects (blue X's numbered one through five)

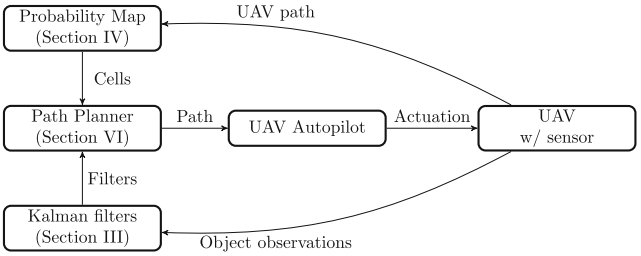


Fig. 2 Control architecture. Notice that the probability map and the Kalman filters are constructed independently from each other and then combined in the path planner

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.

3 Kalman Filters for Moving Objects

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

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

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 [45]. Examples of sensors are a camera, radar, and spectral camera [1]. 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 [46].
- 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, [46] reports a 99.6% accuracy for detecting objects making this assumption not unrealistic.

When modifying the Kalman equation, we use a similar approach to [47]. The measurement equation for a single object is

$$\boldsymbol{y}_{i,k} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{\xi}_{i,k} + \boldsymbol{v}_{i,k} \tag{2}$$

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

$$p(\boldsymbol{v}_{i,k}|\mu_{i,k}) = \begin{cases} \mathcal{N}([0 \quad 0]^T, \boldsymbol{R}), & \mu_{i,k} = 1 \\ \mathcal{N}([0 \quad 0]^T, \sigma^2 \boldsymbol{I}), & \mu_{i,k} = 0 \end{cases} \tag{3}$$

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 [47], 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 [47]. The a priori step becomes (here we drop the subscript i for notational simplicity)

$$\hat{\boldsymbol{\xi}}_{k+1}^{\text{priori}} = \boldsymbol{A}\hat{\boldsymbol{\xi}}_k, \qquad \hat{\boldsymbol{\xi}}_0 = \begin{bmatrix} \boldsymbol{y}_0 & 0 & 0 \end{bmatrix}^T, \tag{4a}$$

$$\boldsymbol{P}_{k+1}^{\text{priori}} = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^T + \Delta_t \boldsymbol{Q}, \quad \boldsymbol{P}_0 = \begin{bmatrix} \boldsymbol{R} & v_{\text{R}}\Delta_t \boldsymbol{R} \\ v_{\text{R}}\Delta_t \boldsymbol{R} & v_{\text{R}}\boldsymbol{I} \end{bmatrix} \tag{4b}$$

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

$$\boldsymbol{K} = \boldsymbol{P}_{k+1}^{\text{priori}}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_{k+1}^{\text{priori}}\boldsymbol{C}^T + \boldsymbol{R})^{-1} \tag{5a}$$

$$\hat{\boldsymbol{\xi}}_{k+1}^{\text{post}} = \hat{\boldsymbol{\xi}}_{k+1}^{\text{priori}} + \mu_k \boldsymbol{K}(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{\xi}}_{k+1}^{\text{priori}}) \tag{5b}$$

$$\boldsymbol{P}_{k+1}^{\text{post}} = \boldsymbol{P}_{k+1}^{\text{priori}} - \mu_k \boldsymbol{K}\boldsymbol{C}\boldsymbol{P}_{k+1}^{\text{priori}} \tag{5c}$$

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 Eq. 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.

4 Probability Map

To keep track of the movements of the UAV in the open area, we use a probability map. 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 7, we use a circular FOV and a cell side length equal to the FOV radius). The setup is illustrated in Fig. 3.

We use probability to describe the most likely location of objects in the area. We define $p_i(t)$ to be the probability

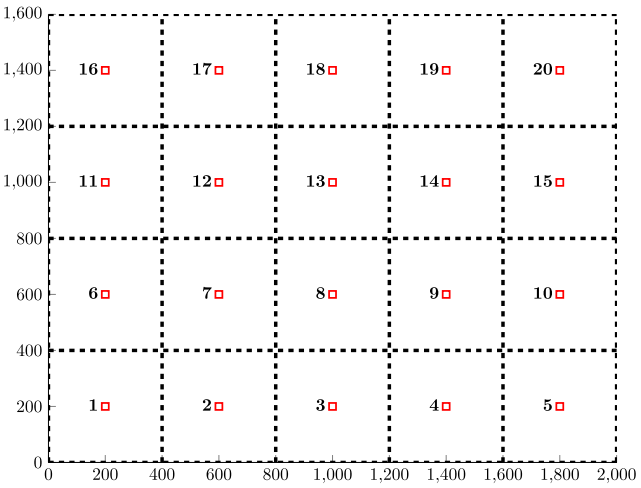


Fig. 3 Illustration of area divided into a grid. The cells are numbered one to twenty and marked with a red square

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*. 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

$$p_0(t) = 1 - \left(\frac{n_{\text{cell}} - 1}{n_{\text{cell}}}\right)^{n_{\text{unknown}}(t)} \tag{6}$$

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 Fig. 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 Fig. 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

$$\dot{p}_i(t) = -e^{-k_0 d_i^2} p_i(t) + q g_i(t) \tag{7}$$

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 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 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 1 (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 Eq. 4b. Then, we note that the estimation error distance is the sum of two normally distributed random variables, giving it a second-order chi-squared distribution.

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

$$\tilde{\boldsymbol{\xi}}_k = \boldsymbol{\xi}_k - \hat{\boldsymbol{\xi}}_k \tag{8}$$

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:

$$\begin{aligned} t_{\text{NVP}} &= \max_n(n\Delta_t) \\ &\text{s.t.} \\ p(|\tilde{\boldsymbol{s}}_n| \leq \text{FOV}_{radius}) &\leq p_{\text{FOV}} \end{aligned} \tag{9}$$

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

$$\boldsymbol{P}_k = \begin{bmatrix} \sigma_{11,k}^2 & \sigma_{12,k} & \sigma_{13,k} & \sigma_{14,k} \\ \sigma_{12,k} & \sigma_{22,k}^2 & \sigma_{23,k} & \sigma_{24,k} \\ \sigma_{13,k} & \sigma_{23,k} & \sigma_{33,k}^2 & \sigma_{34,k} \\ \sigma_{14,k} & \sigma_{24,k} & \sigma_{34,k} & \sigma_{44,k}^2 \end{bmatrix} \tag{10}$$

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

where $\boldsymbol{Q}_k$ is the discretized version of the $\boldsymbol{Q}$ matrix from Eq. 1. Notice that both matrices are symmetric.

We can now derive the following result (which is an extension of Theorem 1 from [44]).

Theorem 1 (Necessary Visitation Period) *Given an object characterized by* Eq. 1 *and a Kalman filter for estimating the state of the object given by* Eqs. 4 *and* 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*

$$\varphi(n; \boldsymbol{\lambda}_a) = 0 \tag{12a}$$

$$\varphi(n; \boldsymbol{\lambda}_b) = 0 \tag{12b}$$

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* Eqs. 10 *and* 11 *with initial condition given by* Eq. 4b*.*

Then, if the sensor takes a measurement at the estimated position of the object given by Eq. 4a *at the time*

$$t_{nvp} = \min(n_1, n_2)\Delta_t, \tag{13}$$

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 Eq. 4b, which we apply recursively to arrive at

$$\boldsymbol{P}_{k+n} = \boldsymbol{A}^n \boldsymbol{P}_k (\boldsymbol{A}^n)^T + \sum_{i=0}^{n-1} \boldsymbol{A}^i \boldsymbol{Q} (\boldsymbol{A}^i)^T \tag{14}$$

where

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

Furthermore, following the notation from Eq. 10, we calculate the upper corner of the matrix $\boldsymbol{P}_{k+n}$

$$\sigma_{11,k+n}^2 = \sigma_{11,k}^2 + 2n\Delta_t \sigma_{13,k} + (n\Delta_t)\sigma_{33,k} + \textstyle\sum_{i=0}^{n-1}[q_{11} + 2i\Delta_t q_{13} + (i\Delta_t)^2 q_{33}] \tag{15}$$

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

$$\sigma_{11,k+n}^2 = \sigma_{11,k}^2 + 2n\Delta_t \sigma_{13,k} + (n\Delta_t)^2 \sigma_{33,k} + \frac{1}{3} q_{33} \Delta_t^2 n^3 + (-\frac{1}{2} q_{33} \Delta_t^2 + q_{13}\Delta_t) n^2 + (\frac{1}{6} q_{33} \Delta_t^2 - q_{13}\Delta_t + q_{11}) n \tag{16}$$

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

$$\sigma_{22,k+n} = f(\sigma_{22,k}, \sigma_{24,k}, \sigma_{44,k}, q_{44}, q_{24}, q_{22}, \Delta_t) \tag{17}$$

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

$$|\text{FOV}_{\text{radius}}| \geq |\tilde{\boldsymbol{s}}_{k+n}| \tag{18}$$

$$\text{FOV}_{\text{radius}}^2 \geq \tilde{x}_{k+n}^2 + \tilde{y}_{k+n}^2. \tag{19}$$

The variance of error in both directions is given by Eq. 16. If we let $\sigma_{\text{pos},k+n}^2 = \max(\sigma_{11,k+n}^2, \sigma_{22,k+n}^2)$ and divide both sides of Eq. 19 by it, we get

$$\frac{\text{FOV}_{\text{radius}}^2}{\sigma_{\text{pos},k+n}^2} \geq \frac{\tilde{x}_{k+n}^2}{\sigma_{\text{pos},k+n}^2} + \frac{\tilde{y}_{k+n}^2}{\sigma_{\text{pos},k+n}^2} \tag{20}$$

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

$$\chi_2^2 \geq \frac{\tilde{x}_{k+n}^2}{\sigma_{\text{pos},k+n}^2} + \frac{\tilde{y}_{k+n}^2}{\sigma_{\text{pos},k+n}^2} \tag{21}$$

Finally, we can combine this equation with Eq. 20 to get an expression for the maximum variance

$$\sigma^2_{\text{pos},k+n} = \frac{\text{FOV}_{\text{radius}}}{\chi^2_2} \tag{22}$$

Combine Eqs. 16 and 17 with Eq. 22 to arrive at Eq. 1. If we use the minimum of the solutions to the two equations, we are guaranteed that Eq. 21 will hold.

Furthermore, both Eq. 1 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

$$\Delta = 18\alpha\beta\gamma\delta - 4\beta^3\delta + \beta^2\gamma^2 - 4\alpha\gamma^3 - 27\alpha^2\delta^2 \tag{23}$$

When the discriminant is negative, the cubic function will have only one real and two complex conjugated solutions [48]. Since we must choose all elements of $\boldsymbol{Q}_k > 0$ and $\Delta_t > 0$, and we use initial conditions given by Eq. 4b, the discriminant is always negative, and thus the cubic functions in Eq. 1 will each have only one real solution, n_1 and n_2, respectively. □

6 Search and Track Algorithm

In this section, we present the path planning algorithm from Fig. 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 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 3 and 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 shows an example of the different algorithm steps.

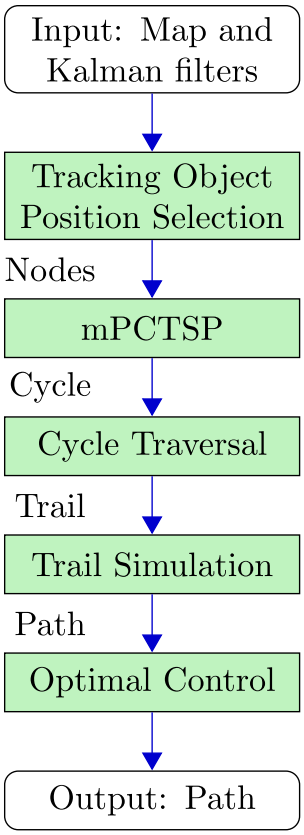


Fig. 4 Flowchart for the search and track algorithm

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 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 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.

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 [49]).

$$\max \sum_{i=n_{\text{object}}+1}^{n} x(i)p(i) \tag{24a}$$

s.t.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} y(i,j)d(i,j) \leq d_{\max} \tag{24b}$$

$$\sum_{j=1}^{n} y(j,i) = 2, \ \sum_{j=1}^{n} y(i,j) = 2 \quad i \in [1, n_{\text{object}}] \tag{24c}$$

$$\sum_{j=1}^{n} y(j,i) = 2x(i), \ \sum_{j=1}^{n} y(i,j) = 2x(i)$$
$$i \in [n_{\text{object}} + 1, n] \tag{24d}$$

$$\sum_{i,j \in \boldsymbol{E}_s} y(i,j) = |\boldsymbol{S}| - 1 \quad \forall \quad |\boldsymbol{S}| = 2, 3, ..., n_{\text{cycle}} \tag{24e}$$

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 Eq. 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 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, Eq. 24a, maximizes the probability of detection of objects from the open area. The constraint, Eq. 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 (24c), while cell nodes must be visited only if they are in the cycle (24d). The last constraint, Eq. 24e, is a subcycle elimination. See [49], 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 finds a trail from the UAV's position to the object's position. This is achieved by replacing constraint (24c) with

$$\sum_{j=1}^{n} y(j,1) = 0, \sum_{j=1}^{n} y(1,j) = 1 \tag{25a}$$

$$\sum_{j=1}^{n} y(j,n) = 1, \sum_{j=1}^{n} y(n,j) = 0 \tag{25b}$$

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.

6.3 Cycle Traversal

The cycle for the mPCTSP algorithm does not take the UAV's position into consideration. We formulate an optimization problem to decide with which direction and node to start traversal. A *cycle traversal* is a *trail* (see introduction of Section 6). If the cycle has length n_{cycle}, there are $2n_{\text{cycle}}$ trails (depending on which node is visited first and the direction in which the UAV traverses the cycle). For example, given the map illustrated in Fig. 3, and thay the mPCTSP produces a cycle of node [2, 8, 12]. The possible trails are then [2, 8, 12], [2, 12, 8], [8, 12, 2], [8, 2, 12], [12, 2, 8], and [12, 8, 2].

Let each traversal be numbered $j \in [1, 2n_{\text{cycle}}]$ and use the notation $\textbf{trail}^j()$ with subscript t and p to note the time, $t \in \mathbb{R}$, and position, $p \in \mathbb{R}^2$, for trail number j. For example, the visit time for node i on trail j is noted as $\text{trail}_t^j(i)$ with position $\text{trail}_p^j(i)$.

Cell nodes have a stationary position, which will, therefore, be the same for each traversal. The object nodes are moving, and thus their position will depend on when they are visited. To calculate the time and position of the object nodes, we let $\text{trail}_t^j(0) = 0$ and $\text{trail}_p^j(0) = \text{UAV}_{\text{pos}}$ and apply the following iterative formula

$$a(\text{trail}_t^j(i+1))^2 + b\text{trail}_t^j(i+1) + c = 0 \tag{26a}$$

where

$$a = |U|^2 - |\hat{\boldsymbol{\xi}}^v(i)|^2$$
$$b = -2(\begin{bmatrix} 1 & 1 \end{bmatrix}\hat{\boldsymbol{\xi}}^p(i) - \text{trail}_p^j(i)\hat{\boldsymbol{\xi}}^v(i))$$
$$c = -|\hat{\boldsymbol{\xi}}^p(i) - \text{trail}_p^j(i)|^2$$

$$\text{trail}_p^j(i+1)^T = \hat{\boldsymbol{\xi}}^p(i) + \text{trail}_t^j(i)\hat{\boldsymbol{\xi}}^v(i) \tag{26b}$$
$$\forall i = [1, \ldots, n_{\text{trail}}]$$

where U is the UAV's velocity. The velocity and position of node i are $\hat{\boldsymbol{\xi}}^p(i)$ and $\hat{\boldsymbol{\xi}}^v(i)$. Notice that we ignore the dynamics of the UAV, i.e. Eq. 29b. The iterative formula is derived by considering the UAV and object to be two moving points where we control the UAV without movement restriction and calculate their intersection.

Furthermore, let $\text{trail}^j_{\Delta\psi_1}$ and $\text{trail}^j_{\Delta\psi_2}$ be the *difference* between the UAV's heading and the angle between the UAV and the first and second nodes of trail j.

This enables us to formulate the following optimization problem, which we use to select the trail.

$$c_1 = \min_j \max(0, (|\text{trail}^j_{\Delta\psi_1}| - \tfrac{\pi}{2})^2) \tag{27a}$$

$$c_2 = \min_j \max(0, (|\text{trail}^j_{\Delta\psi_2}| - \tfrac{\pi}{2})^2) \tag{27b}$$

$$c_3 = \min_j \sum_{i=1}^{n_{\text{trail}}} \left(t_{\text{ideal}}(i) - \text{trail}^j_t(i) \right)^2 \tag{27c}$$

where $\boldsymbol{t}_{\text{ideal}}$ is a vector containing the desired time at which the UAV should visit each node. The object nodes have this value set in Section 6.4, while cell nodes are not assigned a desired time for visiting. Therefore, they are set to $t_{\text{ideal}}(i) = \text{trail}^j_t(i)$.

This is a multi-layered optimization. The first two layers c_1 and c_2 typically have multiple non-unique solutions. These two layers are included to make sure the UAV does not change direction. The SaT algorithm is run often, and these layers ensure continuity of the solution. The final layer c_3 typically has a unique solution.

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 Eq. 29b, then

$$\begin{aligned} \psi_{\text{desired}} &= \text{atan2}(y_{\text{desired}} - y, x_{\text{desired}} - x) \\ u &= \psi_{\text{desired}} - \psi \end{aligned} \tag{28}$$

where atan2 is a four-quadrant arctangent function.

We use a Runge-Kutta method [50] for the simulation in which we simulate the UAV equation, Eq. 29b, and all the cell equations, Eq. 7.

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 Fig. 5c and d.

To limit the size of the problem, we include only the nodes used by the trail simulation in Section 6.4. In addition, all nodes are treated as simplified cell nodes, meaning that we use Eq. 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

$$\min_{u(\cdot)} \int_0^{t_{\text{end}}} \mu \sum_{i=1}^{n_{\text{node,ocp}}} p_i^2(t) + u^2(t) dt \tag{29a}$$

s.t.

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

$$\dot{\boldsymbol{z}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} U\cos(\psi) \\ U\sin(\psi) \\ u \end{bmatrix} \tag{29b}$$

$$-u_{\text{lim}} \leq u \leq u_{\text{lim}} \tag{29c}$$

$$-\eta \leq x \leq X + \eta \tag{29d}$$

$$-\eta \leq y \leq Y + \eta \tag{29e}$$

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 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 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 (29d) and (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.

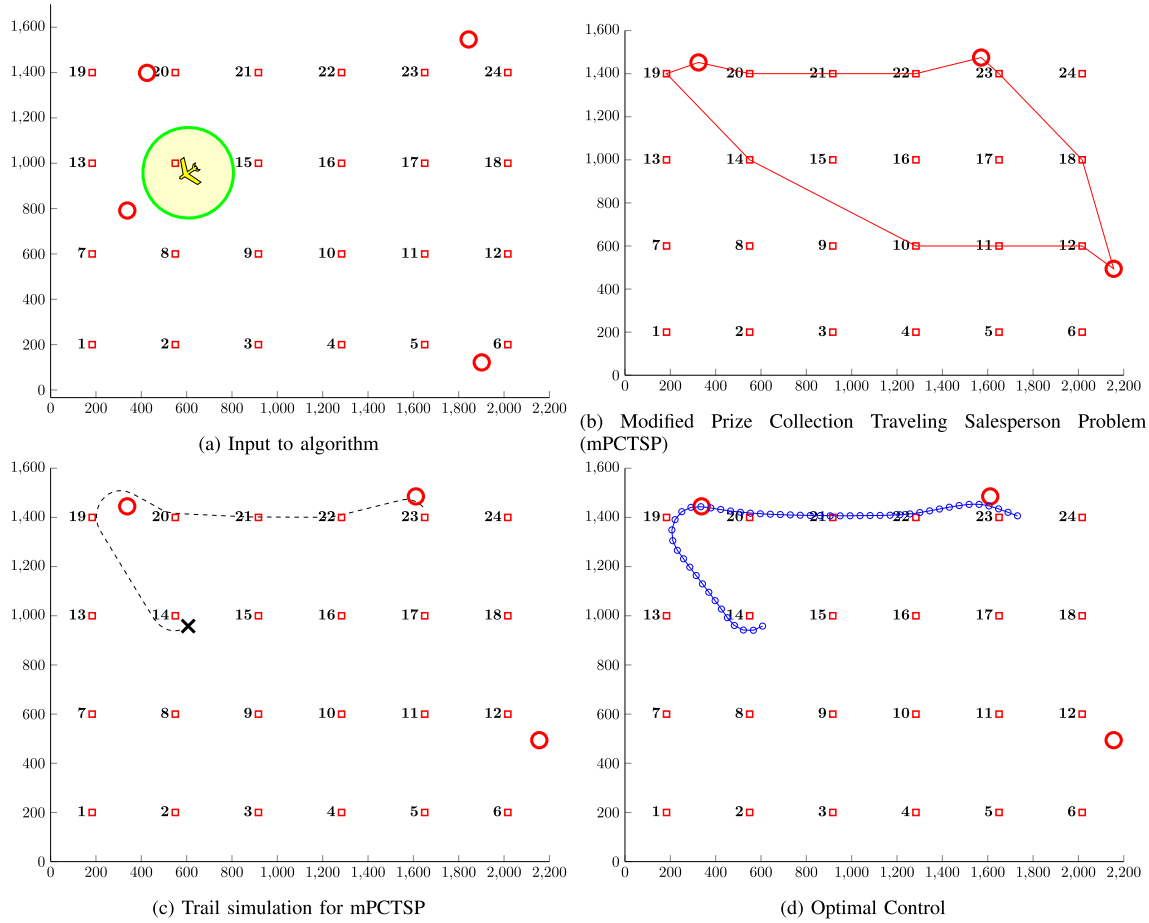
6.6 Tunable Parameters

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

$$k_0 = 5 \times 10^{-4} \tag{30}$$

The other parameter, q, decides the rate at which any cell reaches the default probability in the absence of observation.



(a) Input to algorithm

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

(c) Trail simulation for mPCTSP

(d) Optimal Control

Fig. 5 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

It can be set in relation to the expected velocity of the moving objects. We use the following value

$$q = 0.01 \tag{31}$$

In preparing the Kalman filters for the mPCTSP algorithm, we need to select a probability of redetection for an object at its estimate position, which we will denote p_{ideal}. 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 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

$$p_{\text{ideal}} = \min\left(0.85, 1 - \frac{1}{n_{\text{cell}}}\sum_{i=0}^{n_{\text{cell}}} p(i)\right) \tag{32}$$

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 6.1). We use the following heuristic to set it

$$d_{\max} = \begin{cases} \max(U t_{\text{ideal}}(1), |\text{UAV}_{\text{pos}} - \hat{\boldsymbol{\xi}}^p(1)|) & \text{if } n_{\text{object}} = 1 \\ d_{\text{tsp}} + 1.5 \times U \max(0, c_{3,\text{avg}}(\text{TSP})) & \text{else} \end{cases} \tag{33}$$

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 Eq. 27c 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 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

$$\mu = \frac{6}{n_{\text{cell}}} \tag{34}$$

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 6.2 was written using YALMIP [51] and solved using IBM's CPLEX [52]. To simulate the differential equations from Section 6.4, we used the Matlab integrated method ode45 [53]. For the optimal control problem from Section 6.5, we wrote the formulation using CasADi [54] and solved it using interior point solver IPOPT [55] with the linear solver mumps.

7 Simulation

To validate the algorithm from Section 6, we compare it to multiple base cases and an omniscient case which utilizes the actual position and velocity of each object.

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 Eq. 1, and an available UAV with a limited FOV to search and track the objects. This is illustrated in Fig. 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 let the objects behave as the snake from the popular arcade game [56], 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 Fig. 6.

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

$$H = \frac{1}{n_{\text{object}} n_{\text{sim}}} \sum_{i=1}^{n_{\text{object}}} \sum_{k=1}^{n_{\text{sim}}} h_i(k) \tag{35a}$$

where

$$h_i(k) = \begin{cases} 1 & \tilde{s}_i(k) \in \text{FOV} \\ 0 & \text{else} \end{cases} \tag{35b}$$

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

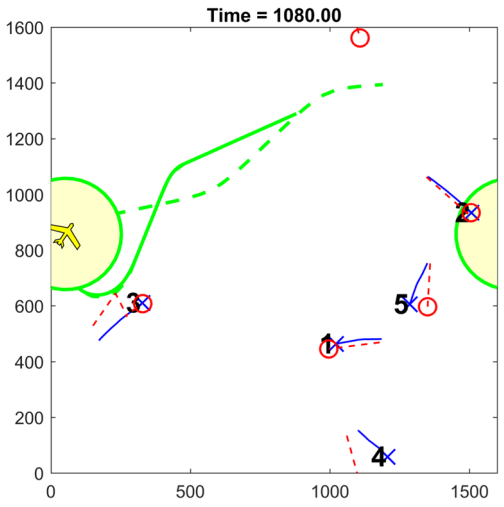


Fig. 6 Monitoring a 1600 x 1600 m^2 area. The UAV is currently sensing object number 2 across the edge of the area. The estimated position of object 4, red circle, has moved across from one side of the area to the other

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 $\tilde{s}_i(k)$. 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.

7.2 Base Cases

The first base case sets the UAV to follow a simple straight line patrol, as illustrated in Fig. 7. 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 6.5.

In the second base case, we have the UAV following a looping pattern, which is shown in Fig. 8. This performs slightly better than the straight line patrol.

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.

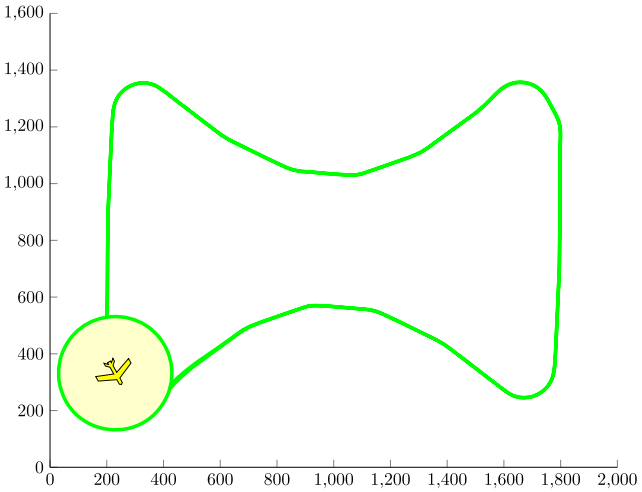


Fig. 8 Base case 2. Looping patrol of an example area of size 2000 × 1600 m^2

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

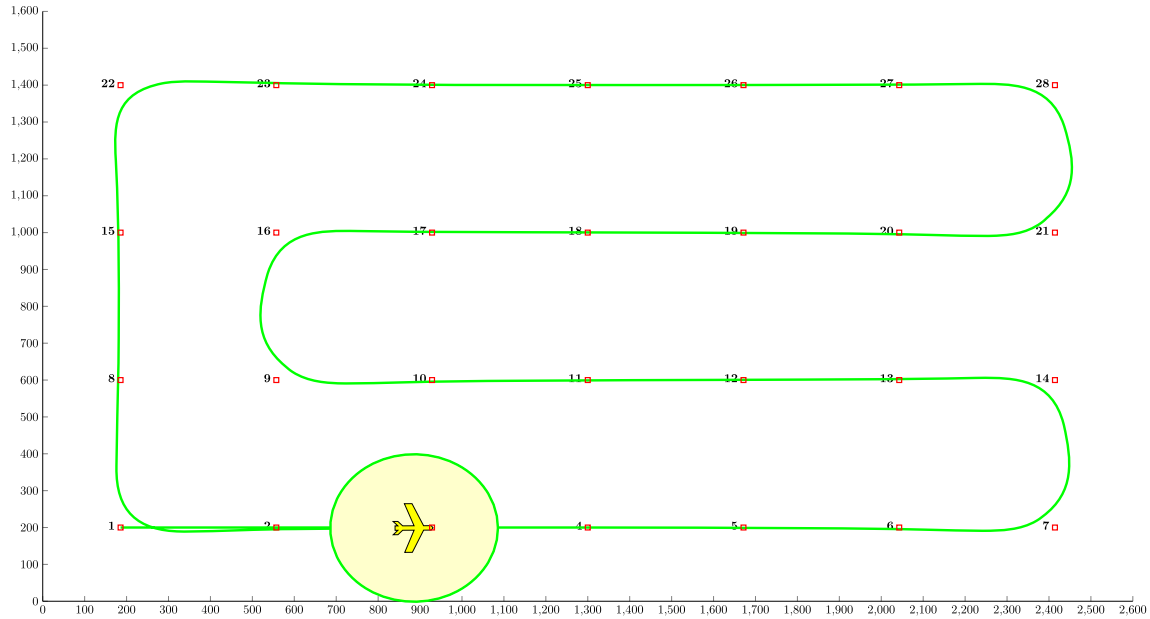


Fig. 7 Base case 1. Straight line patrol of an example area of size 2600 × 1600 m^2

to zero. Then, the algorithm minimizes the following two layered optimization functions

$$A) \quad \min_j \sum_{i=1}^{n_{\text{trail}}} (t_{\text{last}}(i) - \text{trail}_t^j(i))^2 \tag{36a}$$

$$B) \quad \min_{\text{trail}(\cdot)} \text{Tour Length} \tag{36b}$$

The algorithm works by trying all possible permutations of the object visitation sequence calculated using Eq. 6.3. If more than one sequence has the same value for objective A, the shortest trail is selected (objective B).

7.4 Results

The parameters used in the simulation are given in Table 1. 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 [57].

Table 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]

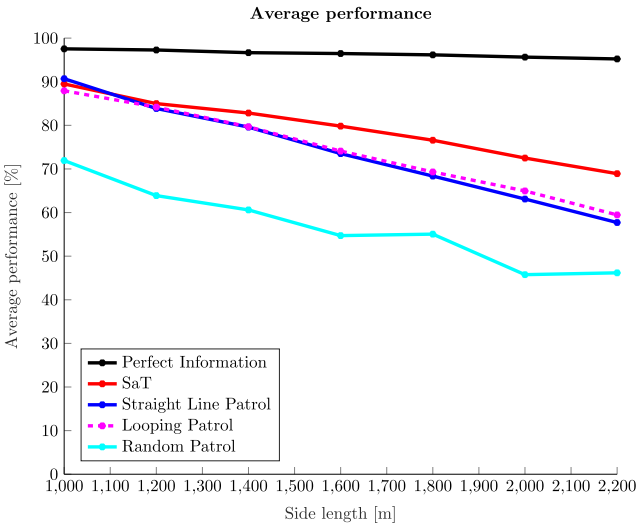


Fig. 9 Comparison of SaT algorithm to base cases and the case of utilizing perfect information for areas of varying size using 5 objects

Figure 9 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 10 illustrates the results of 30 simulations for a constant map size (2000m × 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.

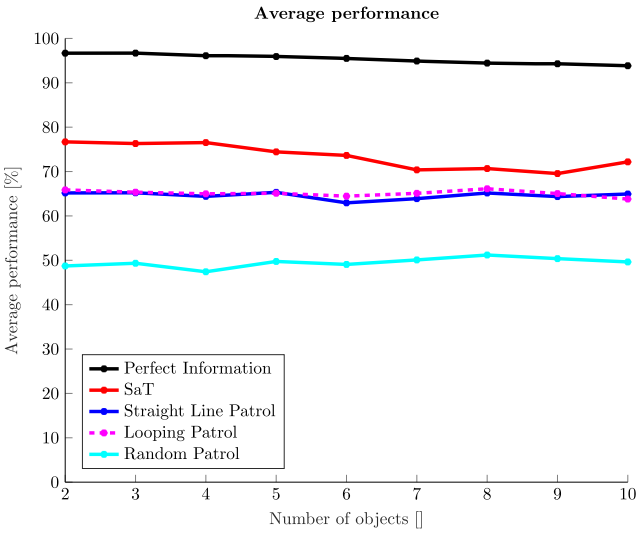


Fig. 10 Comparison of SaT algorithm to base cases and utilizing perfect information for varying numbers of objects using an area of size 2000 × 1600 m^2

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 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 Eq. 32 from Section 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 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 Fig. 5, 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.

9 Conclusion

In this paper, 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.

10 Future Work

Future work will include:

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

References

1. Eik, K.: Review of experiences within ice and iceberg management. J. Navig. **61**(4), 557–572 (2008)
2. Lešinskis, I., Pavlovičs, A.: 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 (2011)

3. Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I.L., Ruess, F., Suppa, M., Burschka, D.: Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. IEEE Robotics & Automation Magazine **19**(3), 46–56 (2012)
4. Goodrich, M.A., Morse, B.S., Gerhardt, D., Cooper, J.L., Quigley, M., Adams, J.A., Humphrey, C.: Supporting wilderness search and rescue using a camera-equipped mini uav. Journal of Field Robotics **25**(1-2), 89–110 (2008)
5. Glade, D.: Unmanned Aerial Vehicles: Implications for military operations, DTIC Document, Tech. Rep. (2000)
6. Eggers, J., Draper, M.H.: Multi-UAV control for tactical reconnaissance and close air support missions: operator perspectives and design challenges. In: Proceedings NATO RTO Human Factors and Medicine Symp. HFM-135. NATO TRO, Neuilly-sur-Siene. CEDEX, Biarritz (2006)
7. DeSena, J.T., Martin, S.R., Clarke, J.C., Dutrow, D.A., Kohan, B.C., Kadar, I.: Decentralized closed-loop collaborative surveillance and tracking performance sensitivity to communications connectivity. In: Proceedings of SPIE Vol, vol. 8745, pp. 874507–1 (2013)
8. Girard, A.R., Howell, A.S., Hedrick, J.K.: Border patrol and surveillance missions using multiple unmanned air vehicles. In: 43rd IEEE Conference on decision and control, 2004. CDC. vol. 1. IEEE, pp. 620–625 (2004)
9. Isenor, A.W., Allardb, Y., Lapinskia, A.-L.S., Demersb, H., Radulescub, D.: Coordinating UAV information for executing national security-oriented collaboration, vol. 9248 (2014)
10. Schrijver, A.: On the history of combinatorial optimization (till 1960). Handbooks in Operations Research and Management Science **12**, 1–68 (2005)
11. Dantzig, G., Fulkerson, R., Johnson, S.: Solution of a large-scale traveling-salesman problem. J. Oper. Res. Soc. Am. **2**(4), 393–410 (1954)
12. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: The traveling salesman problem: A computational study (2011)
13. Balas, E.: The prize collecting traveling salesman problem. Networks **19**(6), 621–636 (1989)
14. Gutin, G., Punnen, A.P.: The traveling salesman problem and its variations (2006)
15. Pesch, H.J., Plail, M., Munich, D.: The maximum principle of optimal control: a history of ingenious ideas and missed opportunities. Control and Cybernetics **38**(4A), 973–995 (2009)
16. Betts, J.T.: Practical methods for optimal control and estimation using nonlinear programming. SIAM (2010)
17. Biegler, L.T.: Nonlinear programming: concepts, algorithms and applications to chemical processes. SIAM (2010)
18. Stone, L.D.: Or forum—what's happened in search theory since the 1975 Lanchester prize? Oper. Res. **37**(3), 501–506 (1989)
19. Koopman, B.O.: The theory of search. i. kinematic bases. Oper. Res. **4**(3), 324–346 (1956)
20. Koopman, B.O.: The theory of search. ii. target detection. Oper. Res. **4**(5), 503–531 (1956)
21. Koopman, B.O.: The theory of search: Iii. the optimum distribution of searching effort. Oper. Res. **5**(5), 613–626 (1957)
22. Walton, C.L., Gong, Q., Kaminer, I., Royset, J.O.: Optimal motion planning for searching for uncertain targets. IFAC Proceedings Volumes **47**(3), 8977–8982 (2014)
23. Stone, L.D., Royset, J.O., Washburn, A.R.: Optimal search for moving targets. Springer (2016)
24. Golden, B.L., Raghavan, S., Wasil, E.A.: The vehicle routing problem: latest advances and new challenges. Springer Science & Business Media (2008)
25. Oberlin, P., Rathinam, S., Darbha, S.: Today's traveling salesman problem. IEEE Robotics & Automation Magazine **17**(4), 70–77 (2010)
26. Smith, S.L.: Task allocation and vehicle routing in dynamic environments. University of California Santa Barbara (2009)
27. Yao, M., Zhao, M.: Unmanned aerial vehicle dynamic path planning in an uncertain environment. Robotica **33**(3), 611–621 (2015)
28. Meng, W., He, Z., Su, R., Yadav, P.K., Teo, R., Xie, L.: Decentralized multi-UAV flight autonomy for moving convoys search and track. IEEE Trans. Control Syst. Technol. **25**(4), 1480–1487 (2017)
29. Zhao, C., Zhu, M., Liang, H., Wu, Z.: The sustainable tracking strategy of moving target by UAV in an uncertain environment, Control Conference (CCC), 2016 35th Chinese, pp. 5641–5647 (2016)
30. Furukawa, T., Bourgault, F., Lavis, B., Durrant-Whyte, H.F.: Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. IEEE International Conference on Robotics and Automation (ICRA), pp. 2521–2526 (2006)
31. Tian, X., Bar-Shalom, Y., Pattipati, K.R.: Multi-step look-ahead policy for autonomous cooperative surveillance by UAVs in hostile environments. In: 47th IEEE Conference on decision and control, 2008. CDC 2008, pp. 2438–2443 (2008)
32. Kassas, Z.M., Arapostathis, A., Humphreys, T.E.: Greedy motion planning for simultaneous signal landscape mapping and receiver localization. IEEE Journal of Selected Topics in Signal Processing **9**(2), 247–258 (2015)
33. Peterson, C.K., Newman, A.J., Spall, J.C.: Simulation-based examination of the limits of performance for decentralized multi-agent surveillance and tracking of undersea targets, SPIE Defense+ Security, pp. 90910F–90910F (2014)
34. Pitre, R.R., Li, X.R., Delbalzo, R.: UAV Route planning for joint search and track missions—an information-value approach. IEEE Trans. Aerosp. Electron. Syst. **48**(3), 2551–2565 (2012)
35. Sinha, A., Kirubarajan, T., Bar-Shalom, Y.: Autonomous ground target tracking by multiple cooperative UAVs. In: 2005 IEEE Aerospace Conference, pp. 1–9 (2005)
36. Sinha, A.: Autonomous surveillance by multiple cooperative UAVs. In: Proceedings of SPIE Vol, vol. 5913, pp. 59131V–1 (2005)
37. Mavrommati, A., Tzorakoleftherakis, E., Abraham, I., Murphey, T.D.: Real-time area coverage and target localization using receding-horizon ergodic exploration, arXiv:1708.08416 (2017)
38. Oispuu, M., Sciotti, M., Charlish, A.: Air route selection for improved air-to-ground situation assessment. SPIE Defense, Security, and Sensing, pp. 87420M–87420M (2013)
39. Vanegas Alvarez, F.: Uncertainty based online planning for UAV missions in GPS-denied and cluttered environments, Ph.D. dissertation Queensland University of Technology (2017)
40. Tichavsky, P., Muravchik, C.H., Nehorai, A.: Posterior cramér-Rao bounds for discrete-time nonlinear filtering. IEEE Trans. Signal Process. **46**(5), 1386–1396 (1998)
41. Hernandez, M.: Performance bounds for target tracking: computationally efficient formulations and associated applications. Integrated Tracking, Classification, and Sensor Management: Theory and Applications, pp. 255–310 (2012)
42. Esmailifar, S.M., Saghafi, F.: Cooperative localization of marine targets by UAVs. Mech. Syst. Signal Process. **87**, 23–42 (2017)
43. Koohifar, F., Kumbhar, A., Guvenc, I.: Receding horizon multi-UAV cooperative tracking of moving rf source. IEEE Commun. Lett. **21**(6), 1433–1436 (2017)
44. Albert, A., Imsland, L.: Performance bounds for tracking multiple objects using a single UAV, pp. 1539–1546 (2017)
45. Gelb, A.: Applied optimal estimation. MIT press (1974)
46. Leira, F., Johansen, T.A., Fossen, T.I.: Automatic detection classification and tracking of objects in the ocean surface from uavs using a thermal camera (2015)

47. Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M.I., Sastry, S.S.: Kalman filtering with intermittent observations. IEEE Trans. Autom. Control **49**(9), 1453–1464 (2004)
48. Irving, R.S.: Integers, polynomials and rings: A course in algebra. Springer Science & Business Media (2003)
49. Chen, D.-S., Batson, R.G., Dang, Y.: Applied integer programming: Modeling and solution. Wiley (2010)
50. Dormand, J.R., Prince, P.J.: A family of embedded runge-Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980)
51. Löfberg, J.: YALMIP: A Toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD Conference (2004)
52. IBM Ilog CPLEX optimization studio CPLEX user's manual, http://www.ibm.com, 2015, accessed: 2015-09-12
53. Shampine, L.F., Reichelt, M.W.: The matlab ODE suite. SIAM J. Sci. Comput. **18**(1), 1–22 (1997)
54. Andersson, J.: 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 (2013)
55. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
56. Punyawee, A., Panumate, C., Iida, H.: Finding comfortable settings of snake game using game refinement measurement. In: International Conference on Computer Science and its Applications, pp. 66–73 (2016)
57. Fossen, T.I.: Handbook of marine craft hydrodynamics and motion control. Wiley (2011)

Anders Albert received his MSc degree in Engineering Cybernetics in 2013 at Norwegian University of Science and Technology, Trondheim, Norway. Currently he is pursuing a Ph.D. degree at the same department. The research topic of his Ph.D. thesis is "Mission and Path Optimization Sensor Network". His main research interests are optimization and path planning with the application of iceberg search and tracking.

Lars Imsland received the Ph.D. degree in electrical engineering from the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. He was a Visiting Research Scholar with the Institute for Systems Theory in Engineering, University of Stuttgart, Stuttgart, Germany, for part of his Ph.D. studies. He was a Post-Doctoral Researcher with NTNU, a Research Scientist with SINTEF ICT, and a Specialist with Cybernetica AS, before becoming a full-time Professor of Control Engineering with NTNU in 2009. His current research interests include theory and application of nonlinear optimizing control and estimation. Examples of applications are within the oil and gas industry (both drilling and production), active safety in the automotive industry, and the use of mobile sensor networks for monitoring of local ice features.