DISSERTATION

SPANNING SENSOR RESOURCE MANAGEMENT

Submitted by

Lucas W. Krakow

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2018

Doctoral Committee:

Advisor: Edwin K. P. Chong

Patrick Burns
Ali Pezeshki
Jie Luo

ABSTRACT

SPANNING SENSOR RESOURCE MANAGEMENT

This paper presents multiple applications of sensor resource management. The general focus entails two chapters on adaptive estimation of time-varying sparse signals and three chapters exploring autonomous control of unmanned aerial vehicles (UAVs) sensor platforms employed for target tracking. All of the included applications are posed as decision control problems formulated in the rigorous framework of a partially observable Markov decision process (POMDP) and solution methods based on Bellman's equation are exercised, generating adaptive control policies for action selections in the given scenarios. Specifically, the rollout optimization method is administered in the cases of signal estimation under the objective of maximizing the information gain about the unknown sparse signal. For the UAV sensor platform control, nominal belief-state optimization (NBO) is employed for control selection for optimizing objectives including target-tracking error, surveillance performance and fuel efficiency. The empirical studies in each investigation present evidence that non-myopic solution methods, accounting for both the immediate and future costs of the current action choices, provide performance gains for these scenarios.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Sensor resource management is the monitoring and control of sensor capabilities and assets with the sole focus of optimizing the objective defined for the sensor system. The overarching objective is the increase of the gain of information ascertained from the sensor(s). Control of the degrees of freedom of sensor platforms is required due to the constraints that naturally arise in applications. Such constraints range from hardware limitations (e.g., battery power, platform maneuverability, sensor modes, etc.) to scenario specifics (e.g., terrain, covertness requirements, etc.).

Simple sensor resource management approaches account for constraints through a priori planning. For example, a fixed flight pattern for surveilling an area of interest (AOI) considers the operational longevity of an aircraft and the coverage of the known terrain in the AOI. However, dynamic components of the scenario such as wind speeds can adversely affect the performance by increasing power requirements for the fixed turn radius in the predefined flight plan. This suboptimal resource management can be vastly improved through adaptive control, determining resource expenditures based on real-time feedback from the system.

The studies included in the following chapters focus solely on adaptive control of sensor resources for several scenarios and objectives in question. We use the unifying framework of a Partially Observable Markov decision process (POMDP) to succinctly define the said scenarios. Defining the POMDP formulation allows the application of POMDP solution theory based on Bellman's equation [1]. The diversity of the following studies emphasizes the benefits of lookahead approaches across adaptive compressive sensing (Chapter 2 and 3) and autonomous control of unmanned aerial vehicle sensor platforms (Chapters 4, 5, and 6).

Chapter 2 introduces the problem of adaptively designing compressive measurement matrices for estimating time-varying sparse signals. We formulate this problem as a *Partially Observable Markov Decision Process (POMDP)*. This formulation allows us to use Bellman's principle of

optimality in the implementation of multi-step lookahead designs of compressive measurements. We compare the performance of adaptive vs. traditional non-adaptive designs and study the value of multi-step (non-myopic) vs. 1-step (myopic) lookahead adaptive schemes by introducing two variations of the compressive measurement design problem. In the first variation, we consider the problem of sequentially selecting measurement matrices with fixed dimensions from a pre-specified library of measurement matrices. In the second variation, the number of compressive measurements, i.e., the number of rows of the measurement matrix, is adaptively chosen. Once the number of measurements is determined, the matrix entries are chosen according to a prespecified adaptive scheme. Each of these two problems is judged by a separate performance criterion. The gauge of efficiency in the first problem is the conditional mutual information between the sparse signal support and the measurements. The second problem applies a linear combination of the number of measurements and the conditional mutual information as the performance measure. Through several simulations, we study the effectiveness of different designs in various settings. The primary focus in these simulations is the application of a method known as *rollout*. However, the computational load required for using the rollout method has also inspired us to adapt two data association heuristics to the compressive sensing paradigm. These heuristics show promising decreases in the amount of computation for propagating distributions and searching for optimal solutions.

Chapter 3 follows the formulation of Chapter 2 considering an application of adaptive compressive sensing for estimating time-varying sparse signals. The scenario entails the corruption of the sparse signal by additive observational noise and erasure. We formulate the problem as a partially observable Markov decision process (POMDP) and apply a multi-step lookahead solution technique, *rollout*. To reduce computations involved in the posterior distribution propagation and improve estimates of the unobservable state, we incorporate an approximation adapted from multiple hypothesis tracking. Each action decision selects a fixed size measurement matrix from a predefined library composed of matrices whose rows are members of a Grassmannian packing. The performance of the matrix selections is gauged by the ability to maximize the conditional mu-

tual information between the sparse signal support and the resulting observations. Extending the study from Chapter 2 there is a restriction placed on the repeated usage of vectors from the signal library, representing a requirement for covertness. Through simulation, we compare rollout with an adaptive heuristic and a greedy algorithm.

Chapters 2 and 3 culminate the endeavors into adaptive compressive sensing. Chapters 4, 5, and 6 shift the focus to scenarios concentrated on the autonomous control of UAV sensor platforms. The POMDP formulations in each of these chapters is similar, but each emphasizes a specific need for non-myopic action selection.

Chapter 4 connects the persistent surveillance and target tracking. These tasks are often accomplished through the use of unmanned aerial vehicles (UAVs). In many scenarios both tasks are concurrently required, but are disparate in their area investigation techniques; surveillance requires global exploration, whereas tracking takes a local target-centric view. Thus, the two tend to require different actions when controlling UAVs tasked with their respective objectives. We propose formulating the UAV control problem for concurrent surveillance and target tracking as a partially observable Markov decision process (POMDP) applying a Q-value approximation technique called nominal belief-state optimization (NBO). Using the Probability Hypothesis Density (PHD) filter as the tracking algorithm, we are able to exploit the birth-intensity components combined with non-myopic action selection inspiring the desired accompanying surveillance behaviors.

Chapter 5 investigates the addition of gimbal control for camera aiming. Inexpensive gimbal-mounted cameras have become a viable option for replacing fixed camera systems mounted to a small UAVs for target tracking applications. The gimballed camera adds degrees of freedom, expanding the limited field of view of the camera. This is advantageous but the additional degrees of freedom equate to augmenting the control space to include the gimbal controls. While the addition of gimbal controls is not an insurmountable obstacle, the slew rate of the gimbal axis motors is often limited due to hardware performance capabilities or system definitions. This imposes constraints on the controls. Target-tracking scenarios often include sparsely distributed targets in which long-term tracking performance greatly benefits from non-myopic control. Un-

der these considerations, the gimbal control constraints affect the reachable set of camera-aiming angles, in turn directly impacting the UAV navigation planning. To maximize the benefits provided by the gimbal, we formulate the problem of simultaneous UAV and camera gimbal control as a partially observable Markov decision process (POMDP), applying a Q-value approximation technique called nominal belief-state optimization (NBO). The finite-horizon objective function is simply the expected cumulative tracking error. Furthermore, we formulate a heuristic expected cost-to-go (HECTG) which approximates target-tracking performance over an extended horizon at a low computational cost. Not only does NBO with the HECTG apply to single UAV cases but extends to the coordination of multiple UAVs with gimballed cameras.

Chapter 6 extends the objective to accommodate fuel efficiency optimization during target tracking maneuvers. The ability to effectively track moving targets is a critical capability for future autonomous aircraft. While many methods have been developed for performing target tracking, minimal work has focused on fuel-efficient options to extend mission duration. The ability to tightly track a target is critical for certain missions; however, increased tracking errors can be accepted in certain scenarios to extend endurance. Partially Observable Markov Decision Processes (POMDPs) have been shown to be effective for tracking fixed and moving targets. This paper provides a fuel-efficient option that shows a 10% endurance increase with adequate target tracking. The algorithm provides tracking with a limited field of view fixed sensor that will have limited observations depending on mission requirements. The POMDP formulation proposed in this chapter is robust enough to handle observations while also providing options for improved fuel efficiency. We perform 500 Monte Carlo simulations per configuration to provide statistical confidence in the performance of the algorithm.

Finally, Chapter 7 describes the steps in combining the camera gimbal control from Chapter 5 to assist in the fuel-efficiency goals described in Chapter 6. These two chapters combine naturally; the POMDP formulations are highly related and the solution approaches (NBO) are in sync. We present the modified POMDP components required for this unification. Although the solution methods are the same, their subcomponents also necessitate alterations for the combining of these

two studies. After the problem and the solution method specifics are defined, the outcomes are presented showing the increased fuel efficiency and tracking performance amassed from the added degrees of freedom of the steerable camera.

Chapter 2

Adaptive Estimation of Time-Varying Sparse Signals

Compressive Sensing (CS) concentrates on solving the problem of recovering sparse (or compressible) signals using linear compressive measurements (see, e.g., [2–6]). CS results suggest that a sparse signal can be completely recovered if a sufficient number of measurements, but much lower than the sparse signal dimension, are made in a noiseless environment. In particular, these results show that the exact recovery of the sparse signal can be achieved with a high probability if compressive measurement matrices with random Gaussian or Bernoulli entries are used for measuring the signal.

Since then, numerous methods for designing the compressive measurement matrix have been proposed (see, e.g., [7]–[12]). For example, the work in [11] provides a deterministic structure for the measurement matrix which requires more measurements to recover the sparse signal but the overall computational cost is much lower than the random design. Recently, more realistic settings for the sparse signal recovery problem have been considered (see, e.g., [13]–[15]), opening new horizons for alternative compressive measurement designs.

The traditional CS methods are non-adaptive in the sense that the measurement matrix is determined in advance, prior to receiving any measurements of the sparse signal. The recent theoretical result in [16] shows that, in a certain asymptotic sense, adaptive designs have limited performance advantage over traditional non-adaptive schemes. On the other hand, several works (see, e.g., [17]–[21]) have proposed adaptive designs for certain scenarios, showing (non-asymptotic) improvements resulting from adaptation under these scenarios.

The above works all assume a *static* sparse signal. This means that over time, the locations and values of the nonzero entries of the sparse signal stay constant. Although this assumption is valid in some applications, in many practical scenarios, considering a static signal seems unrealistic. Examples of time-varying sparse signals arise naturally in target tracking, high-speed video capturing, time-varying multi-path in communication, etc. However, little attention has so far been paid to

the time-varying case (notable exceptions being [22]–[26]), where one would expect adaptation to play a more significant role.

2.1 Problem Description

In this chapter, we study the problem of adaptively designing compressive measurement matrices for estimating time-varying sparse signals. We take a unique perspective that enables melding CS and data association techniques for multi-target tracking: We view the time-varying sparse signal as a representation of the combination of target locations (the vector support) and target values (the non-zero values located at the support indices) of s targets moving over N possible locations according to a discrete motion-model.

At each time step k, the locations and strength values of these targets can be represented by an s-sparse vector $\mathbf{x}_k$ in $\mathbb{R}^N$. We call an N-dimensional signal *s-sparse* if it has at most $s \ll N$ nonzero entries. Moreover, we assume that only the target locations, or equivalently the *support* of the sparse signal $\mathbf{x}_k$, change over time and not the target values. Our goal is to estimate the support of the sparse signal at each time step from compressive measurements.

During each time step k, we collect $1 \leq l_k \leq l_{\max}$ measurements ($l_{\max}$ is a prespecified integer), represented by the l_k-vector $\mathbf{y}_k = [y_1, y_2, \ldots, y_{l_k}]^T$, according to the linear model $\mathbf{y}_k = \mathbf{A}_k\mathbf{x}_k + \mathbf{w}_k$. In this model, $\mathbf{A}_k \in \mathbb{R}^{l_k \times N}$ is a compressive measurement matrix with rows $\mathbf{A}_k(i)$, $i = 1, \ldots, l_k$, and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}_{l_k}, \sigma_w^2 \mathbf{I}_{l_k})$, where $\mathcal{N}(\mathbf{0}_{l_k}, \sigma_w^2 \mathbf{I}_{l_k})$ denotes the l_k-variate normal distribution with the zero mean vector $\mathbf{0}_{l_k}$ and the covariance matrix $\sigma_w^2 \mathbf{I}_{l_k}$, and $\mathbf{I}_{l_k}$ is the $(l_k \times l_k)$ identity matrix. We assume that $\|\mathbf{A}_k(i)\|_2 = 1$ for $i = 1, \ldots, l_k$ and $k = 1, \ldots, m$. Also, we assume that at each time step, the movement of the targets is independent of the measurement matrix $\mathbf{A}_k$ or the number of measurements l_k. Thus, at any time step, multiple targets can move to the same location, effectively reducing the number of nonzero entries in the signal to less than s.

We now introduce two problems of interest. These problems both contain the essence of compressive sensing as described above. At the same time, each problem presents unique goals and

constraints. The differences in these problems ultimately define the type of their favored solution methods.

Problem 1. Fix $l_k = l$ for $k = 1, 2, \ldots, m$, where $1 \leq l \leq l_{\text{max}}$ is a fixed integer. Our goal is to sequentially select measurement matrices $\mathbf{A}_k, k = 1, 2, \ldots, m$, from a prespecified library to maximize a measure of performance for estimating locations of the nonzero entries (support) of the sparse signal $\mathbf{x}_k$ at each time step k. The performance measure for this problem is the conditional mutual information between the measurement vector $\mathbf{y}_k$ and the support of the signal $\mathbf{x}_k$ given the previous measurements.

Problem 2. Here, l_k is not fixed and is, in fact, the action input chosen sequentially at each time step, $k = 1, 2, \ldots, m$. Once the value l_k is chosen, the measurement matrix $\mathbf{A}_k$ is constructed using a prespecified scheme, explained in Section 2.4. The performance criterion in this problem is defined as a weighted sum of the number of measurements l_k and the conditional mutual information between the measurements and the support of the sparse signal at each time step.

The problems introduced above present two variations of support identification for sparse signals. Also, both of these problems lend themselves to adaptive measurement solutions, successively selecting action inputs as observations are collected. We formulate each problem as a finite-horizon partially observable Markov decision process (POMDP) (see [27] and [28]). This formulation enables us to bring to bear Bellman's principle of optimality. Furthermore, the POMDP framework accommodates two categories of adaptive solution methods, allowing their implementation and comparison.

When variations of Problem 1 have been considered in the CS literature, the common solution approach is categorized as a "1-step lookahead" or "myopic" method. In other words, only the performance at the current time step is considered when selecting the compressive measurement matrices. Although Section 2.4 *includes* results from 1-step lookahead methods, our primary focus in this chapter is the application of "multi-step lookahead" solutions.

It is important to realize that multi-step lookahead methods can perform better than myopic solutions only when there is a benefit in accounting for long-term effects. Several factors play a

role in providing such benefits. The performance criterion of the problem and the action space restrictions are examples of such factors. In this chapter, throughout several simulations, we show that although adaptive designs outperform the non-adaptive schemes, the performance criterion and the action space of Problem 1 preclude the multi-step lookahead solution from surpassing the myopic solutions. The alterations in Problem 2, namely the action space and the performance measure, exemplify long-term considerations where the performance of multi-step lookahead solutions exceeds that of myopic schemes. We verify this claim throughout several simulations. In fact, we have previously verified this claim for static and time-varying 1-sparse signals in noisy environments in [29] and [30].

This chapter is an extension of the previous work in which we concentrate on estimating time-varying s-sparse signals with more than one nonzero entry. Note that once we assume more than one nonzero moving entry in the signal, we must consider the problem of data association (see, e.g., [31]–[33]) that naturally arises in our problem. Therefore, we have to modify our POMDP formulation (Section 2.2) accordingly to take this problem into account.

It is well known that solving a POMDP problem is typically computationally prohibitive (see, e.g., [34] and [35]). The following approximation techniques decrease the computation volume, alleviating some of the conventional concerns:

1) We introduce two heuristics that are motivated from the well known techniques *joint probabilistic data association* (JPDA) and *multiple hypothesis tracking* (MHT) in the target tracking literature (see, e.g., [36]–[39]) to simplify the update of the belief state in the POMDP.

2) In the multi-step lookahead variation of our method, we use an approximation method, known as *rollout* (see [40]), to estimate an optimal solution for the POMDP.

2.2 POMDP Formulation

First, we formulate the problems introduced above as POMDPs. Since most of the POMDP elements for these two problems are similar, we present one POMDP formulation, but clarify the differences in the formulation of each problem where it is necessary.

States and Transition Law: The s-sparse vector $\mathbf{x}_k$ is fully characterized by its support (the locations of the targets) and the strength values (the values of the nonzero entries). Therefore, we take the state of the POMDP at time k to be $\mathbf{s}_k = (\mathbf{d}_k, \mathbf{v}_k)$, where $\mathbf{d}_k$ and $\mathbf{v}_k$ are $(s \times 1)$ vectors. The ith entry of these vectors, $\mathbf{d}_k(i)$ and $\mathbf{v}_k(i)$, represent the location and the strength value of the ith target at time k, respectively. Let the set Ω be defined as $\Omega = \{1, \ldots, N\}$. Also, let Ω_s be the set containing all the $(s \times 1)$ vectors $\mathbf{d}$ such that $\mathbf{d}(i) \in \Omega$ for $i = 1, \ldots, s$. Note that by using this definition for Ω_s, we are allowing any group of targets to be in the same location at the same time. Therefore, the cardinality of the set Ω_s is equal to $|\Omega_s| = N^s$. The POMDP state space is then defined as the Cartesian product $\mathcal{S} = (\Omega_s \times \mathbb{R}^s)$.

The state transition law of the POMDP, defined by the movement of the targets, is independent of our actions. Although the above definition for the POMDP state allows for the strength values of the targets to change over time, we make the simplifying assumption that these values stay constant over time. Accordingly, the state transition law of the POMDP can be described as:

$$P\{\mathbf{s}_{k+1} = (\mathbf{d}, \mathbf{v}) | \mathbf{s}_k = (\mathbf{e}, \mathbf{z})\} = \begin{cases} p_{\mathbf{ed}}, & \mathbf{v} = \mathbf{z}, \\ 0, & \text{otherwise}, \end{cases}$$

where $p_{\mathbf{ed}}$ is the probability that targets in locations represented by the vector $\mathbf{e}$ transition to locations represented by the vector $\mathbf{d}$. Depending on the movement of the targets, the probabilities $p_{\mathbf{ed}}$ could have different values. For example, if we consider the case where targets stay in the same locations at all time steps, then $p_{\mathbf{ed}} = 1$ if $\mathbf{e} = \mathbf{d}$, and $p_{\mathbf{ed}} = 0$ if otherwise.

Actions: In Problem 1, the action at each time step k is the selection of the measurement matrix $\mathbf{A}_k$. Therefore, the action space $\mathcal{A}$ is a prespecified subset of all matrices $\mathbf{A} \in \mathbb{R}^{l \times N}$ such that, for each row $\mathbf{A}(i), i = 1, \ldots, l$, $\|\mathbf{A}(i)\|_2 = 1$. In Problem 2, choosing the number of measurements l_k at each time step k is the POMDP action. In this case, the action space $\mathcal{A}$ is the set of natural numbers. For simplicity, we use the notation $\mathbf{u}_k$ for the POMDP action at time k for both problems.

Observations and Observation Law: The POMDP observations are the measurements $\mathbf{y}_k$ at each time step k. The observation law can be described in the following way: Given $\mathbf{s}_k = (\mathbf{d}, \mathbf{v})$

and the matrix $\mathbf{A}_k = \mathbf{A}$ at time step k, it can be shown that $\mathbf{y}_k|(\mathbf{s}_k = (\mathbf{d}, \mathbf{v}), \mathbf{A}_k = \mathbf{A}) \sim \mathcal{N}(\mathbf{A}_\mathbf{d}\mathbf{v}, \sigma_w^2\mathbf{I}_{l_k})$, where $\mathbf{A}_\mathbf{d}$ is a matrix whose ith column is the $\mathbf{d}(i)$th column of the matrix $\mathbf{A}$.

Cost: Let $I(\mathbf{y}_k; \mathbf{d}_k|H_k)$ be the conditional mutual information between the signal support $\mathbf{d}_k$ and the measurements $\mathbf{y}_k$ given the history $H_k = \{\mathbf{u}_1, \mathbf{y}_1, \ldots, \mathbf{u}_k, \mathbf{y}_k\}$. The per-time-step cost $c_k(\mathbf{s}_k, \mathbf{u}_k)$ for Problem 1 is then simply defined as $c_k(\mathbf{s}_k, \mathbf{u}_k) = -I(\mathbf{y}_k; \mathbf{d}_k|H_k)$. For Problem 2, the cost is a combination of the number of measurements l_k and the conditional mutual information $I(\mathbf{y}_k; \mathbf{d}_k|H_k)$ at each time step k. More specifically, the per-time-step cost is defined as

$$c_k(\mathbf{s}_k, \mathbf{u}_k) = l_k - \gamma I(\mathbf{y}_k; \mathbf{d}_k|H_k), \tag{2.1}$$

where $\gamma \geq 0$ is a weighting parameter chosen based on the priority of either l_k or $I(\mathbf{y}_k; \mathbf{d}_k|H_k)$. For example, if we are not particularly concerned with the number of measurements, then we choose a large value for γ. On the contrary, if we have to be careful with the total number of measurements used in m steps, we use a small value for γ.

Belief State: The belief state $\mathbf{b}_k$ at time k is defined as the posterior (conditional) distribution of the state $\mathbf{s}_k$ given the history H_{k-1}. Using the belief state definition, our POMDP can be presented as an equivalent Markov Decision Process (MDP) (see [41]). This MDP, which is also referred to as the *belief MDP*, has the following components:

1. A state space consisting of all the possible belief states $\mathbf{b}_k$ for the POMDP.

2. A state transition law that is computed using the state transition law and the observation law of the POMDP. This computation is referred to as the *belief state update*, shown in detail in the next section.

3. An action space which is the same as that of the POMDP.

4. A cost $C_k(\mathbf{b}_k, \mathbf{u}_k)$, referred to as the *belief cost*, which is defined as:

$$C_k(\mathbf{b}_k, \mathbf{u}_k) = \mathbf{E}\left[c_k(\mathbf{s}_k, \mathbf{u}_k)|H_k, \mathbf{b}_k\right], \tag{2.2}$$

where the expectation is with respect to the posterior distribution of the state $\mathbf{s}_k$ given the history H_k at time step k, i.e., the belief state $\mathbf{b}_k$.

Using the belief MDP model, we can now apply the approximation methods available in the literature, specifically rollout [42].

2.2.1 Belief State Update

Let $P_{\mathbf{d}_k|H_k}$ be the conditional probability mass function of $\mathbf{d}_k$ given H_k, and $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ be the conditional density function of $\mathbf{v}_k$ given $\mathbf{d}_k$ and H_k at time step k. Consider a simple example, shown in Figure 2.1, where there are three *nodes*, $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$, representing the possible signal supports at any time step. Figure 2.1 also displays the *links* between nodes, representing allowed support transitions. For example, there are two possible transitions arriving at $\mathbf{e}_1$, namely, those originating from nodes $\mathbf{e}_1$ and $\mathbf{e}_2$. Attached to the initial node of each link is a weight value, computed from a prior conditional probability mass function, and also a prior conditional density function. Upon choosing an action and receiving measurements, we can compute a posterior weight value as well as a posterior density function for the terminal node of the link.

To clarify, consider the selected *route*, comprised of successive links and outlined in red in Figure 2.1, that begins at node $\mathbf{e}_3$ at time $k = 1$ and after passing through node $\mathbf{e}_2$ at time $k = 2$, it finally arrives at node $\mathbf{e}_1$ at $k = 3$. At time $k = 1$, once measurement $\mathbf{y}_1$ is received, we can compute the posterior weight value $P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1)$ and the posterior density function $f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3, H_1)$ using the initial prior functions $P_{\mathbf{d}_0}$ and $f_{\mathbf{v}_0|\mathbf{d}_0}$ (this calculation is shown later in this section). Then, examining the link from $\mathbf{e}_3$ to $\mathbf{e}_2$, the prior weight and density function attached to node $\mathbf{e}_3$ for the next time step are $P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1)$ and $f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3, H_1)$, respectively. The prior values are updated using the received observations, resulting in the posterior weight and density function $P_{\mathbf{d}_1|\mathbf{d}_0,H_1}(\mathbf{e}_2|\mathbf{e}_3, H_1)$ and $f_{\mathbf{v}_1|\mathbf{d}_1,\mathbf{d}_0,H_1}(\cdot|\mathbf{e}_2, \mathbf{e}_3, H_1)$, respectively. For the next link in the route from $\mathbf{e}_2$ to $\mathbf{e}_1$, the prior weight and density function used for $\mathbf{e}_2$ will be the posterior weight and density function of this node at $k = 2$, i.e., $P_{\mathbf{d}_1|\mathbf{d}_0,H_1}(\mathbf{e}_2|\mathbf{e}_3, H_1)$ and $f_{\mathbf{v}_1|\mathbf{d}_1,\mathbf{d}_0,H_1}(\cdot|\mathbf{e}_2, \mathbf{e}_3, H_1)$, respectively. Again, after choosing an action and receiving measurements, the posterior weight

$P_{\mathbf{d}_2|\mathbf{d}_1,H_2}(\mathbf{e}_1|\mathbf{e}_2,H_2)$ and posterior density function $f_{\mathbf{v}_2|\mathbf{d}_2,\mathbf{d}_1,H_2}(\cdot|\mathbf{e}_1,\mathbf{e}_2,H_2)$ for node $\mathbf{e}_1$ are computed.

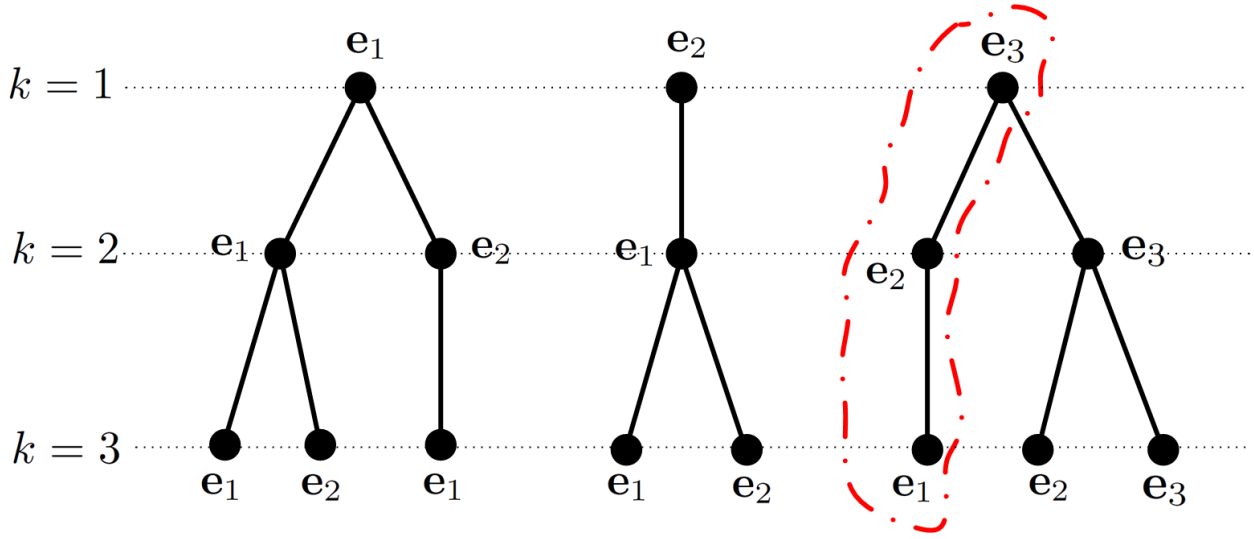


Figure 2.1: A simple example showing the construction of various links and routes from only three support possibilities $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ over time.

It is important to realize that after the first time step, the posterior weight and density function for the ending node of any link are always conditioned on the initial node of that link. Thus, knowing the positions of the targets and their prior weight and density at the previous time step is required. At any time step k, given each possible link, we define an ordered tuple $\boldsymbol{\tau}$ consisting of four elements: the terminal node, the initial node, the prior weight value, and the prior density function attached to the initial node and let $\boldsymbol{\tau}(i)$ be the ith element of the tuple. For the same route used in the above example, the tuple defined for the link from $\mathbf{e}_3$ to $\mathbf{e}_2$ is $\boldsymbol{\tau} = \big(\mathbf{e}_2, \mathbf{e}_3, P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1), f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3,H_1)\big)$. Analogously, the tuple for the last link in this route, from $\mathbf{e}_2$ to $\mathbf{e}_1$, is $\boldsymbol{\tau} = \big(\mathbf{e}_1, \mathbf{e}_2, P_{\mathbf{d}_2|\mathbf{d}_1,H_2}(\mathbf{e}_1|\mathbf{e}_2,H_2), f_{\mathbf{v}_2|\mathbf{d}_2,\mathbf{d}_1,H_2}(\cdot|\mathbf{e}_1,\mathbf{e}_2,H_2)\big)$.

At any time step, multiple tuples with the same initial and terminal nodes may exist. For example, in Figure 2.1, from time $k=2$ to time $k=3$, there are two links starting at $\mathbf{e}_2$ and ending at $\mathbf{e}_1$. These two links can be distinguished by the unique priors attached to the starting nodes of either link. Let $\mathcal{T}_k$ be the set containing all the possible tuples at time step k. Also, define

$\mathcal{T}_{k,\mathbf{d}}$ to be $\mathcal{T}_{k,\mathbf{d}} = \{\boldsymbol{\tau} : \boldsymbol{\tau} \in \mathcal{T}_k, \boldsymbol{\tau}(1) = \mathbf{d}\}$, the set of all tuples representing links ending with support $\mathbf{d}$ at time k. At each time step k, the number of possible links, which depends on the number of initial nodes and the state transition law, determines the cardinality $|\mathcal{T}_k|$ of the set $\mathcal{T}_k$, which increases exponentially as time evolves. We will discuss this issue later.

Recall that the state of the POMDP at time step k is $\mathbf{s}_k = (\mathbf{d}_k, \mathbf{v}_k)$. The belief state $\mathbf{b}_k$ is the posterior joint distribution of $\mathbf{d}_k$ and $\mathbf{v}_k$ given H_k, or equivalently, the functions $P_{\mathbf{d}_k|H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$. Therefore, updating the belief state $\mathbf{b}_k$ is equivalent to updating these functions. Moreover, given $\mathbf{y}_k = \mathbf{y}$ and $\mathbf{u}_k = \mathbf{u}$, the functions $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ and $P_{\mathbf{d}_k|H_k}$ can be computed from $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ using Bayes' rule and the law of total probability in the following way:

$$f_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\mathbf{v}|\mathbf{d},H_k) = \frac{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} g(\mathbf{v},\mathbf{d},\boldsymbol{\tau}) P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2),H_k)\boldsymbol{\tau}(3)}{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2),H_k)\boldsymbol{\tau}(3)}, \tag{2.3}$$

and,

$$P_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k) = \frac{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2),H_k)\boldsymbol{\tau}(3)}{\sum_{\mathbf{e}\in\Omega_s}\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{e}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{e}|\boldsymbol{\tau}(2),H_k)\boldsymbol{\tau}(3)}, \tag{2.4}$$

where $g(\mathbf{v},\mathbf{d},\boldsymbol{\tau}) = f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d},\boldsymbol{\tau}(2),H_k)$. Note that the value $\boldsymbol{\tau}(3)$ is the prior weight value of the starting node $\boldsymbol{\tau}(2)$ for the link represented by the tuple $\boldsymbol{\tau}$. This means that to update the belief state $\mathbf{b}_k$ at time step k, it is sufficient to update functions $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ for all the possible links that the state transition law allows at each time step.

In this chapter, we assume a dynamic linear model and a initial Gaussian distribution for $\mathbf{v}_0|\mathbf{d}_0$ given each $\mathbf{d}_0 \in \Omega_s$. Each distribution can be characterized by a density function $f_{\mathbf{v}_0|\mathbf{d}_0}$ with the mean vector $\boldsymbol{\mu}_{\mathbf{d_0}}$ and the covariance matrix $\mathbf{C}_{\mathbf{d}_0}$. Consequently, the density function $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$, $k \geq 1$, will also be Gaussian, and we use $\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ for the mean vector and $\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ for the covariance matrix.

Given the action $\mathbf{u}_k = \mathbf{u}$, the measurements $\mathbf{y}_k = \mathbf{y}$, and the history H_{k-1}, the functions $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ can be computed using the following two steps:

1. For a given any link, represented by a tuple $\boldsymbol{\tau}$ in $\mathcal{T}_k$, the following system of linear equations describes the dynamics of the moving targets along that link:

$$\begin{cases} \mathbf{v}_k = \mathbf{v}_{k-1}, \\ \mathbf{y}_k = \mathbf{A}_{\boldsymbol{\tau}(1)}\mathbf{v}_k + \mathbf{w}_k. \end{cases}$$

In this system, $\mathbf{v}_k \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}, \mathbf{C}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)})$. The vector $\mathbf{v}_{k-1}$ also has a Gaussian distribution, which is represented by the density function $\boldsymbol{\tau}(4)$ in the tuple $\boldsymbol{\tau}$. Therefore, the values $\boldsymbol{\mu}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}$ and $\mathbf{C}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}$ for the given support pair $\mathbf{d}_k = \boldsymbol{\tau}(1)$ and $\mathbf{d}_{k-1} = \boldsymbol{\tau}(2)$ can be computed by using a Kalman filter. Note that for each possible tuple $\boldsymbol{\tau} \in \mathcal{T}_k$, we have to use a separate Kalman filter to update the corresponding prior distribution function $\boldsymbol{\tau}(4)$ of the related link. In the next section, we will discuss the computational issues that usually arise. Once this update is completed, the value of the function $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\boldsymbol{\tau}(1),\boldsymbol{\tau}(2),H_k)$ for any $\mathbf{v} \in \mathbb{R}^s$ can then be computed.

2. For a given tuple $\boldsymbol{\tau} \in \mathcal{T}_k$, the posterior weight value is computed in the following way:

$$P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2),H_k) = \frac{f_1(\mathbf{y},\mathbf{u},\boldsymbol{\tau},\boldsymbol{\tau})f_2(\boldsymbol{\tau},\boldsymbol{\tau})}{\sum_{\boldsymbol{\nu}\in\mathcal{T}_k} f_1(\mathbf{y},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\tau})f_2(\boldsymbol{\nu},\boldsymbol{\tau})}, \tag{2.5}$$

where functions $f_1(\mathbf{y},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\tau})$ and $f_2(\boldsymbol{\nu},\boldsymbol{\tau})$ are defined as:

$$\begin{aligned} f_1(\mathbf{y},\mathbf{u},\boldsymbol{\nu},\boldsymbol{\tau}) &= f_{\mathbf{y}_k|\mathbf{d}_k,\mathbf{d}_{k-1},\mathbf{u}_k,H_{k-1}}(\mathbf{y}|\boldsymbol{\nu}(1),\boldsymbol{\tau}(2),\mathbf{u},H_{k-1}) \\ f_2(\boldsymbol{\nu},\boldsymbol{\tau}) &= P_{\mathbf{d}_k|\mathbf{d}_{k-1}}(\boldsymbol{\nu}(1)|\boldsymbol{\tau}(2))\boldsymbol{\tau}(3) = p_{\boldsymbol{\tau}(2)\boldsymbol{\nu}(1)}\boldsymbol{\tau}(3). \end{aligned}$$

Note that the value $p_{\boldsymbol{\tau}(2)\boldsymbol{\nu}(1)}$, that can be computed from the state transition law, is equal to zero for the non-existing links from $\boldsymbol{\tau}(2)$ to $\boldsymbol{\nu}(1)$. Moreover, for the function f_1, it can be easily shown that

$$\mathbf{y}_k|\mathbf{d}_k,\mathbf{d}_{k-1},\mathbf{u}_k,H_{k-1} \sim \mathcal{N}(\mathbf{A}_{\mathbf{d}_k}\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}, \mathbf{A}_{\mathbf{d}_k}\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}\mathbf{A}'_{\mathbf{d}_k} + \sigma_w^2\mathbf{I}_{l_k}),$$

where $\mathbf{A}'_{\mathbf{d}_k}$ is the transpose of the matrix $\mathbf{A}_{\mathbf{d}_k}$. Note that the values $\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ and $\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ have already been computed in the previous step.

As mentioned before, after going through the above two steps for all the tuples in $\mathcal{T}_k$ and also computing the functions $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ and $P_{\mathbf{d}_k|H_k}$ given any $\mathbf{d}_k \in \Omega_s$ using (2.3) and (2.4), the update of the belief state $\mathbf{b}_k$ is completed.

2.2.2 Computational Issues in the Belief State Update

The belief state update procedure described in the previous section shows the computations required once measurements are received at each time step. To highlight the computational issues, consider the extreme case where the transition law $P_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ allows the targets to move freely to any position. Also, assume a uniform initial support density function $P_{\mathbf{d}_0}$ over Ω_s. Knowing that $|\Omega_s| = N^s$, then for each support candidate $\mathbf{d}_0$ in Ω_s, there would be N^s possibilities for the support $\mathbf{d}_1$. Therefore, N^{2s} Kalman filters must be employed to update the belief state at time $k = 1$. Extending this line of thought, at any time k, $N^{(k+1)s}$ Kalman filters are needed which is an exponential increase in computations over time.

Of course in reality, the assumed transition law $P_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ does not provide such freedom for the moving target(s). Although this reduces the number of possible transitions, the exponential increase in computation remains. This is seen in the following example. Again assume, at time $k = 1$, a uniform prior distribution $P_{\mathbf{d}_0}$ over Ω_s. However, now assume the transition law restricts the number of possible transitions for each support candidate to $q \leq N^s$. Following the same argument as above, there are now $N^s q$ links involved in the belief state update at time $k = 1$ and for time $k = 2$, $N^s q^2$ links are present. Similarly, $N^s q^k$ links (equivalently Kalman filters) are required for the belief state update at any time step k.

Obviously, updating the belief state according to the above procedure requires a vast amount of time and also computational power. To deal with the computational volume we introduce two techniques explained below. This allows us to examine the proposed solution methods to Problems 1 and 2 and avoid some of the related computational load.

As we have shown, propagating the entire distribution quickly becomes unmanageable due to its exponential growth-rate in terms of the number of possible links. This issue is also a major problem in the context of data association in the multi-target tracking problem. In this problem, the main question asked is: At each time step, which target does of the collected measurements at that time step represent? If all the observations-to-target associations were considered valid, a similar expansion of the distribution representing the current target states would occur.

We can consider the procedure in Section 2.2.1 as one of associating the measurements $\mathbf{y}_k$ to the *possible* support-value pairs, which are characterized by their corresponding belief state components $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$. The similarity to the data association problem motivated the adaptation of available solutions in the target tracking context. We have implemented two separate methods based on popular data association algorithms to combat the expansion of our belief state distributions, namely: multi-hypothesis tracking (MHT) (see [38]) and joint probabilistic data association (JPDA) (see [36] and [37]).

Algorithm 1: The essential question to ask is how to choose or approximate the proper Gaussian distributions and weights to represent different possible support candidates along different possible routes. There are several approaches to accomplish this, e.g., MMSE, MLE, etc., and in the target tracking literature, there are also numerous implementations applying these approximations. We model our first method after MHT.

The basis of MHT lies in multi-hypothesis testing wherein a choice must be made between several concurrent hypotheses based on a sufficient statistic. Given a set of J hypotheses $\mathcal{H} = \{h_i\}_{i=1}^{J}$ and the measurements $\mathbf{y}_k = \mathbf{y}$, the simplest version of multi-hypothesis testing chooses one hypothesis h^* based on finding an optimal solution for the following optimization problem:

$$h^* = \arg\max_{1\leq i\leq J} \left\{ \mathcal{L}\left(h_i|\mathbf{y}\right) P(h_i) \right\},$$

where $\mathcal{L}\left(h_i|\mathbf{y}\right)$ and $P(h_i)$ denote the likelihood given the measurement $\mathbf{y}$ and the prior probability of the hypothesis h_i, respectively. The function $\mathcal{L}\left(h_i|\mathbf{y}\right)$ is equivalent to $f_{\mathbf{y}_k|h}(\mathbf{y}|h_i)$, the conditional probability density function of the measurement vector $\mathbf{y}_k$ conditioned on the hypothesis h.

Therefore, the above optimization problem will be equal to

$$h^* = \arg \max_{1 \leq i \leq J} \{ f_{\mathbf{y}_k, h}(\mathbf{y}, h_i) \} .$$

In our setting, similar to the target tracking setting, more measurements accrue as time progresses, providing more information about the underlying state. Obviously, more information can help increase the accuracy of the association process. Even in dynamic systems, observations of future states provide valuable information about the past system states. This is the basic motivation behind the MHT method.

Consider a time-varying signal estimation problem with a finite horizon of length κ. At each time step, a measurement is acquired through the observation law. Then, the most accurate selection of a hypothesis is made by collecting all of the measurements and maintaining each possible association over the entire horizon. This results in a hypothesis tree, κ levels deep. Each level consists of nodes representing the possible support-value pairs (supports may be repeated) and the corresponding weights $P_{\mathbf{d}_k|H_k}$ for each time step k. At the final time step $k = \kappa$, applying a multi-hypothesis test to the set of leaf nodes selects the leaf terminating the most probable route in the tree (most probable with respect the posterior distribution). This selection not only yields a current estimate of the support and values that belong to the leaf at the end of the route, but this route, itself, describes the most probable sequence of support-value pairs. Note, in our setting, the value component $\mathbf{v}_k$ of the state has a static transition law, and thus the final estimate of $\mathbf{v}_\kappa$ would be the more accurate that the estimates of $\mathbf{v}_k$, $k < \kappa$ because they would not have benefited from the later observations.

Let $\boldsymbol{\rho}_k^{(i)}$ be an ordered k-tuple representing the ith possible route at time k. The first component $\boldsymbol{\rho}_k^{(i)}(1)$ of this tuple is the current node of this route and the last component $\boldsymbol{\rho}_k^{(i)}(k)$ is the initial node of this route. For example, for the selected route in Figure 2.1, which we refer to as the ith route, we have the following tuples for time steps $k = 1, 2$, and $k = 3$, respectively: $\boldsymbol{\rho}_1^{(i)} = \mathbf{e}_3$, $\boldsymbol{\rho}_2^{(i)} = (\mathbf{e}_2, \mathbf{e}_3)$, and $\boldsymbol{\rho}_3^{(i)} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$. Note that at time step k, the total number of possible routes is equal to $|\mathcal{T}_k|$. Let $h_i^{(k)}$ be the hypothesis that at time step k, the ith route is the true route.

If this hypothesis is true, then this means that $\mathbf{d}_k = \boldsymbol{\rho}_k^{(i)}(1), \mathbf{d}_{k-1} = \boldsymbol{\rho}_k^{(i)}(2), \ldots, \mathbf{d}_1 = \boldsymbol{\rho}_k^{(i)}(k)$. Additionally, let $h^{(k)}$ be the random variable representing the true hypothesis; $h^{(k)}$ takes values on set of all hypotheses $h_i^{(k)}$, $i = 1, \ldots, |\mathcal{T}_k|$, at time step k.

Determining the terminal node of the probable route is done by ranking the possible routes with their *track scores*. Let $\boldsymbol{o}_k$ be a tuple defined as $\boldsymbol{o}_k = (\mathbf{y}_1, \ldots, \mathbf{y}_k)$. Then, given the tuple $\boldsymbol{o}_\kappa = \boldsymbol{o}$, where $\boldsymbol{o} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(\kappa)})$, and the history $H_\kappa = \{\mathbf{u}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{u}^{(\kappa)}, \mathbf{y}^{(\kappa)}\}$ at time step κ, the track score $S_{\boldsymbol{\rho}_\kappa^{(i)}}$ of the route i, represented by the tuple $\boldsymbol{\rho}_\kappa^{(i)}$, is defined as

$$S_{\boldsymbol{\rho}_\kappa^{(i)}} = \mathcal{L}(h_i^{(\kappa)}|\boldsymbol{o})P(h_i^{(\kappa)}) = f_{\boldsymbol{o}_\kappa, h^{(\kappa)}}(\boldsymbol{o}, h_i^{(\kappa)}),$$

where $f_{\boldsymbol{o}_\kappa, h^{(\kappa)}}$ is the joint probability density function of $\boldsymbol{o}_\kappa$ and $h^{(\kappa)}$ at time step κ. By expanding the function $f_{\boldsymbol{o}_\kappa, h^{(\kappa)}}$, the value of $S_{\boldsymbol{\rho}_\kappa^{(i)}}$ is equal to

$$\begin{aligned} &f_{\mathbf{y}_\kappa|\mathbf{d}_\kappa, \mathbf{d}_{\kappa-1}, \mathbf{u}_\kappa, H_{\kappa-1}}(\boldsymbol{o}(\kappa)|\boldsymbol{\rho}_\kappa^{(i)}(1), \boldsymbol{\rho}_\kappa^{(i)}(2), \mathbf{u}^{(\kappa)}, H_{\kappa-1}) \\ &\quad \times p_{\boldsymbol{\rho}_\kappa^{(i)}(2)\boldsymbol{\rho}_\kappa^{(i)}(1)} P_{\mathbf{d}_{\kappa-1}|\mathbf{d}_{\kappa-2}, H_{\kappa-1}}(\boldsymbol{\rho}_\kappa^{(i)}(2)|\boldsymbol{\rho}_\kappa^{(i)}(3), H_{\kappa-1}). \end{aligned}$$

Note that all of the components in this expression can be computed using the equations presented in Section 2.2.1. Looking at (2.5), one can conclude that $S_{\boldsymbol{\rho}_\kappa^{(i)}}$ is proportional to $S^{\text{eq.}}_{\boldsymbol{\rho}_\kappa^{(i)}}$ where

$$S^{\text{eq.}}_{\boldsymbol{\rho}_\kappa^{(i)}} = \prod_{k=1}^{\kappa} t(\boldsymbol{o}, \boldsymbol{\rho}_k^{(i)}, H_k, k),$$

and the function t is defined as

$$t(\boldsymbol{o}, \boldsymbol{\rho}_k^{(i)}, H_k, k) = \begin{cases} f_{\mathbf{y}_k|\mathbf{d}_k, \mathbf{d}_{k-1}, \mathbf{u}_k, H_{k-1}}(\boldsymbol{o}(k)|\boldsymbol{\rho}_k^{(i)}(1), \boldsymbol{\rho}_k^{(i)}(2), \mathbf{u}^{(k)}, H_{k-1}) \times p_{\boldsymbol{\rho}_k^{(i)}(2)\boldsymbol{\rho}_k^{(i)}(1)}, & k \geq 2 \\ P_{\mathbf{d}_1|H_1}(\boldsymbol{\rho}_1^{(i)}(1)|H_1), & k = 1 \end{cases}.$$

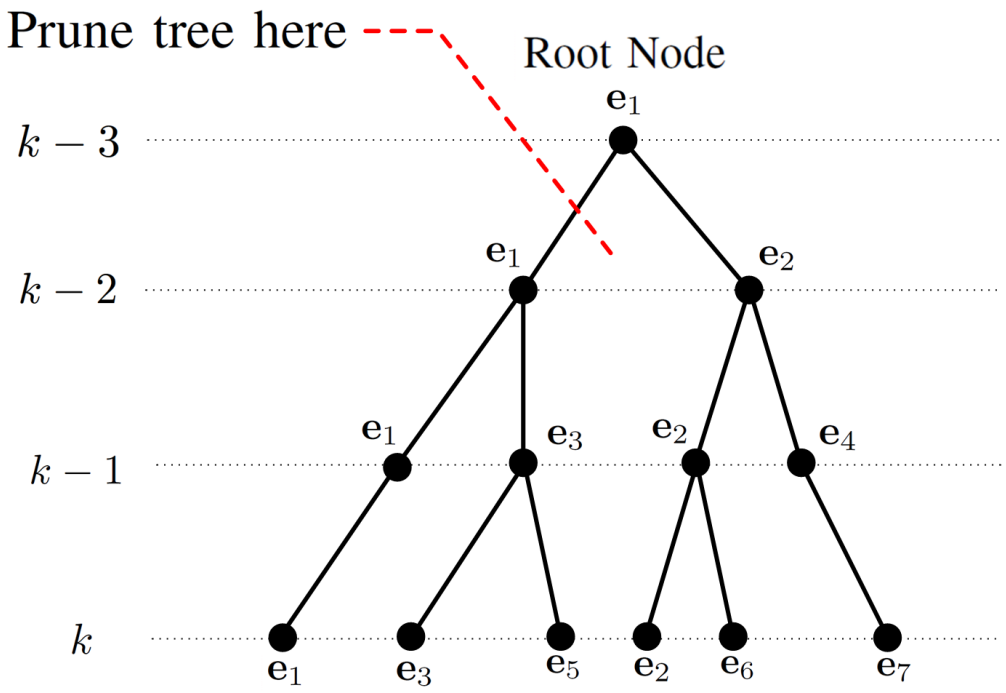


Figure 2.2: This illustration (adapted from [38]) represents a possible hypothesis tree for a time-varying signal scenario. Identically labeled nodes indicate that the support has remained the same over the transition. A sliding window of size $w = 2$ is depicted and the pruning action has indicated that the maximum track score belongs to the route represented by the tuple $(\mathbf{e}_6, \mathbf{e}_2, \mathbf{e}_2, \mathbf{e}_1)$. Thus, the pruning is done as shown. This has reduced the computation for time step $k+1$ by 50%.

Therefore, we can use $S^{\text{eq.}}_{\boldsymbol{\rho}^{(i)}_\kappa}$ equivalently to find most probable route at time step κ. Practical concerns such as numerical round-off errors motivate a conversion of $S^{\text{eq.}}_{\boldsymbol{\rho}^{(i)}_\kappa}$ to a logarithmic representation, $S^{\log}_{\boldsymbol{\rho}^{(i)}_\kappa}$, in the following way

$$S^{\log}_{\boldsymbol{\rho}^{(i)}_\kappa} = \sum_{k=1}^{\kappa} \log\left(t(\boldsymbol{o}, \boldsymbol{\rho}^{(i)}_k, H_k, k)\right).$$

This alternative representation also provides a simple recursive calculation,

$$S^{\log}_{\boldsymbol{\rho}^{(i)}_k} = S^{\log}_{\boldsymbol{\rho}^{(i)}_{k-1}} + \log\left(t(\boldsymbol{o}, \boldsymbol{\rho}^{(i)}_k, H_k, k)\right), \qquad k = 2, \ldots, \kappa.$$

As mentioned earlier, when no approximation techniques are used, the number of possible routes increases exponentially as time evolves. To deal with this problem, most MHT implementations use a sliding window to designate when the decision making process begins. The sliding

window heuristic limits the size of the hypothesis tree that must be maintained. Consider a sliding window of size w. After the first w time steps, a hypothesis test is performed, selecting the current leaf with the maximum track score. The route terminating with the selected leaf is then traced back w levels in the tree. The node at the base of this branch and all of its descendants are retained, while all other nodes and branches from time-step $(k-w)$ onward are discarded. At every time step $k \geq w$, a similar pruning of the tree is performed. The larger the sliding window the better the chances of a correct association due to the increased information accumulated. Figure 2.2 illustrates this process for an arbitrary time step.

The sliding window size must be balanced with the computational time and memory constraints. The maximum reduction in computation from our MHT based algorithm would be a sliding window of size one. Thus, a hypothesis test would be performed at every time step. One data-association technique that uses a sliding window of size one is JPDA (see [36] and [37]). This is the basis for Algorithm 2, described next.

Algorithm 2: Consider the set $\mathcal{T}_k$ defined in Section 2.2.1. Looking at the tuples in this set, each of which is associated with a possible link, we can find multiple tuples that all share a similar ending node. For example, in Figure 2.1, there are 4 instances for node $\mathbf{e}_1$ at time step $k = 3$. To cope with the computational volume we combine the posterior weights and distributions attached to different instances of an ending node. More specifically, at each time step k, instead of keeping all instances of a support candidate $\mathbf{d} \in \Omega_s$, we represent the candidate with only one *approximate* posterior weight value $\hat{P}_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k)$ and one *approximate* posterior distribution function $\hat{f}_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\cdot|\mathbf{d},H_k)$. This will force each set $\mathcal{T}_k$ for $k = 1, 2, \ldots$, to have a fixed cardinality equal to the cardinality of the set Ω_s, i.e., $|\mathcal{T}_k| = N^s$.

To pursue this, consider (2.3). Define $\alpha_{\boldsymbol{\tau},\mathbf{d},H_k}$ to be

$$\alpha_{\boldsymbol{\tau},\mathbf{d},H_k} = \frac{P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2),H_k)\boldsymbol{\tau}(3)}{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2),H_k)\boldsymbol{\tau}(3)}.$$

Then, equation (2.3) simplifies to

$$f_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\mathbf{v}|\mathbf{d},H_k) = \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d},\boldsymbol{\tau}(2),H_k).$$

This is the representation of a Gaussian mixture with weight values $\alpha_{\boldsymbol{\tau},\mathbf{d},H_k}$ and Gaussian components $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$. Given $\mathbf{d}_k = \mathbf{d}$ at time step k, the mean $\boldsymbol{\mu}_{\mathbf{d}}$ of this mixture will be equal to

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{d}} &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \int_{\mathbf{v}} \mathbf{v} f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d},\boldsymbol{\tau}(2),H_k)d\mathbf{v} \\ &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \boldsymbol{\mu}_{\mathbf{d}|\boldsymbol{\tau}(2)}.\end{aligned}$$

Also, the covariance matrix $\mathbf{C}_{\mathbf{d}}$ of this mixture can be computed in the following way:

$$\begin{aligned}\mathbf{C}_{\mathbf{d}} &= \mathbf{E}\left[(\mathbf{v}_k|\mathbf{d},H_k)(\mathbf{v}_k|\mathbf{d},H_k)'\right] - \boldsymbol{\mu}_{\mathbf{d}}\boldsymbol{\mu}_{\mathbf{d}}^{'} \\ &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \times \int_{\mathbf{v}} \mathbf{v}\mathbf{v}' f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d},\boldsymbol{\tau}(2),H_k)d\mathbf{v} - \boldsymbol{\mu}_{\mathbf{d}}\boldsymbol{\mu}_{\mathbf{d}}^{'} \\ &= \sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} \alpha_{\boldsymbol{\tau},\mathbf{d},H_k} \left(\mathbf{C}_{\mathbf{d}|\boldsymbol{\tau}(2)} + \boldsymbol{\mu}_{\mathbf{d}|\boldsymbol{\tau}(2)}\boldsymbol{\mu}_{\mathbf{d}|\boldsymbol{\tau}(2)}^{'}\right) - \boldsymbol{\mu}_{\mathbf{d}}\boldsymbol{\mu}_{\mathbf{d}}^{'}.\end{aligned}$$

Now, we approximate the Gaussian mixture density function with a regular Gaussian density function $\hat{f}_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\cdot|\mathbf{d},H_k)$ that has the mean $\boldsymbol{\mu}_{\mathbf{d}}$ and the covariance matrix $\mathbf{C}_{\mathbf{d}}$. We apply this approximation to all the support candidates in Ω_s.

To further decrease the computation volume, we first compute $P_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k)$ for all possible support values $\mathbf{d} \in \Omega_s$ using (2.4). We then prune this probability function by keeping a certain number h of the weights with the highest probability values in $P_{\mathbf{d}_k|H_k}$ and forcing the rest to be zero. Adjusting the nonzero values to sum to one will result in having an approximate probability function $\hat{P}_{\mathbf{d}_k|H_k}$ to use as the prior at time step $k+1$.

This means that at each time step, we will only have h initial nodes instead of N^s nodes. Therefore, at time step k, instead of using $N^s q^k$ Kalman filters for the scenario explained above,

only hq Kalman filters are required to update the belief state. This will decrease the amount of computation significantly.

As mentioned earlier, the ideas used in Algorithm 2 are motivated by a target tracking data association heuristic algorithm known as JPDA. We refer the interested reader to [36] and [37] to learn more about this heuristic.

2.3 Rollout

In both problems introduced in Section 2.1, we want to make optimal decisions at each time step (based on appropriate criteria) by either selecting the measurement matrix $\mathbf{A}_k$ or choosing the number of measurements l_k, given the history H_k. In Section 2.2, we formulated these problems as POMDPs. In this section, we describe our solution approach based on an approximation method called *rollout*. Our description relies heavily on well established ideas and terminology from POMDP theory, which are readily available in [27] and [28].

In principle, the solution to a POMDP problem is, at each time k, a mapping that takes the history H_k and gives an optimal action from $\mathcal{A}$. However, POMDP theory establishes that is suffices to find an optimal mapping $\pi_k^* : \mathcal{B} \rightarrow \mathcal{A}$ for $k = 1, \ldots, m$, where $\mathcal{B}$ is the set of distributions over the state space $\mathcal{S}$, and $\mathcal{A}$ is the actions space. If all actions are chosen using these optimal mappings, then at time $k = m$, the predefined objective function is optimized and the resulting *optimal policy* $\pi^* = \{\pi_1^*, \ldots, \pi_m^*\}$ has been generated.

We refer to the objective function as the *expected cumulative cost*. Given a policy $\pi = \{\pi_1, \ldots, \pi_m\}$ the expected cumulative cost is defined as

$$V_m^{\pi}(\mathbf{b}_1) = \mathbf{E}\left[\sum_{k=1}^{m} C_k(\mathbf{b}_k, \pi_k(\mathbf{b}_k)) \middle| \mathbf{b}_1\right]. \tag{2.6}$$

Subsequently, the optimal objective function value $V_m^{\pi^*}$ is achieved when $\pi = \pi^*$ in (2.6).

By applying *Bellman's principle* to (2.6), the optimal objective function $V_m^{\pi^*}$ and the optimal policy π^* can be characterized by the following two equations:

$$V_m^{\pi^*}(\mathbf{b}_1) = \min_{\mathbf{u}}(C_1(\mathbf{b}_1, \mathbf{u}) + \mathbf{E}[V_{m-1}^{\pi^*}(\mathbf{b}_2)|\mathbf{b}_1, \mathbf{u}]), \tag{2.7}$$

and

$$\pi_1^*(\mathbf{b}_1) = \arg\min_{\mathbf{u}}(C_1(\mathbf{b}_1, \mathbf{u}) + \mathbf{E}[V_{m-1}^{\pi^*}(\mathbf{b}_2)|\mathbf{b}_1, \mathbf{u}]). \tag{2.8}$$

Let

$$Q_{m-k}(\mathbf{b}_k, \mathbf{u}) = C_k(\mathbf{b}_k, \mathbf{u}) + \mathbf{E}[V_{m-k}^{\pi^*}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}] \tag{2.9}$$

be the Q-*value* of taking action $\mathbf{u}$ at belief state $\mathbf{b}_k$. Combining (2.8) and (2.9), we can now view the optimal action at time k as

$$\pi_k^*(\mathbf{b}_k) = \arg\min_{\mathbf{u}} Q_{m-k}(\mathbf{b}_k, \mathbf{u}).$$

In other words, the optimal action can be found at any time step k by identifying the action with the minimum Q-value at belief state $\mathbf{b}_k$.

The two summands in (2.9), $C_k(\mathbf{b}_k, \mathbf{u})$ and $\mathbf{E}[V_{m-k}^{\pi^*}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}]$, are the immediate cost incurred (by $\mathbf{u}$) and the *expected cost-to-go* (ECTG) at the belief state $\mathbf{b}_k$, respectively. This breakdown inspires a natural separation in action selection approaches, namely *myopic* and *non-myopic*. Myopic action selection only considers the immediate impact (cost) of the action $\mathbf{u}$ at belief state $\mathbf{b}_k$. These approximations yield a *myopic* policy $\hat{\pi}$ which selects actions by

$$\hat{\pi}_k(\mathbf{b}_k) = \arg\min_{\mathbf{u}} C_k(\mathbf{b}_k, \mathbf{u}).$$

In relation to the Q-values, the myopic policy can be viewed as replacing the ECTG with a fixed constant, which can be assumed to be zero without loss of generality. In contrast, non-myopic methods attempt to approximate the ECTG term of the Q-value. If the estimation of the ECTG is suitable, it will evince the impact of the current action on the future costs to be incurred. There are several approaches for Q-value approximation [40]. We choose to implement the *rollout* method.

Rollout replaces the optimal policy used in the ECTG with a so called *base policy* π^{base}, creating the following Q-value approximation:

$$\tilde{Q}_{m-k}(\mathbf{b}_k, \mathbf{u}) = C_k(\mathbf{b}_k, \mathbf{u}) + \mathbf{E}[V_{m-k}^{\pi^{\text{base}}}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}].$$

This approximation differentiates the effects of the actions not only at the current time step but over the future horizon, providing *multi-step lookahead* as opposed to the *1-step lookahead* of the myopic policy. It should also be noted that the simulation results presented in the next section produced by rollout exploit *receding horizon control*, replacing the remaining horizon $m - k$ with a fixed value ϑ. This "artificial" horizon is regarded as the *rollout horizon*; rollout defines a ϑ-lookahead policy.

The technical details justifying both receding horizon control and rollout can be found in [40], but we provide the reader with a meaningful result which motivates the use of rollout. By ranking actions over time with their approximate Q-values, rollout produces a policy $\tilde{\pi} = \{\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_m\}$, where

$$\tilde{\pi}_k(\mathbf{b}_k) = \arg\min_{\mathbf{u}} \tilde{Q}_{\vartheta}(\mathbf{b}_k, \mathbf{u}).$$

The policy $\tilde{\pi}$ is guaranteed to perform at least as well as the base policy π^{base} with respect to the objective function. In fact, it is common for rollout to out-perform its base policy. This is related to the fundamental relation of rollout to policy improvement (see [43]).

The previous two approximations methods, rollout and receding horizon control, are used throughout the simulations in Section 2.4. However, in Section 2.4.4 we apply an additional approximation when Algorithm 1 is employed for data association in the multiple target scenarios. This approximation further limits the amount of computation needed for the Q-value estimation, resembling *completely observable rollout* from [40]. Like completely observable rollout, we endeavour to reduce the computations in the simulations of the future via the reduction of the (simulated) belief state representations. Completely observable rollout accomplishes this by substituting a Dirac delta distribution for the belief states in the rollout approximation of the Q-value and thus

applies a base policy which maps the underlying states to actions. This is an "extreme" approximation of the belief state, but it fairs well in certain situations where such base policies are readily available and easily computed. Our approximation is less drastic than that of completely observable rollout, but it still takes advantage of the knowledge of the underlying system state during the Monte Carlo simulations used for the Q-value approximations.

Recall Algorithm 1 and its hypothesis tree from Section 2.2.2. Although expansion of the hypothesis tree is constrained by the choice of the sliding window size, a large portion of computation is still devoted to the prediction step. During the prediction step the tree must be extended to the next level of nodes representing all possible supports at the pending time step. Furthermore, after this expansion, a Kalman filter update computation is required for each new node in the tree. Upon the culmination of these calculations, the tree is immediately pruned. The pruning process results in a substantial portion of the new nodes being discarded and thus the associated calculations are inconsequential over the remaining time horizon of the scenario. The realization of the limited amount of time, from prediction to pruning, over which these computations are viable motivates our approximation of the belief state for the rollout simulations.

Our belief state reduction, inside rollout, eliminates the computations incurred by this cycle of prediction and pruning. This is accomplished by extending only the tree with nodes representing the best case, where the selected leaf is no longer necessarily the most probable leaf node, but actually represents the "true" support-value pair descending from the "true" route of the sample signal. This is a viable option within rollout since we can access these details for each sample signal in the Monte Carlo simulation. Given this knowledge and a window size of w, at each time step in the ϑ-step rollout horizon we identify the node representing the true route in the tree and trace the route back $w-1$ time steps yielding the routes *initial node*. Then only the $(w-1)$-generation leaves of the initial node are used in the prediction step, resulting in fewer new leaves and fewer Kalman filter updates. In fact, the new leaves are equivalently those which would have remained after the pruning step if the most probable leaf-node, i.e., the one selected by Algorithm 1, was the

true representation of the sample signal. Therefore, we also eliminate the pruning step resulting in a further reduction of computation load.

This extended discussion on Q-value approximation shows that there are obvious trade-offs between myopic and non-myopic policies. The clear benefit of myopic policies is the relatively simple computations of $C_k(\mathbf{b}_k, \mathbf{u})$ required for their *greedy* action choices, based solely on the immediate cost of an action at the current belief state. Often, this alone motivates the implementation of the myopic policy. Conversely, non-myopic approaches must contend with the computational complexity originating from the inclusion of the ECTG term. In particular, rollout must compute the expected cumulative cost of the chosen base policy over $\vartheta - 1$ time steps, and this computation is typically done using Monte Carlo sampling. However, given a scenario where the current action choice substantially impacts the future outlook, rollout provides a distinct advantage due to its lookahead property. The results in the following section empirically compare and contrast the greedy-type policies with rollout.

2.4 Simulation Results

In this section, we present numerical examples for our solutions to the problems introduced in Section 2.1. To better demonstrate the value of adaptivity and in particular multi-step lookahead policies, we consider several scenarios. The first simulation compares non-adaptive methods to adaptive methods, while the remaining simulations compare several adaptive techniques. Since each simulation contains a significant amount of details about its scenario, we briefly explain the goals and the results of running each simulation.

Our first simulation, Simulation 1, considers a very simple case of a steady state 1-sparse signal with a highly informative prior distribution on the location of the nonzero entry. We find solutions for Problem 1 and evaluate the performance of adaptive methods compared to non-adaptive methods. The results of this simulation indicate that even in such a simple scenario, the proposed adaptive methods perform better than the non-adaptive methods. However, multi-step lookahead

does not provide a significant advantage over the 1-step lookahead because of the simplicity of the scenario.

In Simulation 2, an instance of Problem 1, there is a single moving target, i.e., a dynamic 1-sparse signal, that changes its location in an environment with occlusions. When the target occupies a position with an occlusion, the measurements are just noise samples. We assume a uniform prior distribution on the supports, which is less informative than the prior in Simulation 1. These results again show little-to-no difference between the performance of 1-step and multi-step lookahead policies. Even with the augmentation of the problem with occlusions and dynamics, multi-step lookahead does not amass extra *useful* information compared to the 1-step lookahead solution. This relates to the POMDP cost and action space definitions. The simulations based on Problem 2 explicate this further.

In Simulation 3, we use the same scenario as Simulation 2 but we consider solving Problem 2 instead, using a different POMDP action space and cost. Based on this cost, each method must decide between using more measurements or a reduction in support identification accuracy at each time step. The results of this simulation shows a larger performance advantage for multi-step lookahead compared to 1-step lookahead.

Simulation 4 considers Problem 2 with two moving targets in a space with occlusions and assumes semi-informative prior distribution on the supports. Algorithms 1 and 2 from Section 2.2.2 are applied to maintain reduced computations for data association. Again, given the performance criterion and the action space of the problem, the multi-step lookahead outperforms the 1-step lookahead.

In Simulations 1 and 2, we use two libraries of waveforms to build the measurement matrices. The waveforms in each library build a *Grassmannian line packing* (see [44] and [45]) in the space $\mathbb{R}^N$. These libraries share a common property which inspired their use in our simulations; For the given number of waveforms in each respective library, every pair of waveforms in that library achieve a maximum angular distance from one another. This means that the waveforms in each library "slice" the sparse signal space $\mathbb{R}^N$ as equally as possible. Therefore, by using a fraction of

these waveforms as rows of our measurement matrices, we are searching for the sparse signal in a subspace of $\mathbb{R}^N$ in a semi-uniform manner.

In Simulations 3 and 4, we use a method that was originally introduced in [19] to construct the measurement matrix once the number of measurements is chosen at a particular time step. In this method, the function $P_{\mathbf{d}_k|H_k}$ at time step k is used to determine the matrix entries for that time step. This method has also motivated us to implement a similar adaptive scheme in Simulation 2, referred to as the "Bhattacharyya Distance" policy, in which once we build the measurement matrix library from the Grassmannian line packing, we compute a "characterizing" function that specifies how the "energy" of the matrix is distributed along its columns. We then choose the matrix whose energy function has the minimum Bhattacharyya distance to $P_{\mathbf{d}_k|H_k}$ at time step k as the measurement matrix.

2.4.1 Simulation 1: 1-Sparse Static Signal (Problem 1)

In this simulation, we consider a simple static scenario. We assume that the signal $\mathbf{x}_k$ is 1-sparse in $\mathbb{R}^{75}$ and that its nonzero entry stays in the same location at all times. We further assume a specific prior distribution on the support of $\mathbf{x}_k$. The probability mass function (pmf) $P_{\mathbf{d}_0}$ corresponding to this prior is shown in Figure 2.3. This prior indicates the nonzero entry of the signal is located somewhere in the first 16 indices of the signal. This simulation utilizes 100 signal samples and we rerun the experiment 50 times for each sample. Each signal sample is created by drawing a sample from the prior pmf $P_{\mathbf{d}_0}$ for the locations of the target and then drawing samples from a $\mathcal{N}(0, \sigma^2)$ distribution for the amplitudes.

We consider an instance of Problem 1 in this simulation where the number l_k of measurements per step is set to one, and the total time steps (total number of measurements) is $m = 8$. We compare five different (three non-adaptive and two adaptive) methods:

1. Random: In this method, $\mathbf{A}_k$ is a matrix where all its $(l_k \times N)$ entries are i.i.d. samples from a Gaussian distribution $\mathcal{N}(0, 1/N)$. We also normalize the rows of each matrix $\mathbf{A}_k$.

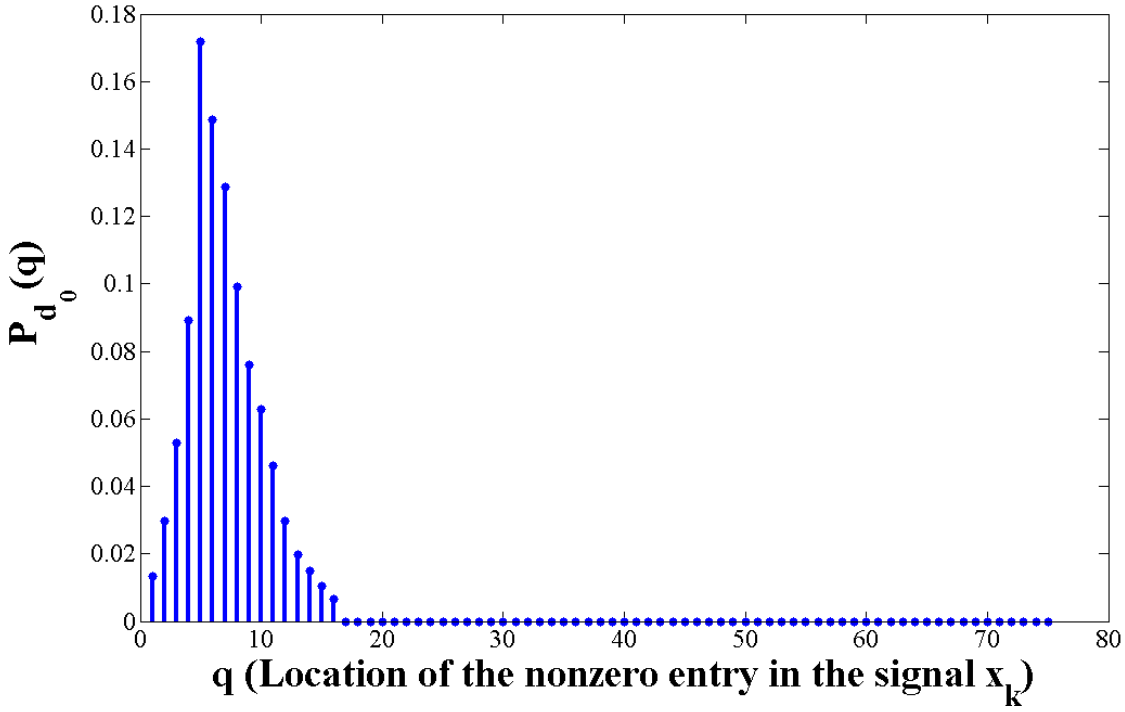


Figure 2.3: Prior structure used for $P_{\mathbf{d}_0}$ in Simulation 1.

2. Limited Random: Knowing the prior $P_{\mathbf{d}_0}$, when assembling rows of the measurement matrix $\mathbf{A}_k$, we divide each measurement row $\mathbf{A}_k(i)$ into two parts: the first part is a 16-dimensional vector, and the second part is a 59-dimensional vector. We fill the first 16 entries with i.i.d. random Gaussian samples and normalize the resulting vector to have norm one. We fill the second part with zeros. This policy relies on knowing the support of the prior, in contrast to the first policy, which does not exploit this information. This policy is to explore the benefits of using a predefined library of vectors that are known to be "good," in contrast to random vectors.

3. Random from Library: Similar to Limited Random, each row of the measurement matrix $\mathbf{A}_k$ is broken into two parts, where the second part is a 59-dimensional vector of zeros. However, the first part of the rows are chosen randomly from a static library of 50 measurement vectors that together, constitute a Grassmannian line packing (see [44] and [45]) in $\mathbb{R}^{16}$.

4. Greedy: In this approach, we also break the measurement rows into two parts like methods 2 and 3 and we determine the first 16-dimensional part of the vector. We set the decision-making horizon to 1, i.e., at each time step k, we choose the best vector from the Grassmannian library that

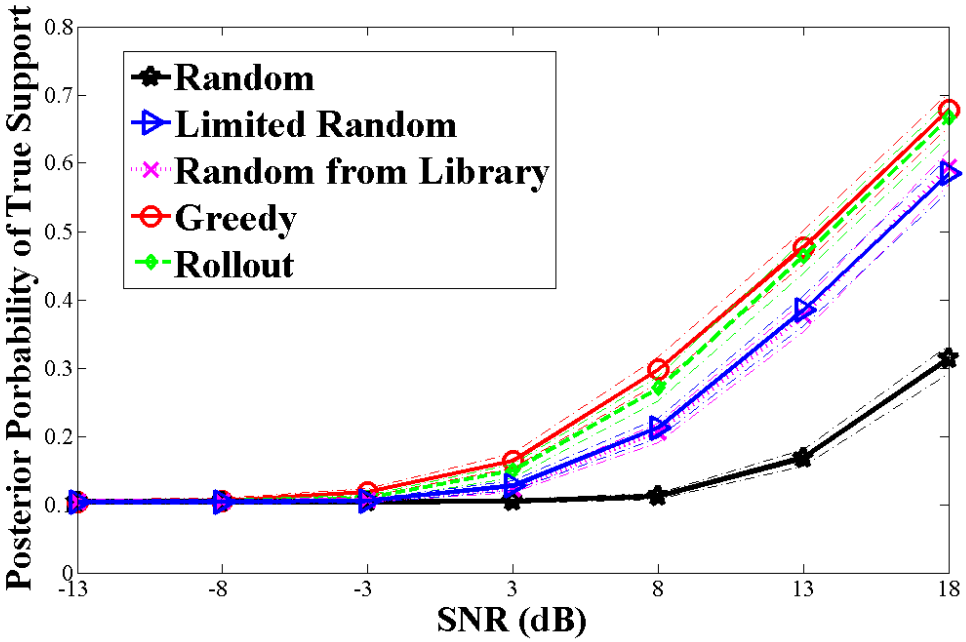
minimizes the 1-step ahead belief cost $C_k(\mathbf{b}_k, \mathbf{u}_k)$. Therefore the actions are chosen in a *greedy* manner.

5. Rollout: This policy is based on the rollout method in Section 2.3, in which at each time step and for each action candidate from the Grassmannian library, the Q-value is approximated over a four step rollout horizon. The estimated Q-value, for each candidate action, is the average of 50 (four step) Q-value samples. The action with the minimum Q-value approximation is selected. The base policy for Rollout is the Random from Library method.

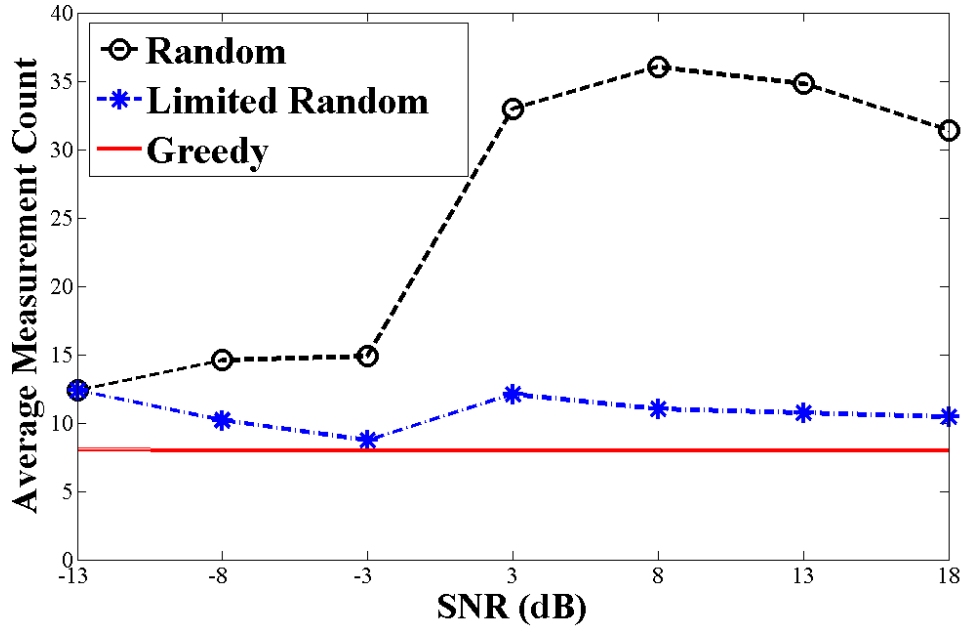
Figure 2.4(a) shows the performance of the five methods introduced above. The metric used in this figure for comparing these methods is the posterior probability of the true support after $m = 8$ measurements, i.e., the value of $P_{\mathbf{d}_8|H_8}(\mathbf{d}_T|H_8)$, where $\mathbf{d}_T$ is the true location of the nonzero entry of the signal. We have shown the performance of these methods for different values of signal-to-noise-ratio (SNR), which is defined as SNR $= \sigma^2/\sigma_w^2$. This figure shows that variations of POMDP, i.e., Greedy and Rollout, perform similar to each other in this simple static scenario, but both perform better than the three non-adaptive methods at moderate and high SNR. At low SNR all methods perform similarly.

A second experiment is run to see, on average, how many more measurements Random and Limited Random require in order to reach the performance of Greedy when the same metric, $P_{\mathbf{d}_8|H_8}(\mathbf{d}_T|H_8)$, is used for comparison. Figure 2.4(b) shows the results for different values of SNR. The number of measurements for Greedy is set to $m = 8$ for all SNRs. The plots suggest that in this simple static scenario with the particular prior used, simply knowing that the true support lies within the first 16 indices provides significant performance gains over the random scheme. Moreover, adaptation only provides marginal gains over methods that exploit the highly informative prior.

We anticipate similar conclusions in the context of Problem 2. That is, due to the simple and static nature of the problem, multi-step lookahead does not offer much advantage over the single-step lookahead. Also, we anticipate that adaptivity would again offer minimal gain over the Limited Random method, which exploits the highly informative prior used in this scenario.



(a)



(b)

Figure 2.4: (a) Performance comparison of all methods in Simulation 1. The dashed lines indicate the 95% confidence intervals, (b) Average number of measurements required for Random and Limited Random to reach the performance of Greedy in Simulation 1.

2.4.2 Simulation 2: 1-Sparse Time-Varying Signal (Problem 1 with Occlusions)

We consider a 1-sparse dynamic signal in $\mathbb{R}^{20}$ for this simulation. Differing from Simulation 1, we assume a uniform prior for $P_{\mathbf{d}_0}$ and the presence of occlusions in certain areas. When the signal support corresponds to occluded locations, the received observations are simply noise samples, i.e., $\mathbf{y}_k = \mathbf{w}_k$. The occlusions for Simulation 2 are located at indices 5, 6, 11, and 12. The movement of the target is modeled by the transition law shown in Figure 2.5. The initial position of the true target is located at the second index and the target moves one position to the right until reaching index 20. At this point, the target begins shifting in the reverse direction back to position one, where the target will again reverse its direction of movement. The horizon of Simulation 2 is 40 time steps, so the target will reverse direction twice. Furthermore, the target will be occluded at times $k = \{4, 5, 10, 11, 27, 28, 33, 34\}$. Finally, we set SNR $= 14$ (dB) and run the experiment 36 times for both the Greedy and Rollout and 300 times for the both the Random from Library and the Bhattacharyya Distance algorithms that we define below.

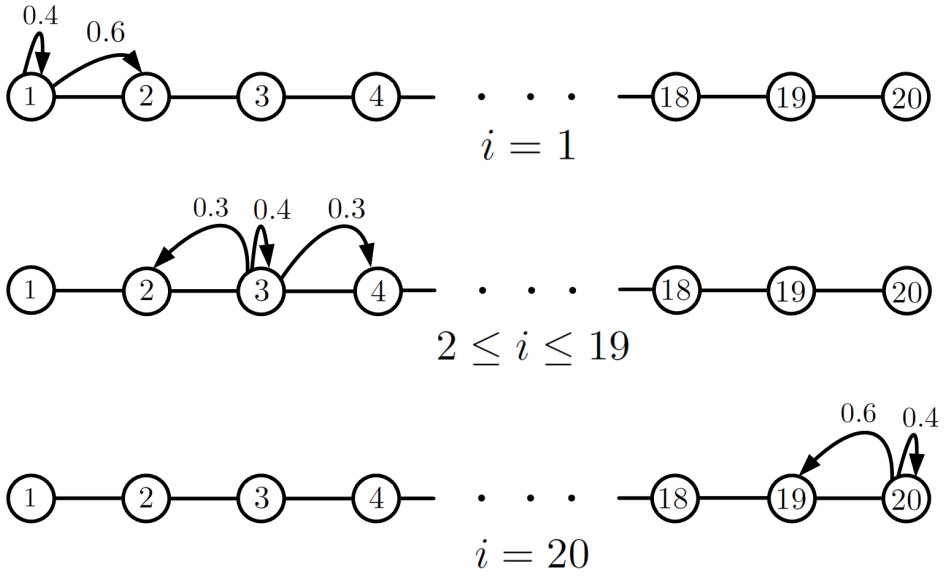


Figure 2.5: State transition law for the moving target.

Similar to Simulation 1, the scenario of interest is based on Problem 1 where $l_k = 5$. The policies compared in this set of simulations consists of one non-adaptive and three adaptive schemes. The action selection is again among a set of measurement matrices. To build this set, we first import a Grassmannian packing consisting of 16 vectors in $\mathbb{R}^{16}$. The 16 vectors that build this Grassmannian packing are then combined into different sets consisting of five vectors. Note that we are considering every possible combination (without repeats) of 5 vectors among 16 vectors, resulting in 4368 sets of vectors. Using the vectors in each set, we build a (5×16) matrix. These matrices are then augmented by inserting four columns of zeros in the respective occlusion positions. These 4368 matrices now comprise the library of compressive measurement matrices.

Due to this large action space we developed a heuristic to restrict the set of possible actions based on the prior distribution on the support at each time step. The restricted sets of actions are generated in advance by a clustering algorithm know as *K-medoids* (see [46]). To begin the clustering process, we form a power-vector to accompany each matrix in the library. These power-vectors help us "classify" the matrices. The power-vector $\boldsymbol{\phi}_{\mathbf{A}}$ of a matrix $\mathbf{A}$ is a (20×1) vector where its jth component $\boldsymbol{\phi}_{\mathbf{A}}(j)$ is defined as

$$\boldsymbol{\phi}_{\mathbf{A}}(j) = \frac{\sum_{i=1}^{5} |\mathbf{A}(i)(j)|}{\sum_{j=1}^{20} \sum_{i=1}^{5} |\mathbf{A}(i)(j)|},$$

where $\mathbf{A}(i)(j)$ is the entry of the matrix $\mathbf{A}$ located at its ith row and its jth column. These power-vectors portray how the power of the matrix is "distributed" over its different columns and resemble probability mass functions. The matrix library is then sorted into 400 clusters by the K-medoids algorithm, using the Bhattacharyya distance to classify the measurement matrices via their corresponding power-vectors. Once the power-vectors, and thus the matrices, are classified, each cluster contains a *medoid* as a representative member. During the simulations (online), the available actions at time step k are identified as the cluster ${\mathcal{A}_{\mathcal{G}}}^{(k)}$, whose medoid achieves the minimum Bhattacharyya distance to $\mathcal{P} = \mathbf{T} \times P_{\mathbf{d}_{k-1}|H_{k-1}}$, where $\mathbf{T}$ is the transition probability matrix associated with Figure 2.5. These action restrictions are implemented for the adaptive methods described below.

1. Random from Library: This is the only non-adaptive scheme considered in Simulation 2. It does not abide by the action restrictions of available measurement matrices. Simply, a single measurement matrix is chosen from the library of 4368 matrices at random.

2. Bhattacharyya Distance: This is an adaptive heuristic inspired by the findings in [19] where the measurement should resemble the prior. This is really an extension of the sorting procedure used for clustering measurement matrix library ${\mathcal{A}_{\mathcal{G}}}^{(k)}$. Once the closest medoid of the 400 groups is identified, the Bhattacharyya distance is again used to rank the measurement matrices in the minimum medoid's cluster. The action selected is the measurement matrix with the minimum distance from its power-vector to the predicted support distribution $\mathcal{P}$.

3. Greedy: This action selection process is based *single* step decision-making horizon. Greedy, as with the Bhattacharyya distance, only considers the available actions based on the medoid selection. At each time step k, the best matrix is chosen from the restricted measurement matrix library ${\mathcal{A}_{\mathcal{G}}}^{(k)}$ minimizing the one-step ahead belief cost.

4. Rollout: This solution method selects an action, $u_k \in {\mathcal{A}_{\mathcal{G}}}^{(k)}$ that minimizes the 3-step lookahead Q-value approximation. The base policy used in this approximation is the Bhattacharyya Distance (method 2).

Fig. 2.6 shows the performance of each method described above over 40 time steps. As expected, the three adaptive methods outperform the non-adaptive method (except in select occlusion areas). However, Greedy and Rollout perform similarly. This simulation shows an example of a case where the multi-step lookahead approach is not gaining extra information compared to the myopic method.

2.4.3 Simulation 3: 1-Sparse Time-Varying Signal (Problem 2 with Occlusions)

We consider another scenario involving a moving target with occlusions. Here, a single target moves among $N = 40$ possible locations. The transition law that describes the movement of this target is shown in Figure 2.5. Like in Simulation 2, we assume a uniform distribution as the prior

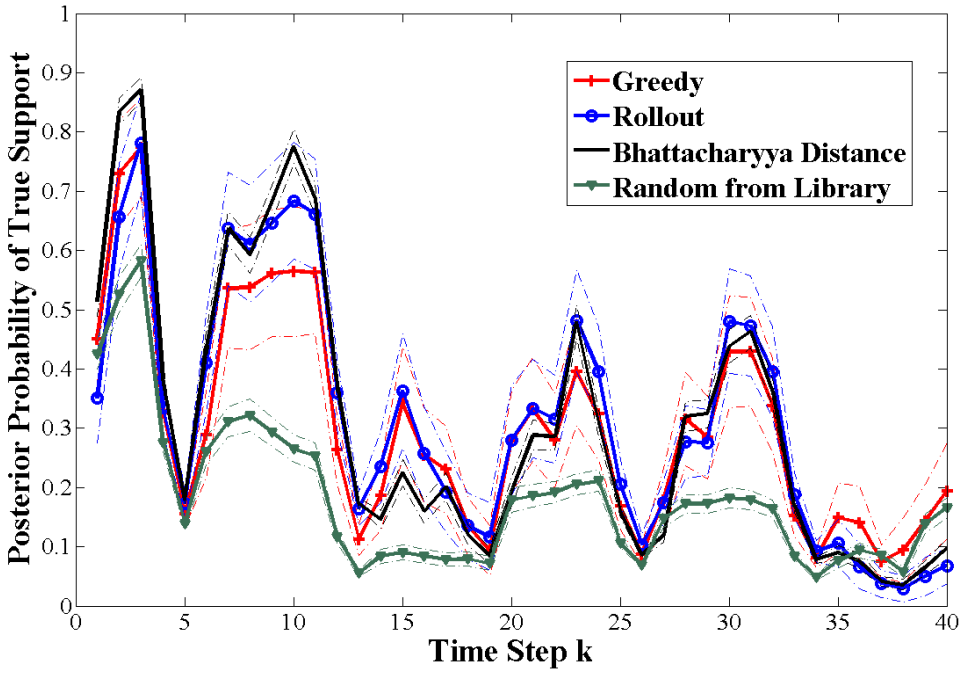


Figure 2.6: Performance comparison of the four policies described in Simulation 2. The dashed lines indicate the 85% confidence intervals.

$P_{\mathbf{d}_0}$, but the occlusions are now located at indices 6 to 10, 16 to 20, and 36 to 40. For simplicity, we consider the case where the target starts in location 2 and moves one location to the right at each time step. This means that over the duration of $m = 10$ time steps, the target will be behind occlusion points from time step 5 to 9. We set γ used in (2.1) to 300 and we set SNR $= 2.5$ (dB). Finally, in this simulation, we use one signal sample and repeat the experiment 50 times.

We consider this scenario in the context of Problem 2. Here, we adaptively select the number of measurements taken at each time step. We set $l_{\max} = 65$, meaning at most 65 measurements are requested per time step. After the number of measurements l_k is determined, we use a fixed scheme to design the measurement matrix $\mathbf{A}_k$. The following describes the scheme used in this simulation: At each time k the jth entry of the measurement row $\mathbf{A}_k(i)$, i.e., $\mathbf{A}_k(i)(j)$, is generated in the following way:

$$\mathbf{A}_k(i)(j) = \begin{cases} 0, & j \text{ is an occlusion point,} \\ x_{P_{\mathbf{d}_k|H_k}(j|H_k)}, & \text{otherwise,} \end{cases}$$

where the value $x_{P_{\mathbf{d}_k|H_k}(j|H_k)}$ is a sample drawn from the normal distribution $\mathcal{N}(0, P_{\mathbf{d}_k|H_k}(j|H_k))$. Each row vector $\mathbf{A}_k(i)$ is then normalized. The idea of using the function $P_{\mathbf{d}_k|H_k}$ to create row entries in the measurement matrix $\mathbf{A}_k$ comes from [19], one of the early works in adaptive design (1-step) of compressive measurement matrices.

In this simulation, we consider three methods of action selection:

1. Greedy: The best number of measurements which minimizes the 1-step lookahead POMDP belief cost.

2. Smart: A specific number of measurements $l_k \leq l_{\max}$ is selected using the following rule:

$$l_k = \begin{cases} 0, & 0.95 < \alpha, \\ 5, & 0.8 < \alpha \leq 0.95, \\ 15, & 0.65 < \alpha \leq 0.8, \\ 45, & 0.25 < \alpha \leq 0.65, \\ 35, & 0.15 < \alpha \leq 0.25, \\ 25, & \alpha \leq 0.15, \end{cases} \tag{2.10}$$

where α is the fraction of $P_{\mathbf{d}_k|H_k}$ at time step k that belongs to occlusion points. This policy suggests that when α implies the target is somewhere far from the occlusion points with high probability, a fair (but not exorbitant) number of measurements is used. When the target approaches the occlusion points, the number of measurements is increased to achieve an accurate estimate of the target location before it "disappears." Once the target is occluded, there is no point in making any more measurements until the target becomes visible again.

3. Rollout: This is a 3-step lookahead solution method. Rollout chooses the action that minimizes the total (approximate) cost incurred over three consecutive time steps. For this method, the

Smart policy is the base policy of Rollout. In each run, 150 Rollout trajectories are averaged to estimate the Q-value of each action candidate.

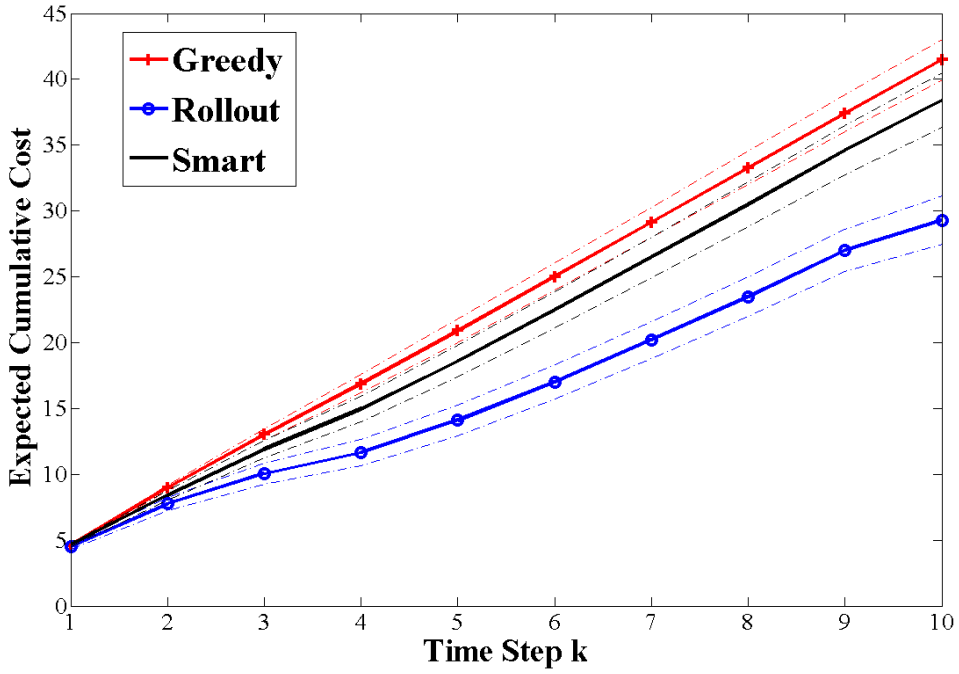
Figure 2.7(a) shows the expected cumulative cost of the above three methods. This figure shows that the Rollout method outperforms the Greedy method, and demonstrates the value of multi-step lookahead. Moreover, it is interesting to see that both Greedy and Smart methods perform similarly while the amount of computation required for the Smart policy is much less than that of the Greedy policy.

We have also plotted the posterior probability of true support, i.e., $P_{\mathbf{d}_k|H_k}(\mathbf{d}_T|H_k)$, at each time step k in Figure 2.7(b). This figure clearly captures the ability of each method tracking the moving target. When the target is outside the occlusion area, Rollout improves the ability to detect the target location better than the other two methods over time.

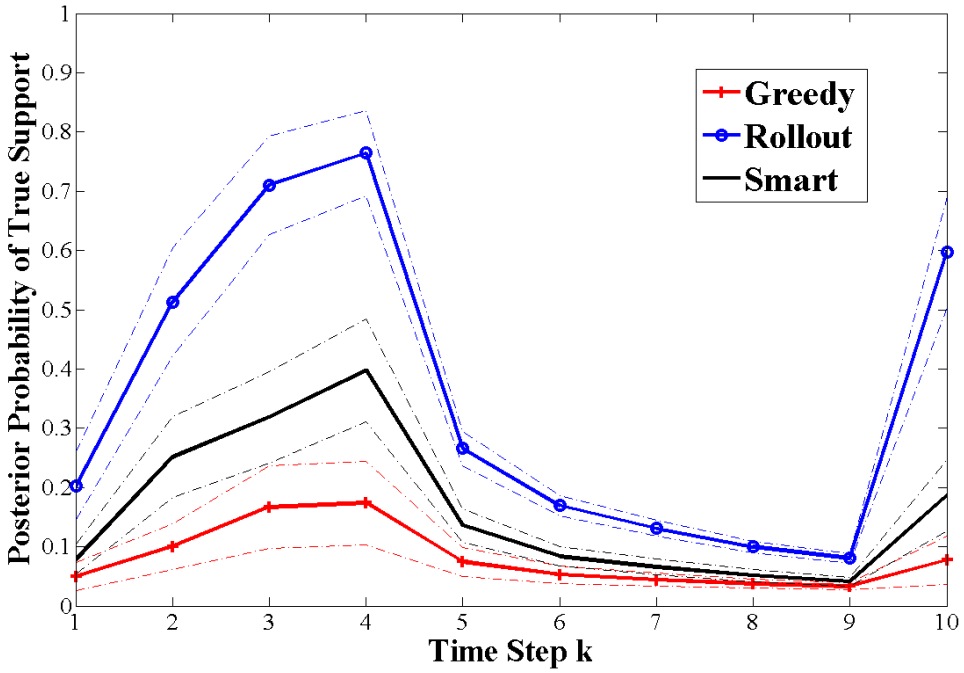
2.4.4 Simulation 4: 2-Sparse Time-Varying Signal (Problem 2 with Occlusions)

Now, we consider a more general setting in which there are multiple moving targets, i.e., a dynamic s-sparse signal. To keep the amount of computation low, we make several simplifying assumptions. First, we only consider $s = 2$ targets that are moving in $\mathbb{R}^{20}$. Note that even for this case, there are $N^s = 400$ possibilities for the support $\mathbf{d}_k$ at each time step. Second, we consider an initial distribution where only 25% of its entries are nonzero with equal values. This prior distribution suggests that the first target is located at one of the locations 1, 2, or 3 and the second target is located either at location 19 or 20. We let $\sigma_w^2 = 1$, and consider the strength values 1.7 and -1.5 for the first and second target, respectively, which means that SNR $= 4$ dB.

Targets 1 and 2 are initially located at positions 1 and 19, and as time evolves they move towards each other one index at a time. The transition law that describes the movement of each of these targets is the same transition law that was used in Simulation 3 and shown in Figure 2.5. Note that these targets move independently from each other. Finally, we consider five occlusion points located at positions $4, 5, 8, 12,$ and 13.



(a)



(b)

Figure 2.7: Performance comparison of all methods in Simulation 3. The dashed lines indicate the 90% confidence intervals.

Our simulation runs for 10 time steps. During this time, the two targets temporarily coincide as well as occupy occluded locations. Table 2.1 summarizes the position of each target at each time step. This table indicates those time steps in which an occlusion occurs by a "$*$" sign next to the position of the target. Based on this table, we expect to see a drop in the performance of all the methods at time steps $3, 4, 6,$ and in particular 7 (where both targets are occluded). Moreover, at time step 9, when the two targets reach the location 10, the strength value is equal to the sum of the strength values of the two targets. This means that at time $k = 9$, the measurements collected are from a 1-sparse signal with the strength value $1.7 - 1.5 = 0.2$. Thus, the SNR value drops to -14 dB and dip in performance is expected for all methods.

Table 2.1: Support positions over 10 time steps in Simulation 4.

k	1	2	3	4	5	6	7	8	9	10
$\mathbf{d}_k(1)$	2	3	4*	5*	6	7	8*	9	10	11
$\mathbf{d}_k(2)$	18	17	16	15	14	13*	12*	11	10	9

Similar to the previous simulations, the performance is compared across four approaches in this simulation:

1. Fixed: This is a non-adaptive method, where a fixed value of 30 measurements are taken at each time step.

2. Smart: This method is similar to the Smart method in Simulation 3 but with two differences: a) The method has been modified to work for 2 targets, and b) The number of measurements for the different cases in (2.10) changes from 0, 5, 15, 45, 35, and 25 to 5, 10, 20, 55, 40, and 30, respectively.

3. Greedy: Similar to the myopic methods in previous simulations, this is the 1-step lookahead solution.

4. Rollout: This method is the 3-step lookahead method with Smart, introduced here, as the base policy.

To estimate the expected cost in both Greedy and Rollout, we generate 25 samples from the belief state at each time step and we repeat the experiment for each trajectory 20 times.

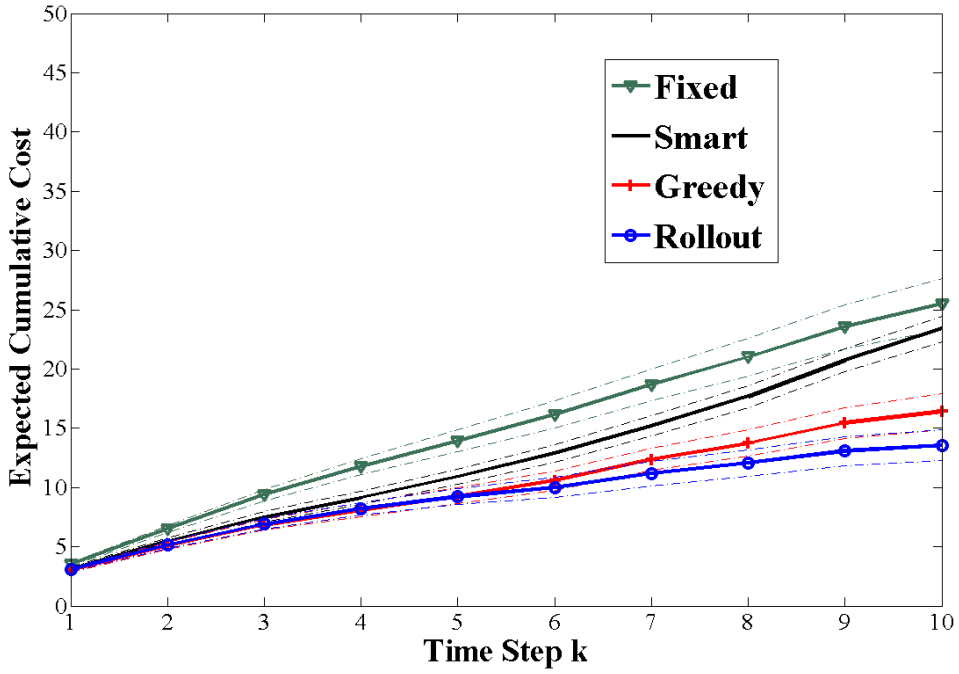
In all of the techniques introduced, once the number of measurements l_k is selected at each time step, we build the measurement matrix the same as in Simulation 3. Moreover, the maximum number of measurements allowed at each time step is $l_{\max} = 65$ and we set γ to be equal to 293. The values shown in the results here are the average values taken over 36 rounds of simulation.

As mentioned in Section 2.2.2, we implement two heuristics, Algorithms 1 and 2, to decrease the amount of computation. In one experiment, Algorithm 1 limits the belief state distribution expansion by maintaining a hypothesis tree with a sliding window of size $w = 4$, resulting in approximately 6000 Kalman filters per time step, after the time step $k = 4$ (and significantly more before this first pruning action). This is clearly more Kalman filter computations than Algorithm 2, but yields in a higher probability of support identification. In a separate simulation, Algorithm 2 approximates the conditional posterior distribution of the support $P_{\mathbf{d}_k|H_k}$ with a distribution $\hat{P}_{\mathbf{d}_k|H_k}$ that contains only 90 nonzero entries for $k > 1$. Therefore, at any point in our simulation (after the first time step), the belief state update requires only 90 Kalman filters.

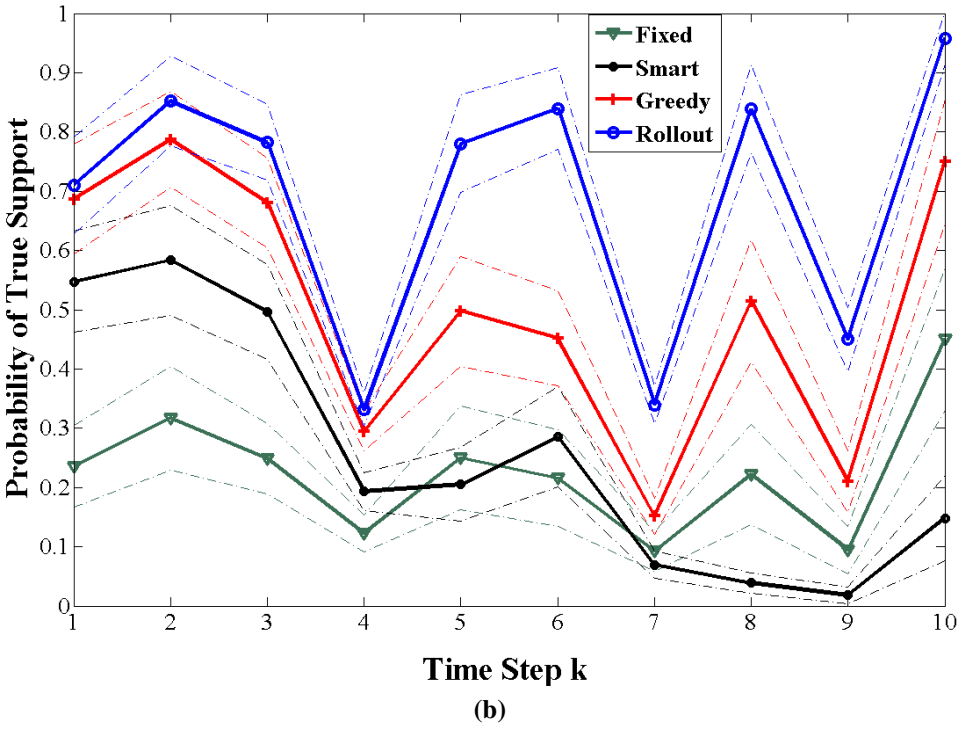
Figure 2.8 and Figure 2.9 show the results from running this simulation when Algorithms 1 and 2 are used, respectively. In both set of results, similar to Simulation 3, Rollout has the best performance among all the methods. Also, notice the performance drop of all methods, as expected, in the occlusions and the low SNR areas shown in Table 2.1.

2.5 Conclusions

This chapter studied the problem of adaptively determining compressive measurements for estimating supports of time-varying sparse signals in the context of multi-target tracking. We formulated this problem as a POMDP and used the rollout method to find an approximate solution. We provided several simulations that consider different scenarios for both single moving target (1-sparse) and multiple moving targets (s-sparse). The simulation results indicate that the adaptive techniques have a performance better than the non-adaptive (traditional) designs. Moreover, when

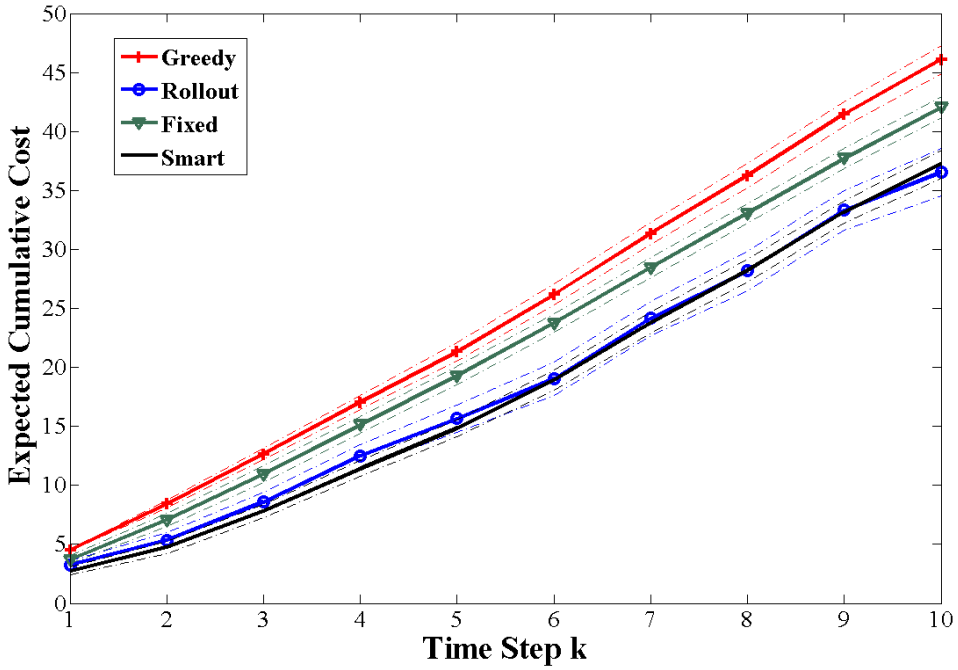


(a)

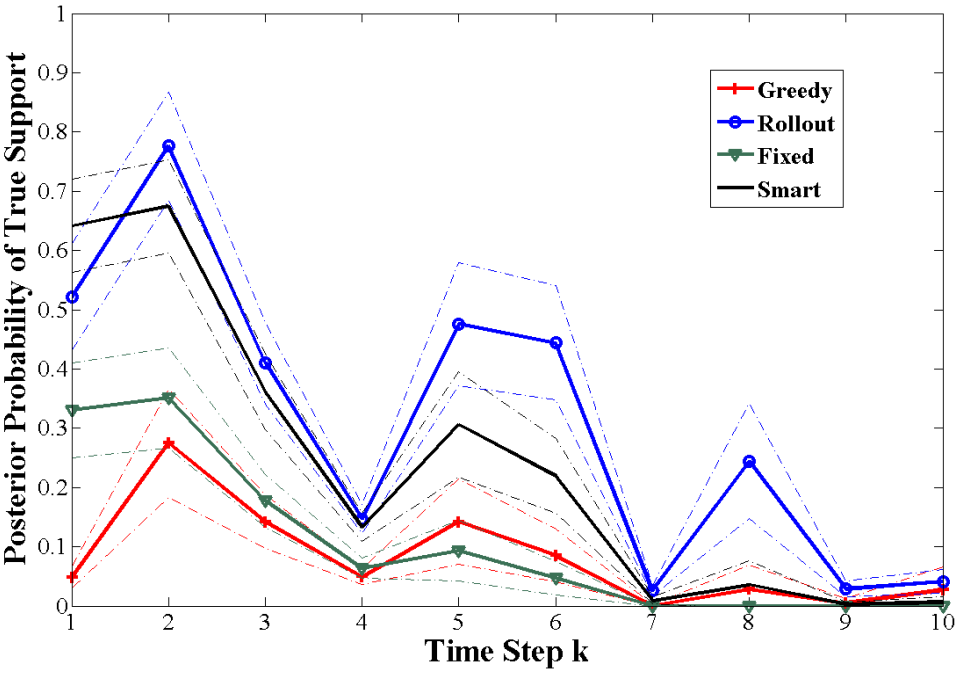


(b)

Figure 2.8: Performance comparison of all methods in Simulation 4 when Algorithm 1 is used. The dashed lines indicate the 85% confidence intervals.



(a)



(b)

Figure 2.9: Performance comparison of all methods in Simulation 4 when Algorithm 2 is used. The dashed lines indicate the 85% confidence intervals.

comparing the myopic vs. non-myopic variations, the scenario specifications and the POMDP cost play a major role in their relative performance. For example, when we consider occlusion areas in the setting and the goal is to minimize the total number of measurements, the non-myopic design has the best performance. On the contrary, in cases where neither of these conditions hold, there is no difference between the performance of myopic and non-myopic designs and using the latter design only increases the computation cost.

Chapter 3

Adaptive Compressive Sensing in the Presence of Noise and Erasure

Adaptive measurement scheduling for compressive sensing has been studied in the static signal setting (e.g., [19]), where the support remains unchanged over time, and in the dynamic signal setting (e.g., [24,26,30]), where the signal support varies over time. In both cases, adaptive scheduling strives to form a closed-loop control system where the measurement selection is based on the optimization of an objective function characterizing the utility of available measurements given the current signal estimate.

We investigate the support identification of an s-sparse dynamic signal $\mathbf{x}_k \in \mathbb{R}^N$. We collect observations $\mathbf{y}_k = \mathbf{A}_k\mathbf{x}_k + \mathbf{w}_k$, where $\mathbf{A}_k \in \mathbb{R}^{l\times N}$ is the measurement matrix and $\mathbf{w}_k$ is a multivariate normal random vector with the zero mean vector $\mathbf{0}_l$ and the covariance matrix $\sigma^2_{\mathbf{w}}\mathbf{I}_l$ ($\mathbf{I}_l$ is the $(l \times l)$ identity matrix). The scenario also accounts for periods of *erasure*, where the measurement return is pure noise. Furthermore, we place a restriction motivated by a covertness objective on the measurement selection process; no row of the measurement matrices can be repeated over *consecutive* time steps, i.e., times k and $k + 1$. Our performance criterion is the conditional mutual information between the observation and the support of the sparse signal given the previous observations and decisions. This study extends our previous work [30].

Given the above scenario and objectives, we formulate the problem as a partially observable Markov decision process (POMDP) [27] in Section 3.1. This formulation is a natural approach when both short-term and long-term rewards must be considered. The measurement matrix selection in the POMDP is based on the recursive calculation of the *belief state*, the posterior distribution of the underlying state given past observations and actions. We discuss the belief state update in Section 3.1.1 and an approximation method to reduce computations in Section 3.1.2. The POMDP framework also lends itself to several solution approaches. Our main focus is on the simulation

based multi-step lookahead Q-value approximation method *rollout* (Section 3.2), but we also employ the same POMDP framework for two other solution methods in our empirical comparisons (Section 3.3).

3.1 POMDP Formulation

The following descriptions detail the POMDP components.

States and Transition Law: The s-sparse signal $\mathbf{x}_k$ is fully characterized by its nonzero entries and their locations, the strength values and support, respectively. Therefore, the state of the POMDP at time k is $\mathbf{s}_k = (\mathbf{d}_k, \mathbf{v}_k)$, where $\mathbf{d}_k$ and $\mathbf{v}_k$ are $(s \times 1)$ vectors. Their ith entries $\mathbf{d}_k(i)$ and $\mathbf{v}_k(i)$ represent the location and the strength value of the ith non-zero entry, respectively. Define the set Ω to be $\Omega = \{1, \ldots, N\}$ and Ω_s as the set containing all the $(s \times 1)$ vectors $\mathbf{d}$ such that $\mathbf{d}(i) \in \Omega$ for $i = 1, \ldots, s$. The POMDP state space is then defined as the Cartesian product $\mathcal{S} = (\Omega_s \times \mathbb{R}^s)$.

The state transition law defines how the signal of interest changes over time. Here, this law is independent of the action selection. Assuming that the strength values remain constant over time, the state transition law is

$$P\{\mathbf{s}_{k+1} = (\mathbf{d}, \mathbf{v}) | \mathbf{s}_k = (\mathbf{e}, \mathbf{z})\} = \begin{cases} p_{\mathbf{ed}}, & \mathbf{v} = \mathbf{z}, \\ 0, & \text{otherwise}, \end{cases}$$

where $p_{\mathbf{ed}}$ is the probability that the support represented by the vector $\mathbf{e}$ transitions to locations represented by the vector $\mathbf{d}$.

Actions: The action $\mathbf{u}_k$ at each time k is the selection of a measurement matrix $\mathbf{A}_k$. The action space $\mathcal{A}$ is a subset of all matrices $\mathbf{A} \in \mathbb{R}^{l \times N}$ such that, for each row $\mathbf{A}(i), i = 1, \ldots, l$, $\|\mathbf{A}(i)\|_2 = 1$. The set of *valid* actions $\mathcal{A}_{\mathcal{G}}{}^{(k)} \subset \mathcal{A}$ at time k is further restricted (for covertness) such that no matrices $\mathbf{A} \in \mathcal{A}_{\mathcal{G}}{}^{(k)}$ contain rows from $\mathbf{A}_{k-1}$.

Observations and Observation Law: The observations are the measurements $\mathbf{y}_k$ at each time step k. Given $\mathbf{s}_k = (\mathbf{d}, \mathbf{v})$ and the matrix $\mathbf{A}_k = \mathbf{A}$ at time k, the observation law can be described

by: $\mathbf{y}_k|\left((\mathbf{d},\mathbf{v}),\mathbf{A}\right) \sim \mathcal{N}(\mathbf{A_d}\mathbf{v}, \sigma_{\mathbf{w}}^2\mathbf{I}_l)$, where $\mathbf{A_d}$ is a matrix whose ith column is the $\mathbf{d}(i)$th column of matrix $\mathbf{A}$.

Cost: Let $I(\mathbf{y}_k;\mathbf{d}_k|H_k)$ be the conditional mutual information between the signal support $\mathbf{d}_k$ and the observation $\mathbf{y}_k$ given the history $H_k = \{\mathbf{u}_1,\mathbf{y}_1,\ldots,\mathbf{u}_k,\mathbf{y}_k\}$. The per-time-step cost is simply defined as $c_k(\mathbf{s}_k,\mathbf{u}_k) = -I(\mathbf{y}_k;\mathbf{d}_k|H_k)$.

Belief State: The belief state $\mathbf{b}_k$ at time k is the (conditional) posterior distribution of the state $\mathbf{s}_k$ given the history H_k.

Given the belief state definition, our POMDP can be presented as an equivalent Markov Decision Process (MDP) (see [41]). This *belief MDP*, has the following components: 1) A state space consisting of all the possible belief states $\mathbf{b}_k$ for the POMDP. 2) A *belief state update* that is computed using the state transition law and the observation law of the POMDP. 3) An action space identical to that of the POMDP. 4) A *belief cost* $C_k(\mathbf{b}_k,\mathbf{u}_k) = \mathbf{E}\left[c_k(\mathbf{s}_k,\mathbf{u}_k)|H_k,\mathbf{b}_k\right]$, where the expectation is with respect to the belief state $\mathbf{b}_k$.

Using the belief MDP, we can now apply the approximate solution methods available in the literature, specifically rollout [42]. But, we first describe the belief state update and an approximation technique.

3.1.1 Belief State Update

In order to provide concise equations representing the belief state update, we define the related terminology and notations. Fig. 3.1 illustrates an example of the belief state progression over time involving three possible supports and several possible *routes* represented by the branches of the trees. Each route consists of multiple consecutive *links* connecting two *nodes*, where each node represents a possible support at a given time k. Thus, a link is associated with the support transition and a route is a set of consecutive transitions over a time horizon.

Notationally, possible signal support instances (nodes) will be represented by $\mathbf{e}_i$, where the subscript i identifies a unique support. Given each possible link at time k, we define an ordered tuple $\boldsymbol{\tau} = (\boldsymbol{\tau}(1),\boldsymbol{\tau}(2),\boldsymbol{\tau}(3),\boldsymbol{\tau}(4))$ whose four elements are: the terminal node, the initial node,

the prior weight value, and the prior density function attached to the initial node. Letting $P_{\mathbf{d}_k|H_k}$ be the conditional probability mass function of $\mathbf{d}_k$, and $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ be the conditional density function of $\mathbf{v}_k$ at time k, we write the tuple for the link from $\mathbf{e}_3$ to $\mathbf{e}_2$ at time $k = 1$ in Fig. 3.1 as $\boldsymbol{\tau} = \big(\mathbf{e}_2, \mathbf{e}_3, P_{\mathbf{d}_1|H_1}(\mathbf{e}_3|H_1), f_{\mathbf{v}_1|\mathbf{d}_1,H_1}(\cdot|\mathbf{e}_3, H_1)\big)$.

Furthermore we define $\mathcal{T}_k$ to be the set containing all the possible tuples at time k, and $\mathcal{T}_{k,\mathbf{d}}$ to be the set of all tuples representing links ending with support $\mathbf{d}$ at time k, $\mathcal{T}_{k,\mathbf{d}} = \{\boldsymbol{\tau} : \boldsymbol{\tau} \in \mathcal{T}_k, \boldsymbol{\tau}(1) = \mathbf{d}\}$. At each time step k, the number of possible links, which depends on the number of initial nodes and the state transition law, determines the cardinality $|\mathcal{T}_k|$ of the set $\mathcal{T}_k$, which increases exponentially as time evolves.

Routes also have associated tuples. The ith route at time k is represented by the k-tuple $\boldsymbol{\rho}_k^{(i)}$. The first component $\boldsymbol{\rho}_k^{(i)}(1)$ of this tuple is the current node of this route and the last component $\boldsymbol{\rho}_k^{(i)}(k)$ is the initial node of this route. For example, for the route outlined in red in Fig. 3.1, which we refer to as the ith route, we have the following tuple at time $k = 3$: $\boldsymbol{\rho}_3^{(i)} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$. Note that at time k, the total number of possible routes is $|\mathcal{T}_k|$ and therefore has the same exponential growth-rate. We will discuss this issue in Section 3.1.2.

Recall that the state of the POMDP at time step k is $\mathbf{s}_k = (\mathbf{d}_k, \mathbf{v}_k)$. The belief state $\mathbf{b}_k$ is the posterior joint distribution of $\mathbf{d}_k$ and $\mathbf{v}_k$ given H_k, or equivalently, the functions $P_{\mathbf{d}_k|H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$. Given $\mathbf{y}_k = \mathbf{y}$ and $\mathbf{u}_k = \mathbf{u}$, the functions $f_{\mathbf{v}_k|\mathbf{d}_k,H_k}$ and $P_{\mathbf{d}_k|H_k}$ can be computed from $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ using Bayes' rule and the law of total probability by:

$$f_{\mathbf{v}_k|\mathbf{d}_k,H_k}(\mathbf{v}|\mathbf{d}, H_k) = \frac{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} g(\mathbf{v}, \mathbf{d}, \boldsymbol{\tau}) P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}, \tag{3.1}$$

and

$$P_{\mathbf{d}_k|H_k}(\mathbf{d}|H_k) = \frac{\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{d}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{d}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}{\sum_{\mathbf{e}\in\Omega_s}\sum_{\boldsymbol{\tau}\in\mathcal{T}_{k,\mathbf{e}}} P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\mathbf{e}|\boldsymbol{\tau}(2), H_k)\boldsymbol{\tau}(3)}, \tag{3.2}$$

where $g(\mathbf{v}, \mathbf{d}, \boldsymbol{\tau}) = f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\mathbf{d}, \boldsymbol{\tau}(2), H_k)$. Note that $\boldsymbol{\tau}(3)$ is the prior weight of the initial node $\boldsymbol{\tau}(2)$ for link $\boldsymbol{\tau}$. Therefore, to update the belief state $\mathbf{b}_k$, it is sufficient to update (3.1) and (3.2) for all the possible links allowed by the state transition law at time k.

We assume a dynamic linear model and an initial Gaussian distribution for $\mathbf{v}_0|\mathbf{d}_0$ given each $\mathbf{d}_0 \in \Omega_s$. Each distribution can be characterized by a density function $f_{\mathbf{v}_0|\mathbf{d}_0}$ with the mean vector $\boldsymbol{\mu}_{\mathbf{d_0}}$ and the covariance matrix $\mathbf{C}_{\mathbf{d}_0}$. Consequently, the density function $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$, $k \geq 1$, will be Gaussian with mean $\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$ and covariance matrix $\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}$.

Given the action $\mathbf{u}_k = \mathbf{u}$, the measurements $\mathbf{y}_k = \mathbf{y}$, and the history H_{k-1}, the functions $P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}$ and $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}$ can be computed using the following two steps:

1. For a given link $\boldsymbol{\tau} \in \mathcal{T}_k$, the following system of linear equations describes the dynamics of the strength values:

$$\begin{cases} \mathbf{v}_k = \mathbf{v}_{k-1}, \\ \mathbf{y}_k = \mathbf{A}_{\boldsymbol{\tau}(1)}\mathbf{v}_k + \mathbf{w}_k. \end{cases}$$

In this system, $\mathbf{v}_k \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}, \mathbf{C}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)})$. The vector $\mathbf{v}_{k-1}$ also has a Gaussian distribution represented by the density function $\boldsymbol{\tau}(4)$. Therefore, the values $\boldsymbol{\mu}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}$ and $\mathbf{C}_{\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2)}$ for each support pair $(\mathbf{d}_k, \mathbf{d}_{k-1}) = ((\boldsymbol{\tau}(1), \boldsymbol{\tau}(2))$ can be computed using a separate Kalman filter. Once this update is completed, the value of the function $f_{\mathbf{v}_k|\mathbf{d}_k,\mathbf{d}_{k-1},H_k}(\mathbf{v}|\boldsymbol{\tau}(1), \boldsymbol{\tau}(2), H_k)$ for any $\mathbf{v} \in \mathbb{R}^s$ can then be computed.

2. For tuple $\boldsymbol{\tau} \in \mathcal{T}_k$, the posterior weight value is computed as

$$P_{\mathbf{d}_k|\mathbf{d}_{k-1},H_k}(\boldsymbol{\tau}(1)|\boldsymbol{\tau}(2), H_k) = \frac{f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\tau}, \boldsymbol{\tau}) f_2(\boldsymbol{\tau}, \boldsymbol{\tau})}{\sum_{\boldsymbol{\nu} \in \mathcal{T}_k} f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\tau}) f_2(\boldsymbol{\nu}, \boldsymbol{\tau})},$$

where

$$f_1(\mathbf{y}, \mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\tau}) = f_{\mathbf{y}_k|\mathbf{d}_k,\mathbf{d}_{k-1},\mathbf{u}_k,H_{k-1}}(\mathbf{y}|\boldsymbol{\nu}(1), \boldsymbol{\tau}(2), \mathbf{u}, H_{k-1})$$

and $f_2(\boldsymbol{\nu}, \boldsymbol{\tau}) = P_{\mathbf{d}_k|\mathbf{d}_{k-1}}(\boldsymbol{\nu}(1)|\boldsymbol{\tau}(2))\boldsymbol{\tau}(3) = p_{\boldsymbol{\tau}(2)\boldsymbol{\nu}(1)}\boldsymbol{\tau}(3)$. The value $p_{\boldsymbol{\tau}(2)\boldsymbol{\nu}(1)}$ can be computed from the state transition law and is equal to zero for non-existing links. It can be shown that the function f_1 represents the Gaussian distribution denoted as $\mathcal{N}(\mathbf{A}_{\mathbf{d}_k}\boldsymbol{\mu}_{\mathbf{d}_k|\mathbf{d}_{k-1}}, \mathbf{A}_{\mathbf{d}_k}\mathbf{C}_{\mathbf{d}_k|\mathbf{d}_{k-1}}\mathbf{A}'_{\mathbf{d}_k} + \sigma_w^2\mathbf{I}_l)$, where $\mathbf{A}'_{\mathbf{d}_k}$ is the transpose of the matrix $\mathbf{A}_{\mathbf{d}_k}$.

Performing the above two steps for all $\boldsymbol{\tau} \in \mathcal{T}_k$ and computing (3.1) and (3.2) completes the belief state update. This representation of the belief state and its propagation becomes cumbersome due to the growing number of possible links.

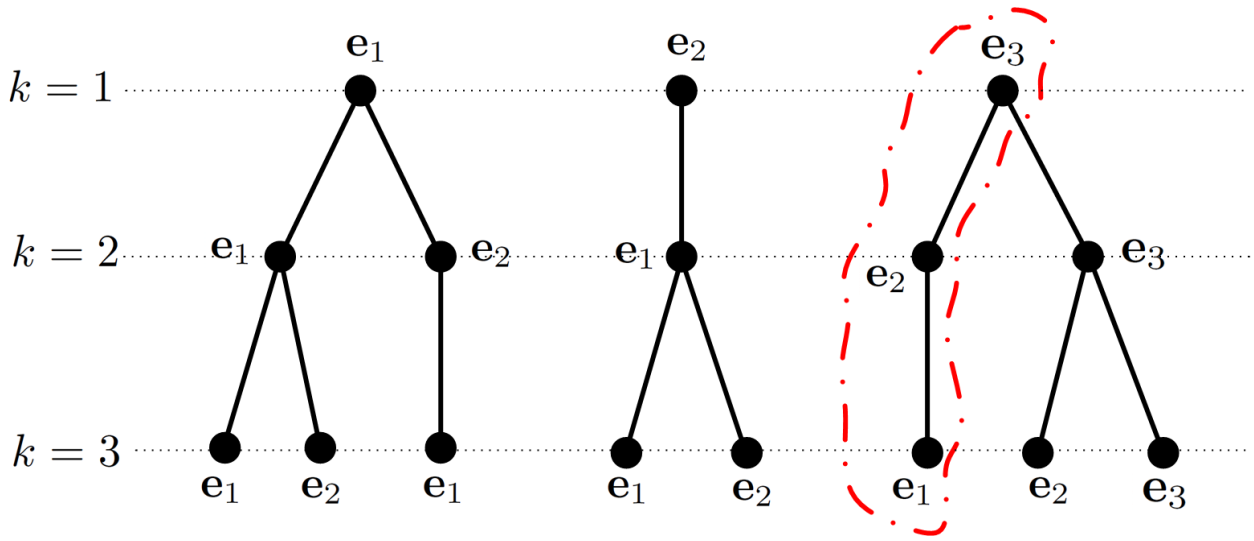


Figure 3.1: Construction of links and routes from three support possibilities $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ over time. This figure also represents the initial stages of the hypothesis tree discussed in Section 3.1.2

3.1.2 Belief State Approximation

As we briefly mentioned, propagating the entire distribution involves an exponential expansion in the number of possible links (and thus routes), quickly becoming unmanageable. To curb this growth-rate, it is reasonable to retain only the most probable routes. We consider this procedure as one of associating the measurements $\mathbf{y}_k$ to the *possible* support-value pairs. The similarity to the data association problem motivated the adaptation of an available solution in the target tracking context, multiple hypothesis tracking (MHT) [38]. We now introduce our belief state approximation method, which combats the expansion of the belief state distribution components.

The essential question to ask is how to choose which distributions and weights should be retained to represent the most viable support-value pairs and associated routes. The following algorithm ranks the routes and determines which routes to discard at each time step.

Recall the notation for route i at time k, $\boldsymbol{\rho}_k^{(i)}$. Let $h_i^{(k)}$ be the hypothesis that the ith route is the true route at time k. If hypothesis $h_i^{(k)}$ is true, then $\mathbf{d}_k = \boldsymbol{\rho}_k^{(i)}(1), \mathbf{d}_{k-1} = \boldsymbol{\rho}_k^{(i)}(2), \ldots, \mathbf{d}_1 = \boldsymbol{\rho}_k^{(i)}(k)$. Also, let $h^{(k)}$ be the random variable taking values on the set of all hypotheses $h_i^{(k)}$, $i = 1, \ldots, |\mathcal{T}_k|$, at time k. Fig. 3.1 can also be viewed as *hypothesis tree*, representing the branching of the possible routes.

Determining the terminal node of the most probable route at time k is done by ranking the possible routes with their *track scores*. Let $\boldsymbol{o}_k$ be a tuple defined as $\boldsymbol{o}_k = (\mathbf{y}_1, \ldots, \mathbf{y}_k)$. Then, given the tuple $\boldsymbol{o}_k = \boldsymbol{o}$, where $\boldsymbol{o} = (\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(k)})$, and the history $H_k = \{\mathbf{u}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{u}^{(k)}, \mathbf{y}^{(k)}\}$ at time k, we define the track score as $S_{\boldsymbol{\rho}_k^{(i)}} = f_{\boldsymbol{o}_k, h^{(k)}}(\boldsymbol{o}, h_i^{(k)})$, where $f_{\boldsymbol{o}_k, h^{(k)}}$ is the joint probability density function of $\boldsymbol{o}_k$ and $h^{(k)}$ at time step k. By expanding the function $f_{\boldsymbol{o}_k, h^{(k)}}$, the value of $S_{\boldsymbol{\rho}_k^{(i)}}$ is equal to

$$
\begin{aligned}
&f_{\mathbf{y}_k|\mathbf{d}_k, \mathbf{d}_{k-1}, \mathbf{u}_k, H_{k-1}}(\boldsymbol{o}(k)|\boldsymbol{\rho}_k^{(i)}(1), \boldsymbol{\rho}_k^{(i)}(2), \mathbf{u}^{(k)}, H_{k-1}) \\
&\quad \times p_{\boldsymbol{\rho}_k^{(i)}(2)\boldsymbol{\rho}_k^{(i)}(1)} P_{\mathbf{d}_{k-1}|\mathbf{d}_{k-2}, H_{k-1}}(\boldsymbol{\rho}_k^{(i)}(2)|\boldsymbol{\rho}_k^{(i)}(3), H_{k-1}).
\end{aligned}
$$

Note that all of the components in this expression can be computed using the equations presented in Section 3.1.1.

Now that we have defined the numerical ranking of the routes, we use a *sliding window* heuristic to eliminate the less feasible ones by *pruning* the hypothesis tree as its depth grows. Consider a sliding window of size w. After the first w time steps, the current node with the maximum track score is selected. The route terminating with the selected node is then traced back w levels in the tree. The node at the base of this branch and all of its descendants are retained, while all other nodes and branches from time $(k - w)$ onward are discarded. At every time step $k \geq w$, a similar pruning of the tree is performed. We direct the reader to [38] for an illustrative example of pruning. The larger the sliding window the better the chances of a correct association due to the increased information accumulated, but the determination of the window size must also consider the additional computations required as the size increases.

3.2 Q-value Approximation: Rollout

After the belief state (or an approximation thereof) is updated, we want to make optimal decisions at each time step by selecting the measurement matrix $\mathbf{A}_k$ given H_k. In this section, we describe our solution approach based on an approximation method called *rollout*. Our description relies heavily on well established ideas and terminology from POMDP theory, which are readily available in [27].

The solution to a POMDP problem is, at each time k, a mapping that takes the history H_k and gives an optimal action from $\mathcal{A}$. However, POMDP theory establishes that it suffices to find an optimal mapping $\pi_k^* : \mathcal{B} \to \mathcal{A}$ for $k = 1, \ldots, m$, where $\mathcal{B}$ is the belief state space and $\mathcal{A}$ is the actions space. If all actions are chosen using these optimal mappings, then at time $k = m$, the predefined objective function is optimized.

We refer to the objective function as the *expected cumulative cost*. Given a policy $\pi = \{\pi_1, \ldots, \pi_m\}$ the expected cumulative cost is defined as $V_m^{\pi}(\mathbf{b}_1) = \mathbf{E}\left[\sum_{k=1}^{m} C_k(\mathbf{b}_k, \pi_k(\mathbf{b}_k))\middle|\, \mathbf{b}_1\right]$. Subsequently, the optimal objective function value $V_m^{\pi^*}$ is achieved when $\pi = \pi^*$, where $\pi^* = \{\pi_1^*, \ldots, \pi_m^*\}$ is the *optimal policy*.

Bellman's principle states that the optimal objective function is characterized by $V_m^{\pi^*}(\mathbf{b}_1) = \min_{\mathbf{u}}(C_1(\mathbf{b}_1, \mathbf{u}) + \mathbf{E}[V_{m-1}^{\pi^*}(\mathbf{b}_2)|\mathbf{b}_1, \mathbf{u}])$, where the optimal policy is $\pi_1^*(\mathbf{b}_1) = \arg\min_{\mathbf{u}}(C_1(\mathbf{b}_1, \mathbf{u}) + \mathbf{E}[V_{m-1}^{\pi^*}(\mathbf{b}_2)|\mathbf{b}_1, \mathbf{u}])$. Letting the *Q-value* of action $\mathbf{u}$ at belief state $\mathbf{b}_k$ be

$$Q_{m-k}(\mathbf{b}_k, \mathbf{u}) = C_k(\mathbf{b}_k, \mathbf{u}) + \mathbf{E}[V_{m-k}^{\pi^*}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}], \tag{3.3}$$

and applying Bellman's principle shows that the optimal action at time k can be viewed as

$$\pi_k^*(\mathbf{b}_k) = \arg\min_{\mathbf{u}} Q_{m-k}(\mathbf{b}_k, \mathbf{u}).$$

The two summands in (3.3), $C_k(\mathbf{b}_k, \mathbf{u})$ and $\mathbf{E}[V_{m-k}^{\pi^*}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}]$, are the *immediate cost* incurred (by $\mathbf{u}$) and the *expected cost-to-go* (ECTG), respectively. A *myopic* action selection only considers the immediate cost of an action, in effect setting the ECTG to zero. In contrast, non-

myopic methods attempt to approximate the ECTG. Rollout is one of the several approaches for Q-value approximation [40].

Rollout replaces the optimal policy used in the ECTG with a so called *base policy* π^{b}, creating the Q-value approximation: $\tilde{Q}_{m-k}(\mathbf{b}_k, \mathbf{u}) = C_k(\mathbf{b}_k, \mathbf{u}) + \mathbf{E}[V^{\pi^{\text{b}}}_{m-k}(\mathbf{b}_{k+1})|\mathbf{b}_k, \mathbf{u}]$. This approximation differentiates the effects of the actions not only at the current time step but over the future horizon, providing *multi-step lookahead* as opposed to the *1-step lookahead* of the myopic policy. The simulation results presented in the next section produced by rollout also exploit *receding horizon control*, replacing the remaining horizon $m - k$ with a fixed value ϑ. This fixed *rollout horizon* defines a ϑ-lookahead policy.

Further reading on receding horizon control and rollout can be found in [40], but provide a result which motivates the use of rollout. By ranking actions over time with their approximate Q-values, rollout produces a policy $\tilde{\pi} = \{\tilde{\pi}_1, \tilde{\pi}_2, \ldots, \tilde{\pi}_m\}$, where $\tilde{\pi}_k(\mathbf{b}_k) = \arg\min_{\mathbf{u}} \tilde{Q}_\vartheta(\mathbf{b}_k, \mathbf{u})$. The policy $\tilde{\pi}$ is guaranteed to perform at least as well as the base policy π^{b} with respect to the objective function.

In our simulations, rollout and receding horizon control are combined with an additional heuristic, derived from the belief state approximation method from Section 3.1.2. This heuristic resembles *completely observable rollout* [40]; we endeavor to reduce the computations in the simulations of the future via the reduction of the (simulated) belief state representations. Our approximation is less drastic than that of completely observable rollout, but it still takes advantage of the knowledge of the underlying system state during the Monte Carlo simulations used for the Q-value approximations.

A majority of the computational energy in the belief state approximation is spent expanding the belief state to represent all possible routes (prediction) and then immediately eliminating several of the less probable routes (pruning) to form the current belief state estimate. Our belief state reduction, inside rollout, eliminates the computations incurred by this cycle of prediction and pruning. The tree is extended with only the nodes representing the best case, where the selected node is no longer necessarily the most probable node but actually represents the "true" support-value pair

descending from the "true" route of the sample signal. This approximation uses far fewer Kalman filters for prediction and completely eliminates pruning during the Q-value approximations. The empirical results in the next section show that, in the problem of interest, this approximation is viable.

3.3 Simulations

We consider a 1-sparse dynamic signal in $\mathbb{R}^{20}$. We assume a uniform prior for $P_{\mathbf{d}_0}$. When the signal support corresponds to erasure index, the received observations are noise samples, $\mathbf{y}_k = \mathbf{w}_k$. The erasure indices for the simulation are 5, 6, 11, and 12. The support transitions are modeled by the transition law shown in Fig. 3.2. The initial signal support is index two. The support index increases by one at each time step until reaching index 20. The support then decreases by one until reaching index 1, where it begins the increasing pattern again. The simulation horizon is 40 time steps, so the signal will experience erasure at times $k = \{4, 5, 10, 11, 27, 28, 33, 34\}$. We set the sliding window size to $w = 4$ and SNR $= 12$ dB. The experiment is run 36 times for the three algorithms.

Our action space $\mathcal{A}$ is comprised of 300 (6×20) matrices. Each row of a matrix is a vector of a Grassmanian packing [44] of 16 vectors in $\mathbb{R}^{16}$. The use of this vector library is inspired by the property of the Grassmannian packing; every pair of waveforms in the library achieves a maximum angular distance from one another. Therefore, by using these waveforms as rows of our measurement matrices, we are searching for the sparse signal in a subspace of $\mathbb{R}^{20}$ in a semi-uniform manner. To form the matrices, we extend each vector to $\mathbb{R}^{20}$ by inserting zeros at the erasure indices. No matrix in the entire action space contains repeated rows and the sequentially restricted action spaces $\mathcal{A}_{\mathcal{G}}{}^{(k)}$, described in Section 3.1, are respected by each of following three solution methods.

Bhattacharyya Distance: This adaptive heuristic is inspired by [19] where the measurement should resemble the prior. This policy requires offline computations of a power-vector corresponding to each matrix in the library. The power-vector $\boldsymbol{\phi}_{\mathbf{A}}$ of a matrix $\mathbf{A}$ is a (20×1) normalized

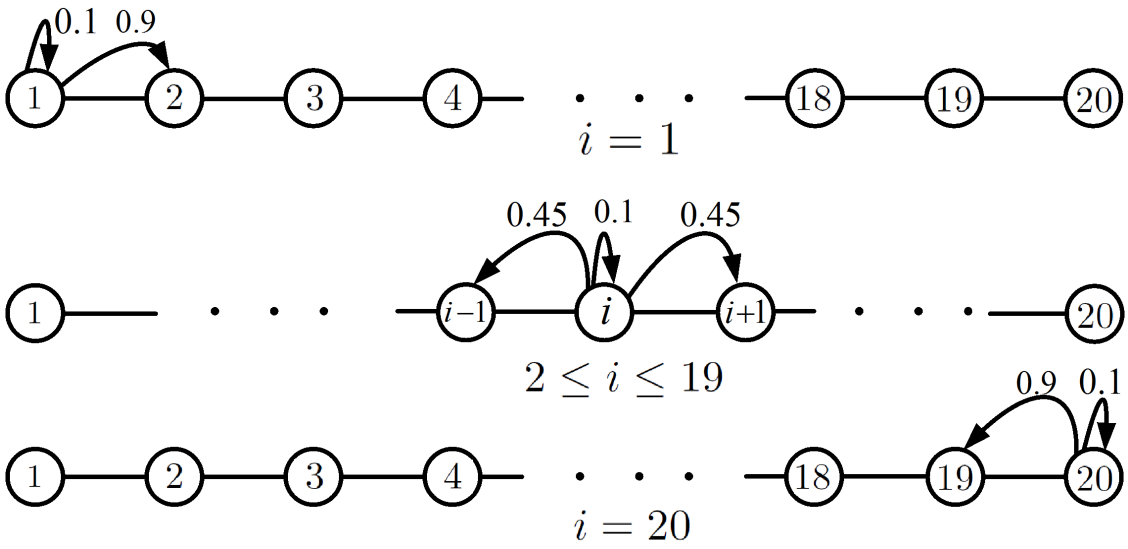


Figure 3.2: State transition law for the dynamic signal support.

vector where its jth component is the sum of absolute values of the entries in the jth column of $\mathbf{A}$. These power-vectors portray the distribution of power over the columns of the matrix $\mathbf{A}$, resembling probability mass functions. The selected measurement matrix $\mathbf{A}_k \in \mathcal{A}_{\mathcal{G}}{}^{(k)}$ attains the minimum Bhattacharyya distance from its power-vector to the predicted support distribution.

Greedy: At each time step k, Greedy selects the matrix from the restricted action space $\mathcal{A}_{\mathcal{G}}{}^{(k)}$ that minimizes the immediate cost.

Rollout: This solution method selects the matrix $\mathbf{A}_k \in \mathcal{A}_{\mathcal{G}}{}^{(k)}$ that minimizes the 2-step lookahead Q-value approximation, using the Bhattacharyya Distance as the base policy.

Fig. 3.3 shows the performance of each method over 40 time steps. As expected, there is a decrease in performance during erasure indices and where the support transitions reverse direction. Rollout out performs both Greedy and the Bhattacharyya Distance heuristic due to its 2-step lookahead. Approximating the ECTG allows Rollout to account for the future action restrictions (long-term effects) induced by its current action selection. The other two policies do not have the advantage of foresight.

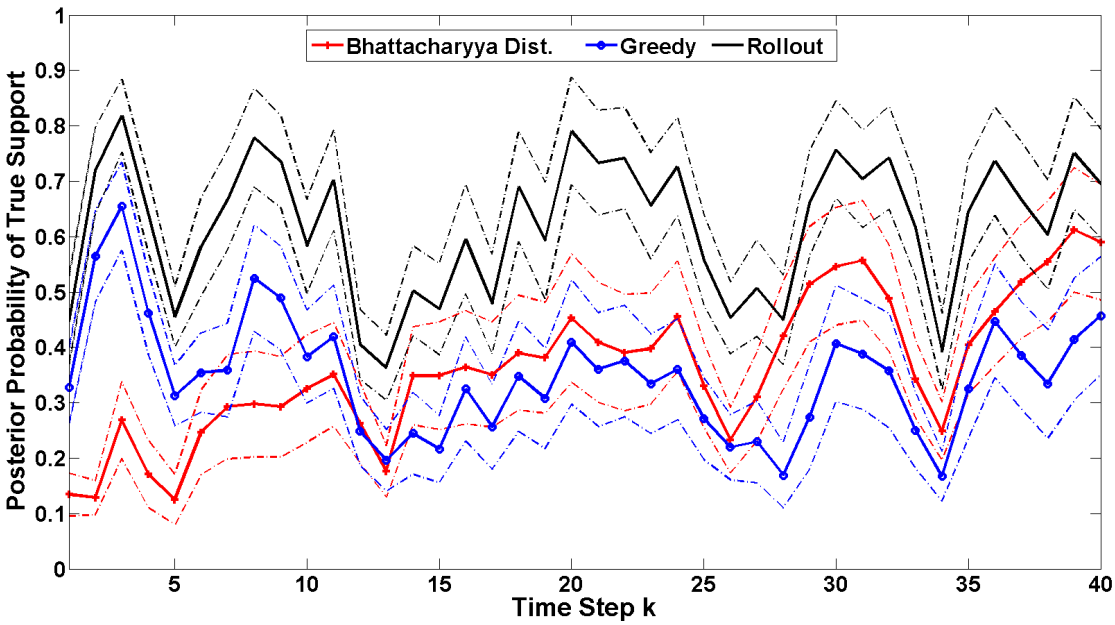


Figure 3.3: Performance comparison of the three policies, Bhattacharyya Distance, Greedy and Rollout. The dashed lines indicate the 85% confidence intervals.

3.4 Conclusion

In this chapter, we have presented a POMDP framework for adaptive measurement selection in compressed sensing using POMDP theory. We introduced a belief state approximation method based on MHT and applied a multi-step lookahead solution, rollout. Our simulations exemplified a scenario including signal erasure and a covertness restriction on the action space. The results show the advantages of a lookahead policy under these constraints.

Chapter 4

Autonomous UAV Control: Balancing Target Tracking and Persistent Surveillance

Persistent surveillance and target tracking are two naturally related tasks. Both tasks are of interest in the area of UAV control; however, it is common that each is considered separately. This is often due to their independent and sometimes conflicting objectives.

Persistent surveillance is focused on aspects related to the monitoring of a broad area. Common objectives are related to maximizing area coverage, minimizing target-detection latency and minimizing the time between re-visits (e.g., blanket time, refresh time). Several approaches to this problem exist, from static pre-computed search/patrolling patterns to dynamic control [47].

The target tracking objective is well known; commonly, we minimize the mean-squared error between the estimated and true target location over time. It is clear that this objective concentrates the observational efforts in the immediate vicinity of estimated targets.

These differing objectives lead to a partition in the research. When considering the joint tasks of surveillance and target tracking, this divide can be bridged. One possibility is to implement a finite-state machine which deterministically assigns individual UAVs a task of either surveillance or target tracking based on the system state. An example of such an implementation can be found in [48].

We propose a novel approach to cooperative UAV control for the combined tracking and surveillance objective. In contrast to controlling the modes of individual UAVs, we formulate the problem as a partially observable Markov decision process (POMDP) [49] allowing the application of a *lookahead* control approach called *nominal belief-state optimization* (NBO) [50]. Accounting for the future ramifications of current control choices aids in unifying the two seemingly different objectives.

4.1 Problem Description

Our problem is of the search, detect, and track nature. A fleet of UAVs, each equipped with a sensor, is employed to monitor a pre-defined surveillance region. Upon initial target detection, an updated estimate of the target's location must be continuously maintained. While tracking the known targets, the search process must continue to prompt the detection of possible new targets appearing in the region.

While surveillance and target tracking frame the objectives of the scenario the true focus is the autonomous UAV control, optimizing the sensor placements over time. This falls under the overarching domain of *sensor resource management* [51]. This domain is often bisected by solution approaches of either myopic or non-myopic.

Myopic solution approaches select actions based explicitly on the minimization of *immediate* cost (e.g., [52]), while non-myopic approaches employ the concept of *lookahead*, selecting actions based on the cost incurred both presently and over the future of the control process. A prime example in recent literature employing non-myopic sensor control is [53]. Our investigation is highly related to both [52] and [53] but provides an unique approach implicitly encouraging UAV movements benefiting area surveillance while tracking targets. This benefit is only realized through non-myopic action selection.

With the general problem statement in mind, we point out the prominent components making the scenario interesting in the target tracking, surveillance, and controls paradigms. Several of the following are modified versions taken from past literature [50, 54].

Randomness The target motion is modeled as a random process and the observations have random errors.

Observations The observations collected from the UAV mounted sensors can be corrupted in three ways: 1) False alarms, 2) missed detection of true targets, 3) spatially varying random measurement error.

Constrained Control The UAV motion is determined by forward acceleration and bank angle control inputs, both of which have minimum and maximum bounds. This restricts the acuteness of the deviations in the UAV flight paths, emulating the physical constraints of a fixed-wing aircraft.

Objective The objective is two-fold: minimize the mean-squared tracking error while continuously surveilling the area of interest.

Each of these elements influences the problem formulation within the POMDP framework, Section 4.2, and is exemplified by results presented in Section 4.4.

4.2 Problem Formulation

The following describes the POMDP components similar to [50] and [54] based on the similarities of the problem description from the previous section.

The specification of the POMDP follows:

State

The POMDP state is comprised of three components. The sensor state S_k describes the UAV positions and velocities. The target state χ_k contains the positions and velocities of all targets present at time k. The filter state $\mathcal{F}_k = (w_k, \xi_k, \mathrm{P}_k)$ is a representation of a Gaussian mixture where w_k, ξ_k, and P_k are the Gaussian mixture weights, posterior means and posterior covariance matrices, respectively. The filter state is supplied by a GM-PHD filter, describe in Section 4.3.1. The POMDP state is then defined as $X_k = (S_k, \chi_k, \mathcal{F}_k)$.

Actions

In the surveillance and tracking scenario the actions are the UAV control input vectors for forward acceleration $a_k = \left[a_k^{(1)}, \dots a_k^{(N_s)}\right]^\top$ and bank angle $\theta_k = \left[\theta_k^{(1)}, \dots \theta_k^{(N_s)}\right]^\top$ forming $u_k = (a_k, \theta_k)$. Both components of the action are constrained such that $a_k^{(i)} \in [-a^*, a^*]$ and $\theta_k^{(i)} \in [-\theta^*, \theta^*]$.

Observation

Since both the sensor state and the filter state are fully observable their state components are their observations. The target state is only accessible through the observations $\mathrm{Z}_k = \{z_k^1, \ldots, z_k^{N_m}\}$ where N_m is the total number of measurements and $z_k^i = [x_i \; y_i]^\top$ is two dimensional position vector.

Observation Law

Each measurement z_k^i is either generated as a result of a true target or from clutter The cardinality of the set of clutter measurements in the observation is Poisson distributed with rate parameter β and uniformly spatially distributed over the surveillance volume of the sensor. At time k, a sensor with position s_k^{pos} measurement originates from a target at location ζ_k^{pos} with probability,

$$p_d\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right) = \begin{cases} \hat{p}_d, & \|s_k - \chi_k\| \leq r \\ 0, & \text{otherwise} \end{cases}, \tag{4.1}$$

of the form $z_k^i = \chi_k^{\text{pos}} + \nu_k^{(z)}$ such that $\nu_k^{(z)} \sim \mathcal{N}\left(0, R\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right)\right)$. Equation (4.1) effectively defines the circular field of view of the sensor with radius r, where $\|\cdot\|$ denotes the Euclidean distance. The covariance matrix $R\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right)$ defines a 10% uncertainty in the range from sensor to target with a minimum range of 10 meters, and 0.01π radian angular uncertainty.

State Transition Law

Define $X_{k+1} = f\left(X_k | u_k\right)$ as the state transition law mapping the state at time k to the new state at time $k+1$. The mapping $f(\cdot|u_k)$ can be decomposed into mappings for each of the POMDP state components. The sensor state transition $S_{k+1} = f^s\left(S_k, u_k\right)$ is the kinematic motion model of the UAV given the action u_k, as defined in [54]. The target state transition is $\chi_{k+1} = f^t\left(\chi_k, v_k\right)$. Assuming the movement among targets is independent then it is sufficient to define the single target motion model [55]

$$\chi_{k+1} = F_k\, \chi_k + \nu_k^{(\chi)},\ \nu_k^{(\chi)} \sim \mathcal{N}\left(0, Q_k\right), \tag{4.2}$$

where,

$$F_k = \begin{bmatrix} 1 & 0 & \Delta_k & 0 \\ 0 & 1 & 0 & \Delta_k \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q_k = \sigma_{\mathrm{v}}^2 \begin{bmatrix} \frac{\Delta_k^4}{4} & 0 & \frac{\Delta_k^3}{2} & 0 \\ 0 & \frac{\Delta_k^4}{4} & 0 & \frac{\Delta_k^3}{2} \\ \frac{\Delta_k^3}{2} & 0 & \Delta_k^2 & 0 \\ 0 & \frac{\Delta_k^3}{2} & 0 & \Delta_k^2 \end{bmatrix}$$

provided Δ_k is the length of time-step k in seconds and σ_{v} is the standard deviation of the process noise in meters per second. Finally, the filter state $\mathcal{F}_{k+1}$ evolves according to the GM-PHD filter, discussed in Section 4.3.1.

One-step Cost

The one-step cost function considered is the mean squared tracking error defined as [54]

$$c\left(X_k, u_k\right) = \mathrm{E}_{\nu_k^{(\chi)}, \nu_k^{(z)}}\left[||\chi_{k+1} - \xi_{k+1}||^2 \,|X_k, u_k\right]. \tag{4.3}$$

Although this cost does not directly include a term to account for the persistent surveillance, Section 4.3.2 provides the details compelling surveillance with this cost alone.

4.3 POMDP Solution

An MDP has perfect knowledge of the system state while a portion of the information about the underlying state is concealed in a POMDP, accessible only through the observations. Applying the MDP solution theory to a POMDP requires the definition of a *belief state*; the belief state b_k at time k is the posterior distribution of the underlying state conditioned on the history of actions and observations, $b_k(X) = P_{X_k}(X|\mathrm{Z}_0, ..., \mathrm{Z}_k; u_0, ..., u_{k-1})$. A corresponding transition law, termed the *belief-state update*, is characterized by the state transition law and the observation law, discussed further in Section 4.3.1 Additionally, given the one-step cost $c\left(X_k, u_k\right)$, the *belief cost* is

$$C(b_k, u_k) = \int c(X, u_k) b_k(X) dX, \tag{4.4}$$

where u_k is the POMDP action at time k. The above constitutes a belief MDP [28], allowing the application of MDP solution theory, examined in Section 4.3.2.

4.3.1 Approximate Belief-State Update

The propagation of the posterior distribution is accomplished through Bayesian filtering in two steps: prediction and update. Barring any assumptions on the structure of the belief state, this computation is often intractable.

The target tracking paradigm supplies a wide variety approximation techniques which have proven effective in practical application. We chose to adopt the GM-PHD filter and its corresponding assumptions about the structure of the posterior distribution of multiple targets (i.e., the belief state) [55]. The GM-PHD filter prescribes a closed form Bayesian update of a belief-state approximation with the intensity (of the belief state) represented as a Gaussian mixture.

The following description of the GM-PHD is adapted from [56] and [57]. For brevity we only mathematically describe the recursive prediction and update steps for the GM-PHD filter and briefly note the pruning and merging process used to alleviate the exponential growth of the number of Gaussian mixture components over time. A comprehensive description of the GM-PHD filter can be found in [55].

Initialization At time $k = 0$, the intensity $\tilde{b}_{k|k}$ is initialized as the Gaussina mixture

$$\tilde{b}_0(\mathrm{x}) = \sum_{j=1}^{J_k} w_0^{(j)} \mathcal{N}(\mathrm{x}; \xi_0^{(j)}, \mathrm{P}_0^{(j)}),$$

where each Gaussian term $\mathcal{N}(\mathrm{x}; \xi_o^{(j)}, \mathrm{P}_0(j))$ has a corresponding weight $w_0^{(j)}$.

Prediction The predicted intensity up to time k is a Gaussian mixture, $\tilde{b}_{k|k-1}(\mathrm{x}) = \tilde{b}_{S,k|k-1}(\mathrm{x}) + \gamma_k(\mathrm{x})$ where, $\tilde{b}_{S,k|k-1}(\mathrm{x})$ is predicted intensity of the existing targets in the surveillance area given by,

$$\tilde{b}_{S,k|k-1}(\mathrm{x}) = p_S \sum_{j=1}^{J_{k-1}} w_{k-1}^{(j)} \mathcal{N}(\mathrm{x}; \xi_{S,k|k-1}^{(j)}, \mathrm{P}_{S,k|k-1}^{(j)})$$

with,

$$\begin{aligned} w_{k|k-1}^{(j)} &= p_S \cdot w_{k-1}^{(j)}; \\ \xi_{S,k|k-1}^{(j)} &= F_{k-1} \xi_{k-1}^{(j)}; \\ \mathrm{P}_{S,k|k-1}^{(j)} &= Q_{k-1} + F_{k-1} \mathrm{P}_{k-1}^{(j)} F_{k-1}^T \end{aligned}$$

and p_S is the probability that a target "survives" from time $k-1$ to k.

$\gamma_k(\mathrm{x})$ is the *birth intensity* representing the *possible* new targets entering the surveillance area,

$$\gamma_k(\mathrm{x}) = \sum_{j=1}^{J_{\gamma,k}} w_{\gamma,k}^{(j)} \mathcal{N}(\mathrm{x}; \xi_{\gamma,k}^{(j)}, \mathrm{P}_{\gamma,k}^{(j)}), \tag{4.5}$$

where $J_{\gamma,k}$ denotes the number of Gaussian mixture components and $w_{\gamma,k}^{(j)}$, $\xi_{\gamma,k}^{(j)}$, $\mathrm{P}_{\gamma,k}^{(j)}$, $j = 1, \ldots, J_{\gamma,k}$ are model parameters defining the Gaussian mixture birth intensity.

Update The measurement update is

$$\tilde{b}_{k|k}(\mathrm{x}) = \tilde{b}_{k|k}(\mathrm{x}; s_k^{\text{pos}}) + \sum_{z \in \mathrm{Z}_k} \tilde{b}_{L,k}(\mathrm{x}; z)$$

where,

$$\tilde{b}_{k|k}(\mathrm{x}; s_k^{\text{pos}}) = \sum_{j=1}^{J_{k-1}} \left(1 - p_d(s_k^{\text{pos}}, \xi_{S,k|k-1}^{(j)})\right) \times w_{k-1}^{(j)} \mathcal{N}(\mathrm{x}; \xi_{S,k|k-1}^{(j)}, \mathrm{P}_{S,k|k-1}^{(j)})$$

with s_k^{pos} as the two-dimensional position of the sensor corresponding to Z_k and $p_d(s_k^{\text{pos}}, \xi_{S,k|k-1}^{(j)})$ is an approximation of the true probability of detection [58]. Furthermore,

$$\tilde{b}_{L,k}(\mathrm{x}; z) = \sum_{j=1}^{J_{k-1}+J_{\gamma,k}} w_{k|k}^{(j)} \mathcal{N}(\mathrm{x}; \xi_{k|k}^{(j)}, \mathrm{P}_{k|k}^{(j)})$$

with,

$$w_{k|k}^{(j)} = \frac{p_d(s_k^{\text{pos}}, \xi_{S,k|k-1}^{(j)}) w_{k|k-1}^{(j)} f_k^{(j)}(z|\mathrm{x})}{\beta c_k(z) + \sum_{l=1}^{J_{k-1}+J_{\gamma,k}} w_{k|k-1}^{(l)} f_k^{(l)}(z|\mathrm{x})}$$

$$f_k^{(j)}(z|\mathrm{x}) = \mathcal{N}(z; H_k \xi_{k|k-1}^{(j)}, S_k^{(i)});$$

$$\xi_{k|k}^{(j)} = \xi_{k|k-1}^{(j)} + K_k^{(j)}[z - H_k \xi_{k|k-1}^{(j)}];$$

$$\mathrm{P}_{k|k}^{(j)} = [I - K_k^{(j)} H_k^{(j)}] \mathrm{P}_{k|k-1}^{(j)};$$

$$K_k^{(j)} = \mathrm{P}_{k|k-1}^{(j)} [H_k^{(j)}]^T [S_k^{(j)}]^{-1};$$

$$S_k^{(j)} = R_k + H_k^{(j)} \mathrm{P}_{k|k-1}^{(j)} [H_k^{(j)}]^\top,$$

and $\beta c_k(z)$ represents the false alarms.

Thus, $J_k = (1 + |\mathrm{Z}_k|)(J_{k-1} + J_{\gamma,k})$ Gaussians are updated with $(1 + |\mathrm{Z}_k|)$ components for each prediction term at time k and the Gaussian mixture is of the form,

$$\tilde{b}_{k|k}(\mathrm{x}) = \sum_{j=1}^{J_k} w_{k|k}^{(j)} \mathcal{N}(\mathrm{x}; \xi_k^{(j)}, \mathrm{P}_k^{(j)}).$$

Pruning & Merging Pruning eliminates components of the Gaussian mixture whose weights fall below a threshold, τ_p. This removal includes a re-weighting of the remaining Gaussian mixture components by distributing the weights of the discarded components proportionally. Merging combines similar Gaussian mixture components whose statistical distance is less than the merge threshold, τ_m. After these processes are complete, the resulting density represents $\tilde{b}_{k|k}(\mathrm{x})$.

Target State Estimation The target state estimates are obtained by selecting the Gaussian means whose weights are above the threshold, τ_w. The set of target state estimates at time k is $\hat{T}_k = \left\{ \left(\xi_k^{(i)}, \mathrm{P}_k^{(i)} \right) : w_k^{(i)} \geq \tau_w \right\}$.

4.3.2 Solution Method

Section 4.2 provides a complete POMDP formulation and Section 4.3.1 explains the the approximation of the belief state update through the use of the GM-PHD filter. With these concepts in place we now describe the approximate solution method *nominal belief state optimization*.

An optimal *policy* which maps the belief state to action for a POMDP over a horizon H minimizes the cumulative belief cost [59],

$$V_H(b_0) = E\left[\sum_{k=0}^{H-1} C\left(b_k, u_k\right) \mid b_0 \right], \tag{4.6}$$

given the initial belief state b_0. Applying *Bellman's principle* to V_H yields the description of the optimal objective function value

$$V_H^*\left(b_0\right) = \min_u \left\{ C\left(b_0, u\right) + \mathrm{E}\left[V_{H-1}^*\left(b_1\right) | b_0, u \right] \right\}.$$

where b_1 is the random next belief state whose distribution depends on action u and V_{H-1}^* is the optimal cumulative cost over the horizon $H-1$. Defining the Q-value as $Q_H(b_0, u) = C(b_0, u) + \mathrm{E}\left[V_{H-1}^*\left(b_1\right) \mid b_0, u \right]$, allows Bellman's principle can be rewritten as $V_H^*\left(b_0\right) = \min_u Q_H(b_0, u)$. Then the optimal policy at any time k is

$$\pi_k^*\left(b_k\right) = \arg\min_u Q_H\left(b_k, u\right).$$

In general, computing true values for the expectation in in the Q-value is difficult. Thus, a viable approximation of the Q-value for applications is desired. We chose the NBO approximation from [50] which performed well in similar tracking scenarios.

NBO proposes the replacement of the objective function in (4.6) with the sum

$$V_H\left(b_0\right) \approx \sum_{k=0}^{H-1} C\left(\hat{b}_k, u_k\right), \tag{4.7}$$

where $\{\hat{b}_i\}_{k=0}^{H-1}$ is the nominal belief state progression. The minimization is performed over the action sequence $\{u_i\}_{k=0}^{H-1}$.

Analogous to [50] and [54] we make linear Gaussian system assumptions representing the target motion and observation models. Then, given a target estimate described by a Gaussian distribution with mean ξ_k and covariance matrix P_k at time k, the nominal belief state progression can be recursively computed with the standard Kalman filter equations with zero noise sequence.Under these assumptions [54] shows the NBO tracking error objective function (based on the one-step cost defined in (4.3)) can be written as

$$V_H\left(b_0\right) = \sum_{k=0}^{H-1} \sum_{i=1}^{N_T} \mathrm{Tr}\, \mathrm{P}_{k+1}^{(i)},$$

where N_T is the number of targets in the nominal belief state and Tr is the matrix trace function. When there are N_s UAV's then $\mathrm{P}_{k+1}^{(i)} = \left[\sum_{j=1}^{N_s} \left(\mathrm{P}_{k+1}^{(i,j)}\right)^{-1}\right]^{-1}$.

Recall, the GM-PHD filter approximates the belief state update, providing a Gaussian mixture as the representation of the posterior distribution, fully characterize by $\mathcal{F}_k = (w_k, \xi_k, \mathrm{P}_k)$. The question remains, which Gaussian components from the filter state should be used in the NBO procedure? The first obvious choice would be to use all of the Gaussians in $\mathcal{F}_k$. This choice becomes less appealing when considering the computational load, running $(1 + |\mathrm{Z}_k|)(J_{k-1} + J_{\gamma,k})$ filters for H time steps. Another choice could be the set of Gaussians represented by GM-PHD target estimates $\hat{T}_k$. This vastly reduces the number of Kalman filters required to approximately the number of true targets. If target tracking was the sole concern this could be a valid choice.

Our joint goal is target tracking and persistent surveillance. The GM-PHD filter supplies us with sufficient information to inspire action selections which result in both. Specifically, we choose the set of Gaussians from $\mathcal{F}_k$ represented by the union of estimated tracks and the those pertaining

to the birth intensities. Define the set of birth intensity means as $\mathcal{B} = \left\{\xi_{k,\gamma}^{(i)}\right\}_{i=1}^{J_{\gamma,k}}$ and the distance $\tilde{D}_{\mathcal{B}}^2(\mu,\Sigma) = \min_{x\in\mathcal{B}} D(\mu,\Sigma,x)$ where $D^2(\mu,\Sigma,x)$ is the Mahalanobis distance between the Gaussian distribution $\mathcal{N}(\mu,\Sigma)$ and the point x. Define the set *pertaining* to the birth intensities as

$$\hat{\mathrm{B}}_k = \left\{\left(\xi_k^{(i)}, \mathrm{P}_k^{(i)}\right) \in \mathcal{F} : D_{\mathcal{B}}^2\left(\xi_k^{(i)}, \mathrm{P}_k^{(i)}\right) \leq \tau_N\right\}.$$

Then the set of Gaussian components upon which the NBO approximation is based is $\hat{\mathrm{T}}_k \cup \hat{\mathrm{B}}_k$ at time k. It is specifically this set combined with the lookahead of NBO that prompts the exploration of the surveillance area while jointly tracking present targets.

4.4 Experiments

Our numerical investigation consists of statistics calculated on 500 Monte Carlo simulations. Each simulation contains three mobile targets entering the surveillance area at different times during the 50 second simulation duration. Their starting locations and times are detailed in Table 4.1. We employ two UAVs from the initialization of the scenario, both starting at position $(200, 1000)$. The UAV motion is constrained by a maximum velocity of 26 m/s, a bank angle constraint of $\theta^* = \pi/12$ radians and a lateral acceleration constraint of $a^* = 3.5$ m/s^2. The sensor FOV range is limited to 350 meters in all directions. The physical layout of the scenario can be viewed in Figure 4.1.

Table 4.1: Target Details

Target No.	Start Pos.	Start Time	Avg. Vel.
1	$(200, 1200)$	1	8 m/s
2	$(1200, 200)$	5	6 m/s
3	$(200, 200)$	15	12 m/s

Our scenario makes a few simplifying assumptions. Our targets originate from the birth intensities shown in Figure 4.1. These five birth intensities are immobile over the scenario. For simulation

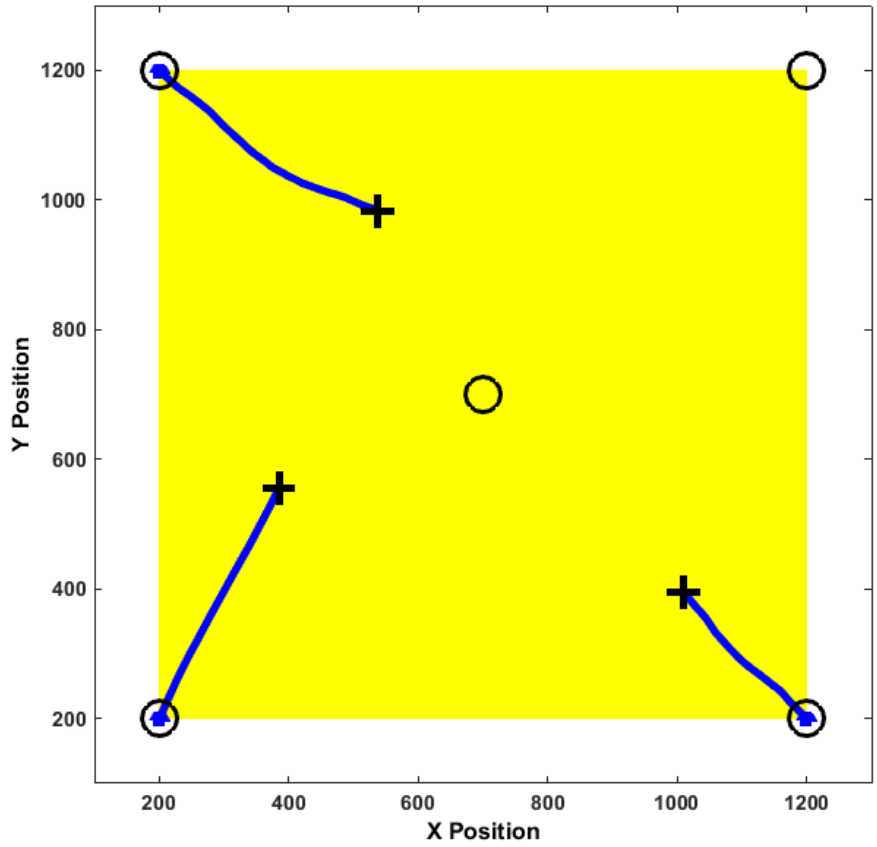


Figure 4.1: The shaded square represents the surveillance area, ∘ are the birth intensities and where the respective targets originate. + are the target ending positions, while the target trajectories are the solid lines.

purposes and concise understandable results, the target trajectories are the same over all simulations. These three trajectories are randomly generated, unknown to the UAVs, by a nearly-constant velocity model.

4.4.1 OSPA

The results of the Monte Carlo simulations are visualized by calculating the Optimal SubPattern Assignment Metric (OSPA) [60]. This metric provides an intuitive interpretation of miss-distance error in the multi-object assignment problem. The basic GM-PHD filter does not make a literal assignment between true targets and the estimated target positions. For this reason the OSPA is the most informative performance metric given this application.

OSPA is a distance metric between two subsets $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$ by accounting for both the difference in cardinality and the spacial distance given the optimal pair-wise assignment of m points of X to the points in Y, assuming $m \leq n$ without loss of generality.

Let the distance $d_2(y, x)$ be the two-dimensional Euclidean distance. The *cut-off distance* is $\tilde{d}(y, x) = \min(d_2(y, x), \epsilon)$ where ϵ is the *cut-off parameter*. Then the OSPA metric is defined as

$$\bar{D}(Y, X) = \bar{e}_L(Y, X) + \bar{e}_{\mathcal{C}}(Y, X)$$

where, $\bar{e}_{\mathcal{L}}(Y, X) = \left(\frac{1}{n} \cdot \min_{\pi \in \Pi_n} \sum_{i=1}^{m} \tilde{d}\left(y_i, x_{\pi(i)}\right)^p\right)^{1/p}$ is the location component, $\bar{e}_{\mathcal{C}}(Y, X) = \left(\frac{\epsilon^p (n-m)}{n}\right)^{1/p}$ is the cardinality component and Π_n is the set of all permutations on $\{1, 2, \ldots, n\}$. The choice of parameter ϵ determines the maximum location penalty *and* the distance given to nonassignable points in X, while the parameter p is chosen to determine the emphasis on either cardinality or location error. For the further discussion on the parameters ϵ and p see [60].

Applying this metric to the simulation results is done on a per time-step basis. At time k we have the ground truth of targets present in the surveillance region Y_k and the predicted set of target locations provided by the GM-PHD filter, $\bar{X}_k$. Our parameter choices for the OSPA parameters are $\epsilon = 150$ and $p = 1$.

4.4.2 Results

We present results for two sets of Monte Carlo simulations. Both simulation sets follow the experiment definitions, differing only in lookahead horizon H from (4.7). We compare myopic, $H = 1$, and non-myopic, $H = 8$, NBO solutions. The results show the necessity of a non-myopic policy in producing a balance between target tracking and persistent surveillance. The OSPA metrics are shown in Figure 4.2.

The OSPA (top) plot in Figure 4.2 show that the non-myopic NBO $(H = 8)$ performs better across the scenario, except perhaps in the first five seconds. Examining the two component plots, OSPA location $\bar{e}_{\mathcal{L}}$ and OSPA cardinality $\bar{e}_{\mathcal{L}}$, can provide further insight. It may appear that the $H = 1$ horizon performs better in the location considerations, but this primarily due to the fact

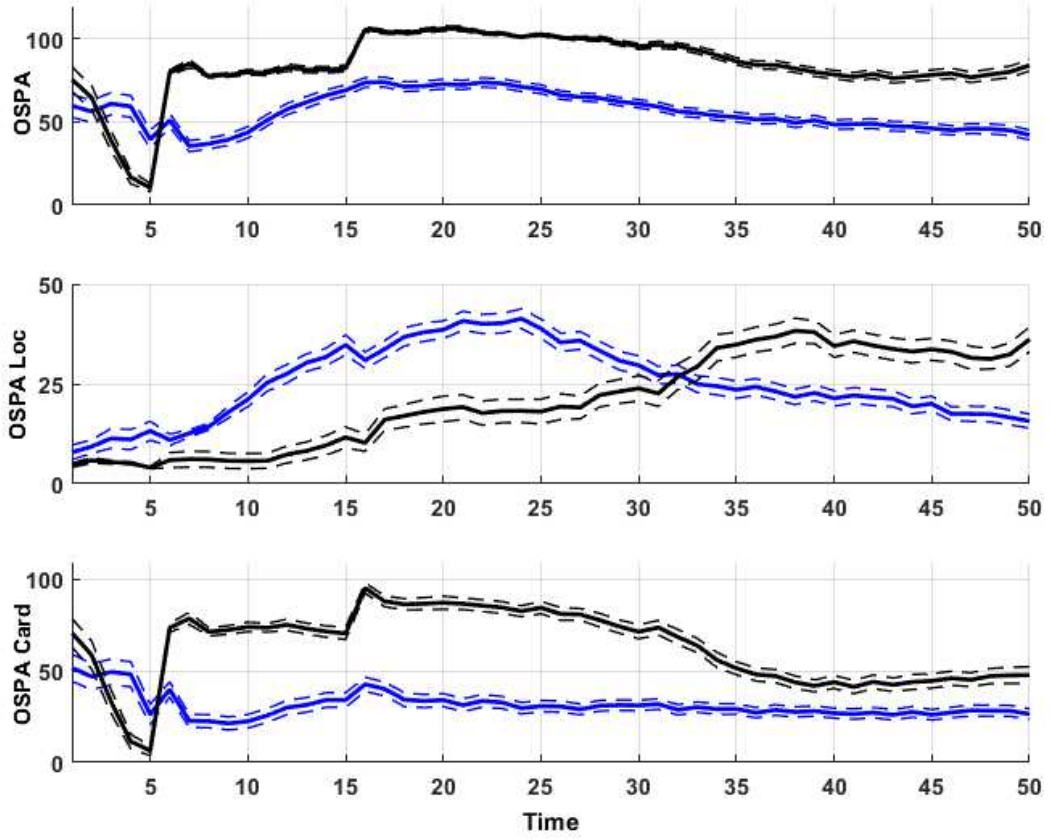


Figure 4.2: The OSPA plot (top) displays the cumulative OSPA metric, $\bar{D}$, the OSPA location (middle) is $\bar{e}_{\mathcal{L}}$, and OSPA cardinality (bottom) is $\bar{e}_{\mathcal{C}}$. In each plot the black solid lines represent the performance of horizon $H = 1$ and the blue solid lines represent horizon $H = 8$. The dashed lines are the 95% confidence intervals.

that fewer targets are detected than are actually present in the surveillance area. This explains the elevated cardinality component values. In juxtaposition, the $H = 8$ horizon has a lower cardinality component showing all targets are more frequently detected. Thus, the location component for the non-myopic NBO includes the cumulative error of the positions of all targets detected at each time-step. There is a rise in the horizon $H = 8$ OSPA location values over the first 20 seconds, the time in which all targets enter the surveillance area. Figure 4.3 displays the true number of targets present in the surveillance area versus the expected cardinality of the GM-PHD estimates, further solidifying the above discussion.

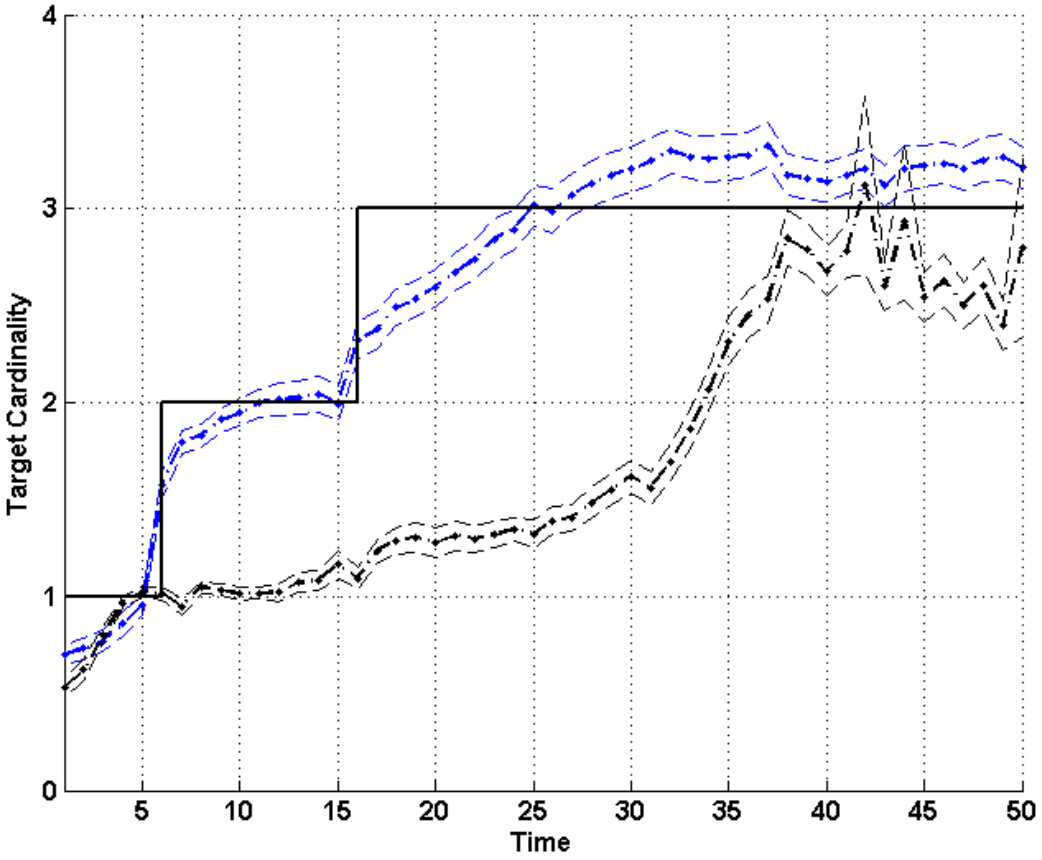


Figure 4.3: True cardinality. The true number of targets present (solid line) compared to the GM-PHD target cardinality (dots) with 95% confidence bounds (dashed). The blue plots represent horizon $H = 8$ and black represent horizon $H = 1$.

Even with the OSPA illustration in Figure 4.2, it may not be obvious that the persistent surveillance objective is sufficiently achieved. The quality of persistent surveillance can be characterized by *target detection latency*. Target detection latency refers to the timespan from when a target enters the surveillance area to the initial recognition of the target.

Target recognition is when the GM-PHD filter extracts the respective estimated state from the GM. The accuracy of the estimates displayed for a lookahead horizon of $H = 8$ in Figure 4.3 is a viable indicator of surveillance performance, showing a rapid response to targets entering the surveillance area. The myopic case $(H = 1)$ shows a delayed reaction time to the target appearances.

4.5 Conclusions and Future Work

This chapter explored the melding of target tracking and persistent surveillance via a UAV fleet. The mathematical scenario description was provided in terms of a POMDP. Using the popular GM-PHD tracking algorithm combined with NBO lookahead control method provided a natural path for the joining of these two objectives. While not a comprehensive study, the simulations provide promising results for the further exploration of this approach.

Throughout this study we realized that the concept of birth intensities, found in the PHD filter, can be used within a non-myopic NBO control algorithm to promote area surveillance by UAV's tasked with tracking targets. While the GM-PHD filter provides a natural adaptation to this method it is apparent that other tracking algorithms (e.g., joint probabilistic data association, multiple hypothesis tracking) could similarly be adapted for the inclusion of surveillance concerns. Moreover, these birth intensities need not be static locations throughout the scenario but can be considered as adaptive components. These and other similar extensions are viable research avenues.

Chapter 5

Simultaneous Non-Myopic Optimization of UAV Guidance and Camera Gimbal Control for Target Tracking

The world of unmanned aerial vehicles (UAVs) has advanced dramatically in the past five years to a point where the general consumer can buy a UAV off the shelf equipped with a gimbal-mounted camera with video streaming capability. While most of the consumer-level platforms are neither intended for autonomous flight and camera gimbal control nor equipped with video processing for target tracking, it shows the market availability of small, light-weight, inexpensive hardware. This same trend has also made such hardware available for research and commercial applications.

In the following research, we direct our attention toward UAVs in conjunction with steerable cameras (gimbals). This combination provides extended control over the observational capabilities. Even without the addition of gimbal controls the mobile platform sensor resource management problem is not trivial. In our past research, we have investigated this topic under many perspectives, including line-of-sight obstacles [50], environmental factors [54], tracking and surveillance [61], etc. These lines of research have continually revealed the advantages of non-myopic control.

In the context of gimbal mounted cameras a myopic approach is often amenable, primarily when considering a gimbal with an unrestricted slew rate with respect to the control step; the gimbal can reach *any* position at the next control input epoch, given its current pose. Literature focusing on the geolocalization of ground targets frequently falls under these assumptions [62]. Another common deterministic approach is to fly a consistent radial distance from the target's expected location while pointing the camera at the center of the UAV orbit (at ground elevation).

A non-myopic gimbal control approach arises twofold in our problem scenario. First, as previously mentioned, our primary focus is on non-myopic autonomous UAV flight control. UAV

navigation and the gimbal camera are intrinsically intertwined; the camera aim and the UAV positioning must be coordinated to ensure target observations. Thus, for non-myopic optimization of the UAV flight path, the future camera angles must also be considered. Secondly, the gimbals considered herein have limited slew rates. Such limitations may originate from hardware constraints or application specifics. In the latter case, a prominent reason is maintaining slow enough camera movement so the streaming video feed can benefit a system operator through its viewing, aiding the human-in-the-loop.

The target-tracking scenario formulation includes details representing common, real-world concerns. First, randomness is represented through the target motion models and random measurement errors. Additionally, the randomness of the measurement error is depends on the distance at which a target is observed. The observations emulate a camera sensor with an image processing algorithm. The returned measurements are two-dimensional world coordinates with heightened angular resolution compared to the range resolution.

Second, the observable space is restricted by the camera's limited field of view (FOV). Modeling the camera sensors to include their projected FOVs based on the UAV and gimbal orientations injects dynamic constraints on the observational capabilities of the system. At low altitudes, this restriction produces a severe impact by shrinking the FOV. Beyond the FOV stipulations, further constraints are present in the control architecture. The UAV motion must abide by the kinematic constraints, restricting the control space (longitudinal acceleration and bank angles) and the velocities. The gimbal movements are also restricted by way of a maximum limit on their slew rates in both the azimuth and elevation adjustments; there is a limited set of reachable gimbal orientations based on the present gimbal position. Each of these impediments inhibits the system's performance in their own right.

There are several means of assessing the capability of a target-tracking system such as track loss or information gain. In general, the objective function of a target-tracking scenario represents the maximization of the quality of the target estimates over time. Our present study takes the

direct representation of maintaining the best estimates of the target positions, minimizing the mean squared tracking error.

These factors, randomness, restricted observations, limited motion models, and bounded control inputs reflect real-world concerns sufficiently to make our investigation a meaningful first step into non-myopic UAV and gimbal camera control. All of these aspects are detailed in the problem formulated as a partially observable Markov decision process (POMDP) [49] in Section 5.1. The POMDP formulation supports a common conversion to a belief-state Markov decision process (MDP). After defining the belief-state MDP, standard MDP solution theory applies directly to the control problem. Our specific solution approximation, nominal belief-state optimization (NBO) [50], supplies *lookahead* capabilities through receding horizon control. To enhance the lookahead of the NBO, we pair it with a heuristic expected cost-to-go, detailed in Section 5.3. Section 5.4 is the culmination of the study, presenting the empirical results of several scenarios including an example of multiple UAV coordinated control.

5.1 Problem Formulation

Describing the problem in a way which lends itself to the application of the NBO is succinctly done by defining a partially observable Markov decision process. The following POMDP element descriptions follow closely with [50] and [54] but include the new unique considerations for gimbal controls and FOV restrictions introduced in the previous section.

The POMDP constituents are as follows:

State

Each component of the system is represented as a member of the POMDP state. The sensor state S_k includes all UAV positions, velocities, orientations (Euler angles) and the accompanying camera gimbal azimuth and elevation angles. Specifically, $S_k = (s_k, G_k)$ where s_k contains the UAV positions, velocities, and Euler angles, $G_k = (A_k, E_k)$ represents the camera gimbal azimuth and elevation angles. The target state χ_k contains the positions and velocities all current targets. Finally, the filter state $\mathcal{F}_k = (\xi_k, \mathbf{P}_k)$ is a representation of the Gaussian distributions of the target

states where ξ_k are the posterior means and $\mathbf{P}_k$ are the posterior covariance matrices generated via a Kalman Filter. The POMDP state can then be summarized as $X_k = (S_k, \chi_k, \mathcal{F}_k)$.

Actions

The actions are defined as a tetrad of vectors, $u_k = (a_k, \theta_k, \alpha_k, \epsilon_k)$. Then, $a_k = \left[a_k^{(1)} \dots a_k^{(N_s)}\right]^\top$ and $\theta_k = \left[\theta_k^{(1)} \dots \theta_k^{(N_s)}\right]^\top$ represent the control inputs for forward acceleration and bank angle, respectively, for N_s UAVs. The vectors $\alpha_k = \left[\alpha_k^{(1)} \dots \alpha_k^{(N_s)}\right]^\top$ and $\epsilon_k = \left[\epsilon_k^{(1)} \dots \epsilon_k^{(N_s)}\right]^\top$ are the gimbal slew rate commands for azimuth and elevation, respectively. All of components of the action are constrained: $a_k^{(i)} \in [-a^*, a^*]$, $\theta_k^{(i)} \in [-\theta^*, \theta^*]$, $\alpha_k^{(i)} \in [-\alpha^*, \alpha^*]$, and $\epsilon_k^{(i)} \in [-\epsilon^*, \epsilon^*]$.

Observation

We take the sensor states and filter states to be completely observable; their observations *are* their POMDP components. The target states are, in general, unknown and only exposed via sensor measurements. A sensor measurement $z_k^i = [x_i \; y_i]^\top$ is a two dimensional position vector. The observation at time k is the set of all N_m sensor measurements $\mathrm{Z}_k = \{z_k^1, \dots, z_k^{Nm}\}$.

Observation Law

A measurement z_k^i is only generated when a target position χ_k^{pos} is contained in the projected FOV of the camera,

$$z_k^i = \begin{cases} \chi_k^{\text{pos}} + \nu_k^{(z)}, & \chi_k^{\text{pos}} \text{ visible} \\ \emptyset, & \text{otherwise} \end{cases}, \tag{5.1}$$

where $\nu_k^{(z)} \sim \mathcal{N}\left(0, \mathbf{R}\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right)\right)$. The visibility of the target is determined in the same manner as in [62], based on the camera specifications, UAV and gimbal orientations and UAV altitude. The observation covariance matrix $\mathbf{R}\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right)$ defines a 10% uncertainty in the range from sensor to target and 0.01π radian angular uncertainty. Letting $r_k = \|s_k^{\text{pos}} - \chi_k^{\text{pos}}\|$ and ρ_k be the angular measurement between the sensor and target then,

$$\mathbf{R}\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right) = \mathbf{G}(\rho_k) \begin{bmatrix} .1 \times r_k & 0 \\ 0 & .1\pi \times r_k \end{bmatrix} \mathbf{G}(\rho_k)^\top$$

where $\mathbf{G}(\rho_k)$ matrix representing the counterclockwise rotation of angle ρ_k. This imposes the spatially varying measurement error. Note, we enforce a minimum effective range of ten meters to avert computational issues inside the Kalman filter updates of the filter state.

State Transition Law

The state transition law $X_{k+1} = f\left(X_k | u_k\right)$ defines the mapping from the state at time k to time $k+1$. The function $f(\cdot|u_k)$ is decomposable into independent functions representing the different components of the state: sensor state, target state and filter state. We further decompose the sensor state transition into the UAV state and the camera gimbal state transitions. The UAV state transition $S_{k+1} = f^s\left(S_k, a_k, \theta_k\right)$ is characterized by a common UAV kinematic motion model. For brevity, we refer the reader to [54] for further details. The camera gimbal update is encapsulated in two simple equations; the azimuth and elevation for the ith camera gimbal are updated as $A_{k+1}^{(i)} = A_k^{(i)} + \Delta_k \alpha_k^{(i)}$ and $E_{k+1}^{(i)} = E_k^{(i)} + \Delta_k \epsilon_k^{(i)}$, respectively, where Δ_k is the length of the kth time-step. The target transitions are independent and can be defined individually for the ith target as

$$\chi_{k+1}^{(i)} = \mathbf{F}_k\, \chi_k^{(i)} + \nu_k^{(\chi)},\ \nu_k^{(\chi)} \sim \mathcal{N}\left(0, \mathbf{Q}_k\right), \tag{5.2}$$

the standard nearly constant velocity (NCV) motion model [63]. Finally, the filter state evolves based on the Kalman filter.

One-step Cost

The immediate cost incurred by taking action u_k given the POMDP state X_k is called the one-step cost. Here we set the one-step to be the mean-squared tracking error,

$$c\left(X_k, u_k\right) = \mathrm{E}_{\nu_k^{(\chi)}, \nu_k^{(z)}}\left[||\chi_{k+1} - \xi_{k+1}||^2 \,|X_k, u_k\right]. \tag{5.3}$$

With the POMDP sufficiently characterized we move to solution approaches. The theory behind solving POMDPs is closely related to that of the standard Markov decision processes (MDP) [42]. The intrinsic difference is the knowledge of the underlying state. This discrepancy can be handled through a conversion to a *belief-state MDP*.

5.2 POMDP Solution

A POMDP is an extension of an MDP where the system states are not completely observable (e.g., target positions). The goal of the MDP solution methods is to minimize the cumulative cost over the scenario's horizon H, $V_H(X_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} c(X_k, u_k)\right]$, through the selection of action u_k. The expectation is taken with respect to the random progression of the states X_k. We wish to perform the same optimization in the case of a POMDP.

MDP solution theory can only be applied to to completely observable state. This leads to the formulation of a belief-state MDP. A belief-state MDP is formed by taking the posterior distribution of the POMDP state as the state of the MDP. Let $b_k(X) = P_{X_k}(X|\mathrm{Z}_0, ..., \mathrm{Z}_k; u_0, ..., u_{k-1})$ be the *belief state* where the distribution is conditioned on the observation and action history to time k. The belief state can be separated into components similar to the POMDP state, $b_k = (b_k^S, b_k^\chi, b_k^\mathcal{F})$. The transition law of the belief-state MDP is the belief-state update computed via the combination of the state transition and observation laws of the POMDP. The action space of the belief-state MDP is exactly that of the POMDP. Lastly, the corresponding cost is $C(b_k, u_k) = \int c(X, u_k) b_k(X) dX$. Now, the cumulative belief cost over horizon H can be written as

$$V_H(b_0) = E\left[\sum_{k=0}^{H-1} C\left(b_k, u_k\right) \mid b_0\right], \tag{5.4}$$

and MDP solution theory may be applied [42].

5.2.1 Belief-State Update

The belief-state update is the propagation of the posterior distribution b_k accomplished by Bayesian filtering. In the target-tracking paradigm there are some common assumptions made

with respect to the distributions of the target estimates. Depending on the assumptions adopted, there are a wide variety of techniques known as data association filters (e.g., PHD, MHT, JPDA) which approximate the exact Bayesian filter. To emphasize the benefits of non-myopic control of the restricted gimbal-mounted camera aiming, we forgo incorporating data association techniques and assume perfect knowledge of observation to target associations.

Under the assumption of perfect data association and the linear Gaussian system, a target's belief state is represented by a Gaussian distribution such that $b_k^{\chi^{(i)}} \sim \mathcal{N}\left(\xi_k^{(i)}, \mathbf{P}_k^{(i)}\right)$. The belief-state update is then performed through standard Kalman filter equations. The POMDP transition and observation laws furnish the required details to compose said Kalman filter.

5.2.2 Optimal Policy

Given the belief-state MDP, an optimal policy is the mapping from the belief-state space to action space which minimizes the cost V_H from (5.4). Exercising *Bellman's principle* allows the optimal objective function to be interpreted as

$$V_H^*(b_0) = \min_u \left\{ C(b_0, u) + \mathrm{E}\left[V_{H-1}^*(b_1) \,|b_0, u \right] \right\}. \tag{5.5}$$

To illustrate the optimal policy $\pi^*(b)$, the Q-value is designated as $Q_H(b_0, u) = C(b_0, u) + \mathrm{E}\left[V_{H-1}^*(b_1) \mid b_0, u\right]$, then the optimal objective cost and the optimal policy can be expressed as

$$V_H^*(b_0) = \min_u Q_H(b_0, u) \tag{5.6}$$

$$\pi_k^*(b_k) = \arg\min_u Q_H(b_k, u). \tag{5.7}$$

If we fix H as a constant, independent of k, the optimal policy is $\pi^*(b) = \arg\min_u Q_H(b, u)$ and $Q_H(b, u) = C(b, u) + \mathrm{E}\left[V_{H-1}^*(b') \mid b, u\right]$ where b' is the random next belief state after taking action u. This is known as *receding horizon control*.

5.2.3 Heuristic Expected Cost-to-Go

Fixing H for receding horizon control effectively truncates the horizon over which the Q-value evaluates the "goodness" of an action u; the impact of choosing action u is not assessed beyond H steps into the future. This truncation is beneficial since propagating the belief state can be computationally expensive. Also, pragmatically, model prediction accuracies wane over extended horizons. However, in certain situations it is beneficial to obtain an approximation of how a current action selection influences the costs incurred beyond the horizon H. The inclusion of a heuristic expected cost-to-go (HECTG) can both provide the desired benefits and avoid the deficiencies of extending the horizon. Denote the HECTG as $\tilde{V}(b_H)$. Once the horizon H has been fixed for receding horizon control and the HECTG is included, then

$$V_H(b_0) = \mathrm{E}\left[\sum_{k=0}^{H-1} C(b_k, u_k) + \tilde{V}(b_H) \mid b_k\right]. \tag{5.8}$$

The corresponding approximations of the Q-value and the optimal policy directly follow from (5.8). $\tilde{V}(b_H)$ is not intended to exactly compute the remaining cost-to-go. It is only an approximation to rank the actions for selection.

5.2.4 Nominal Belief-State Optimization

The receding horizon control discussed in the previous section provides an avenue for reduced computational burden through the truncation of the horizon. However, the evaluation of the expectation, $\mathrm{E}[\cdot]$, in (5.8), usually performed by Monte Carlo simulation, is still computationally expensive. To eliminate this burden we choose to employ NBO [50] as a Q-value approximation. NBO replaces the expectation from (5.8) with the simple summation $\sum_{j=0}^{H-1} C(\hat{b}_{k+j}, u_{k+j})$ where the nominal belief-state progression $\hat{b}_k, \ldots, \hat{b}_{k+H-1}$ is substituted for the random propagation of the belief states. Under NBO with receding horizon control, V_H can be approximated as

$$V_H(b) \approx \sum_{k=0}^{H-1} C(\hat{b}_k, u_k) + \tilde{V}(\hat{b}_H). \tag{5.9}$$

NBO seeks to minimize the cumulative cost function approximation over the set of actions $\{u_k\}_{j=0}^{H-1}$. Note, (5.9) includes the HECTG term. The standard NBO cumulative cost function is seen when $\tilde{V}(\hat{b}_H) = 0$.

5.3 Application Specifics

Section 5.2 concisely depicts the POMDP solution techniques to be employed. Next we apply these techniques to our specific target-tracking scenario. Here we explain the application's nominal belief-state update and NBO cumulative cost function. As proposed, we include a HECTG in the NBO cumulative cost. We compose this HECTG function tailored to the restricted gimbal setup herein.

5.3.1 Nominal Belief-state Update

As mentioned in Section 5.2.1, we assume our target belief states to be represented as $b_k^{\chi^{(i)}} \sim \mathcal{N}\left(\xi_k^{(i)}, \mathbf{P}_k^{(i)}\right)$. At any time k, we take $\hat{b}_k^{\chi^{(i)}} = b_k^{\chi^{(i)}}$ to initialize the NBO. Then the nominal belief-state propagation follows a Kalman filter with exactly zero noise and accounts for the POMDP observation law, using $\hat{\xi}_k^{(i)}$ as the target location. The progression of the ith target's belief state is as follows [64]:

$$\begin{aligned}
\hat{\xi}_{k+1}^{(i)} &= \mathbf{F}_k \hat{\xi}_k^{(i)} \\
\hat{\mathbf{P}}_{k+1} &= \begin{cases} \left[[\hat{\mathbf{P}}_{k+1|k}^{(i)}]^{-1} + \mathbf{S}_{k+1}^{(i)} \right]^{-1}, & \hat{\xi}_{k+1}^{(i)} \text{ visible} \\ \hat{\mathbf{P}}_{k+1|k}^{(i)}, & \text{otherwise} \end{cases}
\end{aligned}$$

The predicted covariance matrix is $\hat{\mathbf{P}}_{k+1|k}^{(i)} = \mathbf{F}_k \hat{\mathbf{P}}_k^{(i)} \mathbf{F}^\top + \mathbf{Q}_k$ and the innovation matrix is $\mathbf{S}_{k+1}^{(i)} = \mathbf{H}_{k+1}^\top \left[\mathbf{R}\left(s_{k+1}^{\text{pos}}, \hat{\xi}_{k+1}^{(i)} \right) \right] \mathbf{H}_{k+1}$.

5.3.2 NBO Cumulative Cost

Following from the nominal state progression equations it has been shown in [50] that the tracking error objective function from Section 5.1 in terms of the nominal belief state is equivalently,

$$V_H\left(\hat{b}_0\right) = \sum_{k=0}^{H-1}\sum_{i=1}^{N_T} \text{Tr}\,\hat{\mathbf{P}}_{k+1}^{(i)} + \tilde{V}(\hat{b}_H), \tag{5.10}$$

where N_T is the number of targets in the nominal belief state and Tr is the matrix trace function. The exact formulation of the HECTG $\tilde{V}(b_H)$ is presented in the following section.

5.3.3 Weighted Trace Penalty

We adopt the name of weighted trace penalty (WTP) from [50] for our HECTG function. This nomenclature suits our HECTG which involves Tr **P**. Our WTP function is uniquely suited to the gimbal-mounted camera platform, accounting for the maximum slew rates, the limited FOV, and the restricted UAV dynamics.

We recognize that the longer a target i remains unobserved, the larger the Tr $\mathbf{P}^{(i)}$ grows over time. This inspires us to create a WTP based on the product of the current Tr $\mathbf{P}^{(i)}$ and the estimated time to observation. Given the nominal target states after an H-step lookahead $\left\{\hat{\xi}_H^{(i)}, \hat{\mathbf{P}}_H^{(i)}\right\}_{i=1}^{N_T}$ and the corresponding sensor state $\hat{S}_H$ resulting from the application of actions $\{u_k\}_{j=0}^{H-1}$ we define three components to the WTP, the flight time, the turning time, and the gimbal adjust time.

Assuming target i is not visible at time H, we define a position $e_{xy}^{(i)} = (e_x^{(i)}, e_y^{(i)})$ at which a UAV should observe target i to maximize the reduction in Tr $\mathbf{P}^{(i)}$. It can be shown that $e_{xy}^{(i)}$ lies on the eigenvector of $\mathbf{P}^{(i)}$ with the maximum eigenvalue. For simplicity we restrict $e_{xy}^{(i)}$ to be a fixed distance from the target on this *primary eigenvector* such that $\|\xi_H^{(i)} - e_{xy}^{(i)}\| = D$. Now, the flight time is $T_1 = \frac{\|\hat{s}_H^{pos} - e_{xy}^{(i)}\|}{\mathcal{V}^+}$ where $\mathcal{V}^+$ is the maximum velocity of the UAV.

Next, we compute difference between the current UAV heading and the direction to e_{xy} as

$$L = \left| \tan^{-1}\left(\frac{\hat{s}_{\dot{y}}}{\hat{s}_{\dot{x}}}\right) - \tan^{-1}\left(\frac{e_y^{(i)} - \hat{s}_y}{e_x^{(i)} - \hat{s}_x}\right) \right|$$

where $(\hat{s}_x, \hat{s}_y)$ and $(\hat{s}_{\dot{x}}, \hat{s}_{\dot{y}})$ are the UAV x an y coordinates and velocities, respectively. Then, the turning time is $T_2 = \frac{L}{\mathcal{V}_R^+}$ where $\mathcal{V}_R^+$ is maximum radial velocity of the UAV.

Finally, we define the gimbal adjust time as,

$$T_3 = \max\left(\frac{|\hat{A}_H - A^*|}{\alpha^*}, \frac{|\hat{E}_H - E^*|}{\epsilon^*}\right)$$

where the gimbal azimuth and elevation required to aim the camera at the target from $e_{xy}^{(i)}$ is computed as $A^* = \tan^{-1}(\|e_{xy} - \hat{s}_x y\|/\hat{s}_z)$ and $E^* = \tan^{-1}\left(\frac{e_y^{(i)} - \hat{s}_y}{e_x^{(i)} - \hat{s}_x}\right) - \tan^{-1}\left(\frac{\hat{\xi}_H^{(i)}(y) - e_y^{(i)}}{\hat{\xi}_H^{(i)}(x) - e_x^{(i)}}\right)$, respectively. The UAV altitude is denoted as $\hat{s}_z$. Now, the time to observation is $\mathcal{T}(b_H) = T_1 + \max(T_2, T_3)$. The maximum arises from the fact that the gimbal can move independently while the UAV is in motion. The WTP is then defined as

$$\tilde{V}(b_H) = \gamma \mathcal{T}(\hat{b}_H) \text{Tr}\, \hat{\mathbf{P}}_H^{(i)} \tag{5.11}$$

such that

$$i = \arg\max_{i \in m} \text{Tr}\, \hat{\mathbf{P}}_H^{(i)}$$

and $m = \{i \mid \hat{\xi}_H^{(i)} \text{is not visible}\}$. The constant γ is a scaling factor.

While this formulation only defines a HECTG for a single UAV tracking multiple targets, there is a simple extension to multiple cooperative UAVs found in [50]. Using our WTP, we employ this extension in our final simulation setup which involves two UAVs.

5.4 Experiments

The following experiments are performed over 250 Monte Carlo simulations per system configuration. Each Monte Carlo sample simulates the tracking system of a short term window (200 seconds). The maximum gimbal slew rates and the NBO horizon lengths are varied to display the effects on the tracking quality. Throughout all of the experiments a control decision including both the UAV and gimbal specific controls is made every two seconds.

The choice of camera sensor is homogeneous across the all the simulations. We model our sensor after a commercial off-the-shelf camera [65] which has an angular FOV of approximately 50.5×31.6 degrees. Additionally, in all the following experiments the maximum slew rates of the gimbal azimuth and elevation are considered equal, $\alpha^* = \epsilon^*$; the gimbal motion is limited to the same rate in both the azimuth and elevation directions.

We realize that at lower altitudes the FOV coverage is severely limited (small area), reducing the number of observations collected. This impairment leads to *track loss*. Once the accuracy of a target estimate diminishes due to lack of observations, the action selection also suffers. To prevent perplexity in the target-tracking performance analysis we present the relevant statistics in two parts, track-loss percentage and average tracking error.

5.4.1 Track Loss

To determine the track-loss percentages of each experiment we define the criteria for a lost track. For track i, let the set $\kappa^{(i)}$ be the set of all time steps at which the tracking accuracy produces a confident target estimate based on the covariance matrix $\mathbf{P}_k^{(i)}$ such that $\kappa^{(i)} = \{k \mid \text{Tr}\, \mathbf{P}_k^{(i)} \leq \Omega\}$ where Ω is a fixed threshold determined through the analysis of the simulation data. Also, let the terminal time step of a simulation be denoted as $\bar{k} = 200$. If a Monte Carlo sample simulation has experienced track loss then $\prod_{i=1}^{N_T} \mathbf{1}_{\kappa^{(i)}}(\bar{k}) = 0$. The indicator function $\mathbf{1}_{\kappa^{(i)}}(\bar{k})$ shows if the estimate of target i is deemed valid or not at the final time step of the simulation. If a simulation experiences track loss, the tracking errors from the simulation do not contribute to the average tracking error statistics in the following results.

5.4.2 Single UAV and Single Target

This first experiment is presented as a motivational example, highlighting the need for lookahead control when using limited slew rate gimbal-mounted sensors. A single UAV attempts to track a target moving at constant velocity (10 m/s) from an altitude of 400 meters. The NBO algorithm is calculated sans WTP in order to show the direct effects of the NBO horizon. Table 5.1 contains the track-loss percentages for slew rate limits versus NBO horizon lengths. While the performance

is not impressive, revealing high track-loss percentages, the results clearly show the benefits of lookahead. The general trend shows that as the horizon length is increased the track-loss percentage is reduced. The track-loss percentage results accentuate the need for extended NBO horizons. Note, the average tracking error is not presented due to high volume of lost tracks.

Table 5.1: Track-loss Percentage: 1 UAV, 1 Target

Slew Rate Limit	NBO Horizon				
	1	2	4	6	8
$12\,\mathrm{deg}\,/\,\mathrm{sec}$	82	69	48	49	49
$24\,\mathrm{deg}\,/\,\mathrm{sec}$	84	72	59	50	46

5.4.3 Single UAV and Two Targets

Our second experiment involves a single UAV at an altitude of 200 meters tracking two targets. The targets maintain a separation of 250 meters while moving at a constant velocity of 10m/s. The simulation is run for multiple NBO horizons and maximum slew rates. Each combination is detailed in Table 5.2 with their accompanying track-loss percentages.

There is a significant reduction in the track-loss percentages due to the inclusion of the WTP term in the NBO cost. Even with the addition of a second target and lower altitude, the track-loss percentages are far below those generated in the previous experiment. The WTP performs quite well at artificially extending the lookahead horizon.

Figure 5.1 displays the empirical cumulative distribution functions (ECDF) of the average tracking errors (in meters) for gimbal slew-rate limits of 12, 24, 30, and 40 degrees per second. Recall, these ECDF plots only include the simulations which maintained valid tracks on *both* targets. With respect to the tracking error results, the scenarios with lower limits on the gimbal slew rates show the advantages of longer NBO horizons. However, as the gimbal slew-rate constraints are relaxed the performance of the one-step NBO (with WTP) attains nearly equivalent performance.

Table 5.2: Track-loss Percentage: 1 UAV, 2 Targets

Slew Rate Limit	NBO Horizon			
	1	2	4	6
$12\,\mathrm{deg}\,/\,\mathrm{sec}$	23.6	16.0	8.4	9.0
$24\,\mathrm{deg}\,/\,\mathrm{sec}$	8.4	10.4	3.6	3.6
$30\,\mathrm{deg}\,/\,\mathrm{sec}$	8.0	6.0	6.8	4.0
$40\,\mathrm{deg}\,/\,\mathrm{sec}$	6.4	5.6	5.2	4.0

5.4.4 Two UAVs and Three Targets

Our final experiment entails two UAVs tracking three targets from an altitude of 400 meters. The primary goal of this set of simulations is to display the cooperative control of multiple UAVs. While the previous experiments were performed on targets with fixed velocity Figure 5.2 is a sample plot of the simulation showing the target and UAV trajectories. The displayed target trajectories (solid lines) are the same across all Monte Carlo runs in this experiment but the UAV paths (dashed lines) change from sample to sample. The plot yields an intuition as to how the UAV's cooperate to maintain tracks on all three targets.

Examining the track-loss percentages in Table 5.3 and tracking error results in Figure 5.3 presents a a dichotomy. The results of the one-step NBO shows a lower track loss percentage but the tracking error on the maintained tracks is better for the NBO horizon $H = 6$. This is a consequence of the WTP dominating the NBO cost when H is small and the targets are widely separated.

Table 5.3: Track-loss Percentage: 2 UAVs, 3 Targets

Slew Rate Limit	NBO Horizon			
	1	2	4	6
$12\,\mathrm{deg}\,/\,\mathrm{sec}$	30.0	44.0	28.0	19.2
$24\,\mathrm{deg}\,/\,\mathrm{sec}$	3.2	13.6	9.6	4.8

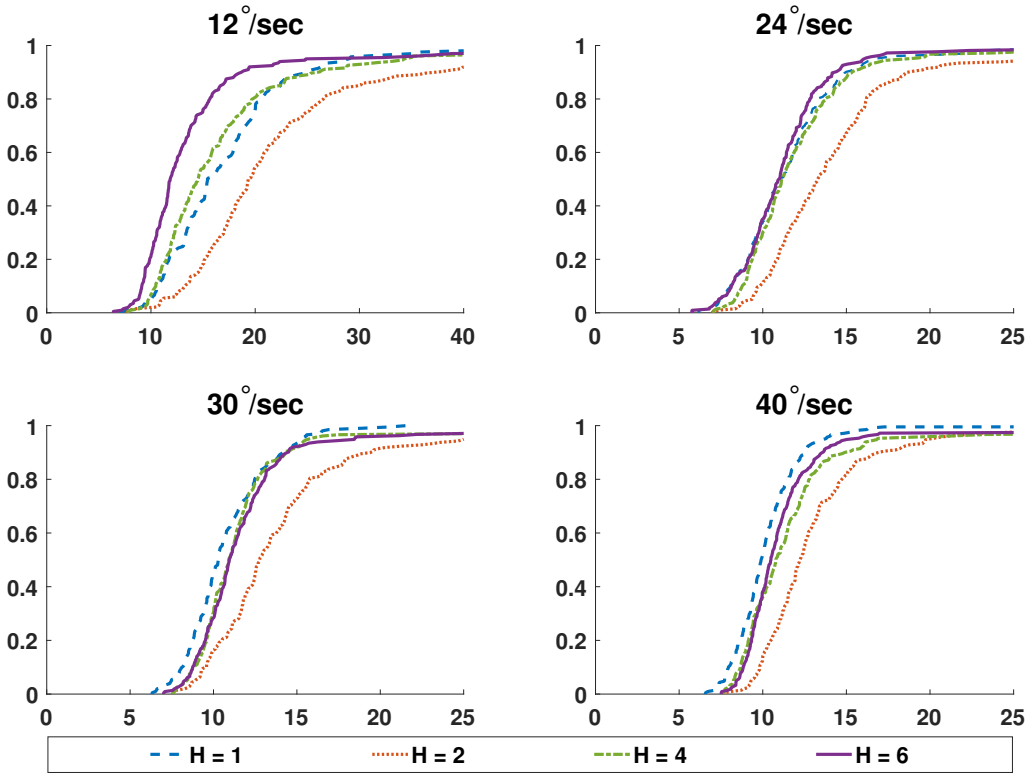


Figure 5.1: Tracking error (meters) ECDF: 1 UAV and 2 targets.

5.5 Conclusions

Unmanned aerial vehicles with gimbal mounted cameras provide an opportunity for fine-grain control over target-tracking observations. To exploit non-myopic UAV navigation control, the gimbal controls must be determined concurrently. This becomes adamantly apparent when the gimbal steering rates are restricted. The POMDP formulation herin factors in this extended action space and the attending control constraints. The effect of lookahead optimization is seen by applying Nominal Belief-State Optimization. Augmenting NBO with our form-fitted HECTG, Weighted Trace Penalty, further enhances the performance gain derived from predictive control. The improvements garnered from elongated NBO horizons and WTP are seen in the abatement of both the track-loss frequency and the mean-squared tracking error.

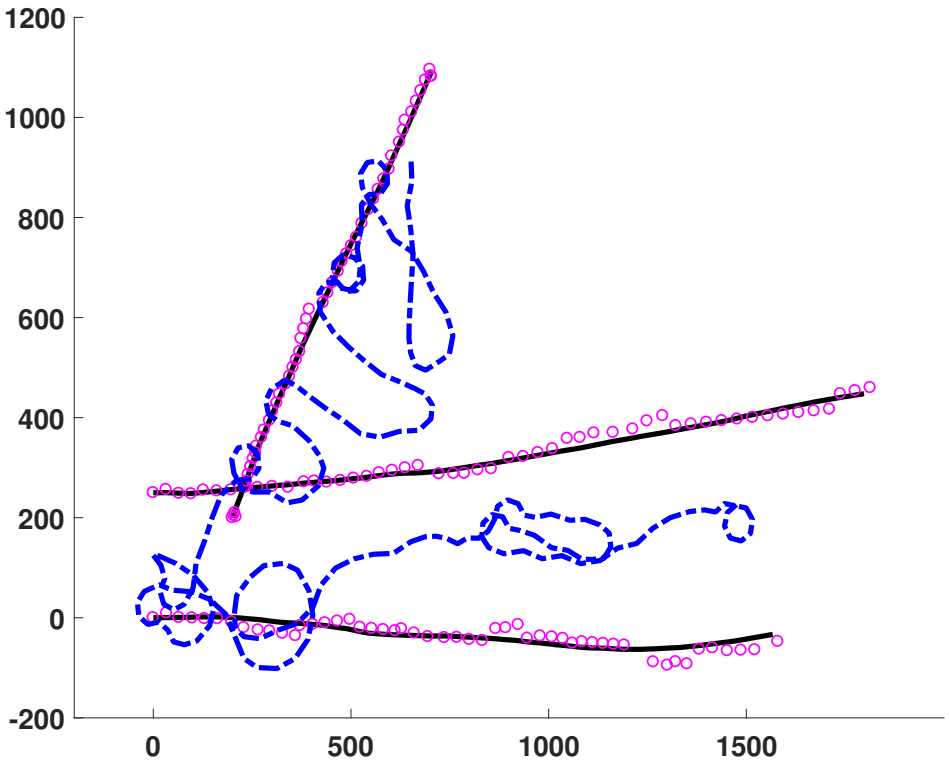


Figure 5.2: Typical scenario plot of NBO horizon $H = 6$ and maximum slew rate of 12 degrees per second. The solid lines are the target trajectories, the dashed lines are UAV paths and the $\circ$ symbolize the estimated target locations.

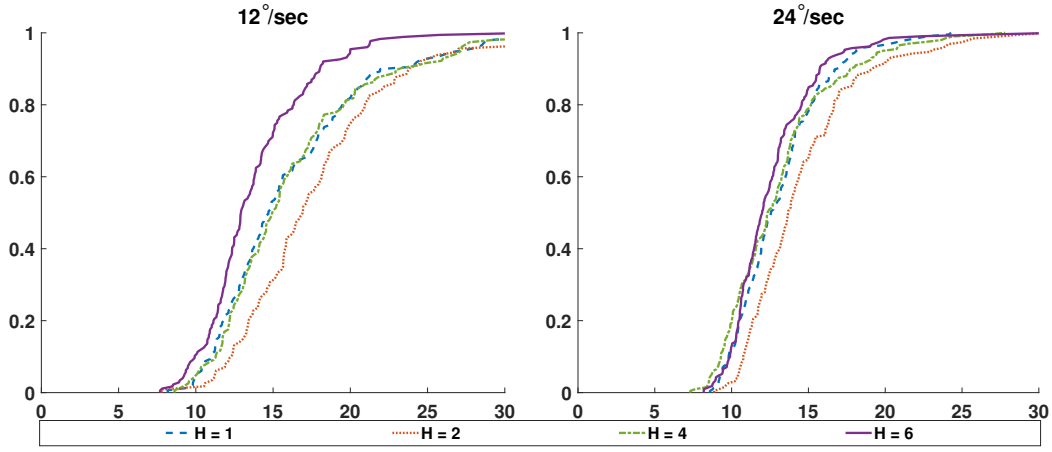


Figure 5.3: Tracking error (meters) ECDF:2 UAVs and 3 Targets

Chapter 6

Fuel Efficient Moving Target Tracking using POMDP with Limited FOV Sensor

Unmanned aerial vehicles (UAVs) are well suited for performing dull, dirty, and dangerous missions normally performed by manned aircraft. One mission of critical interest is tracking moving targets. Significant research has been performed in numerous UAV control areas including path planning for target tracking [66]. However, the majority of path planning algorithms developed to date are focused on completing a specific mission task with a focus on tracking accuracy, target coverage, or expediency of mission completion. This approach provides satisfactory performance but generally result in less than desired UAV dynamics and limited endurance. There is limited consideration for fuel efficiency of the vehicle in the decision process of most of the algorithms. Ignoring the fuel efficiency concern can result in shorter missions or limit the mission effectiveness.

Previously, we had presented a robust path planning algorithm for a limited field of view sensor utilizing a partially observable Markov decision process (POMDP) with a nominal belief-state optimization (NBO) approximation [64], [67], [68]. These efforts have shown the ability of POMDP with NBO to perform effective path planning for tracking fixed and moving targets with a limited field of view (FOV) sensor. Additionally, collision avoidance and wind compensation schemes were also shown to be effective using the same POMDP algorithms [54]. The POMDP with NBO approach uses a receding horizon method that is computationally efficient and provides effective responses to changes in target dynamics or environmental changes. This method also provided a robust capability for tracking the targets across a range of altitudes and sensor configurations.

Here, we design a UAV control method that uses POMDPs with NBO approximation for a fixed FOV sensor with fuel efficiency considerations. We build upon our previous efforts [50,54,64,67], that have shown the POMDP provides sufficient path planning for a UAV tracking of ground based

fixed or moving targets. The cost function is modified to add in variable fuel burn considerations to improve mission endurance. The performance is evaluated using the tracking errors and the fuel burned during the simulations for multiple fuel burn options. The contribution of this paper is to show that the POMDP with NBO can provide effective target tracking while improving fuel efficiency. In this paper, we focus on non-evasive moving targets with limited observations.

6.1 Problem Specification

Ground targets moving in 2-D at constant speed will be tracked. A simplified UAV motion model with forward acceleration and bank angle controls is used. The UAV has a limited speed range that is controlled by commanding longitudinal acceleration. The UAV bank angle is used to change UAV heading, while also changing the sensor observation area. A fixed FOV sensor mounted on the bottom of the vehicle provides a limited observation measurement of target location. Spatially varying random errors, based on UAV and target locations, corrupt the accuracy of the target location measurements. The fuel burn of the UAV is calculated based on the speed and bank angle of the UAV. The UAV altitude can be set for a given scenario, but will maintain a constant altitude during that entire scenario. Changing the UAV altitude is effectively the same as changing the sensor size or FOV. The objective is to provide a spectrum of options to balance the minimization of the mean-squared tracking error with the fuel efficiency of the UAV.

6.2 POMDP and NBO Approximation

A POMDP is a discrete time controlled dynamical process that is useful for modeling resource control problems. A POMDP, unlike a Markov Decision Process that has perfect knowledge of the system state, has some state information that is not directly observed. The NBO approximation method has been shown as computationally efficient for guidance optimization. The method described here builds on our earlier papers [50, 54, 64, 67, 68].

6.2.1 POMDP Ingredients

States

The POMDP states represent time evolving system features. Three subsystems are defined: the sensors, the targets, and the tracker. At time k, the state is defined as $x_k = (s_k, \chi_k, \xi_k, \mathbf{P}_k)$, where s_k is the state of the sensor, χ_k is the target state, and $(\xi_k, \mathbf{P}_k)$ is the tracker state. The sensor state is the UAV position and velocities, and the target state is the target position, velocities, and accelerations. The tracker state is a posterior mean vector, ξ_k, and posterior covariance matrix, $\mathbf{P}_k$, of a standard Kalman filter [63, 69].

Actions

The actions for this problem, or the available system controls, are UAV forward acceleration and the bank angle. Specifically, at time k the forward acceleration, a_k, and bank angle, ϕ_k, define actions $u_k = (a_k, \phi_k)$ of the UAV.

Observations and Observation Law

The target states are not fully observable; at any given time a random observation of the target state may be available. Let χ_k^{pos} be the target position vector and s_k^{pos} be the sensor/UAV position vector. The observation of the target's position can be expressed by:

$$z_k^{\chi} = \begin{cases} \chi_k^{pos} + w_k, & \text{if target is visible} \\ \text{no measurement}, & \text{otherwise} \end{cases} \tag{6.1}$$

where the random measurement error, w_k, has a distribution dependent on target and UAV locations. Full observability is assumed for Sensor/UAV and tracker states.

State-Transition Law

The state-transition law defines the next-state distribution of the three predefined subsystems given a current state and action pair of each subsystem. The sensor state evolves by $s_{k+1} = \Psi(s_k, u_k)$, where Ψ is a mapping function (defined later). The target state progresses according to

$\chi_{k+1} = f(\chi_k) + \nu_k$, with independent and identically distributed random noise sequence, ν_k, and a target motion model, f (also defined later). Kalman filter equations with joint probabilistic data association (JPDA) provide tracker state evolution [70, 71]. The update equation is only evaluated when target observations are available, otherwise only the prediction step in the Kalman filter is performed.

Cost Function

Our cost function, which is the action cost for a given state, considers the mean-squared error between the tracks and the targets:

$$C(x_k, u_k) = E_{\nu_k, w_{k+1}}[\|\chi_{k+1} - \xi_{k+1}\|^2 | x_k, u_k]$$

Belief State

The underlying state posterior distribution, updated incrementally via Bayes rule with observations, defines the belief state. At time k, the belief state is $b_k = (b_k^s, b_k^\chi, b_k^\xi, b_k^{\mathbf{P}})$, where $b_k^s = \delta(s - s_k)$, $b_k^\xi = \delta(\xi - \xi_k)$, $b_k^{\mathbf{P}} = \delta(\mathbf{P} - \mathbf{P}_k)$ (because of full tracker and sensor state observability), and b_k^χ is the posterior distribution of the target state.

6.2.2 Optimal Policy

The objective is to choose actions that minimize the expected cumulative cost over a time horizon $H, k = 0, 1, ..., H-1$. For a time horizon H, the expected cumulative cost can be written as $J_H = E[\sum_{k=0}^{H-1} C(x_k, u_k)]$. Historical knowledge of all observable quantities up to time $k-1$ should inform the chosen action at time k. If an optimal choice of actions exists, then there exists an optimal action sequence that depends only on "belief-state feedback" [72]. Therefore, the belief states can be used to write the objective function as: $J_H = E[\sum_{k=0}^{H-1} c(b_k, u_k) | b_0]$, where $c(b_k, u_k) = \int C(x, u_k) b_k(x) dx$.

Bellman's principle of optimality [1] provides for the optimal objective function value J_H^* given the current belief state b_0. The optimal objective function can be written as: $J_H^*(b_0) =$

$\min_u\{c(b_0,u) + E[J^*_{H-1}(b_1)|b_0,u]\}$, where b_1 is the random next belief state, J^*_{H-1} is the optimal cumulative cost over the horizon $H-1, k = 1,2,...,H-1$, and $E[\cdot|b_0,u]$ is the conditional expectation given the current belief state b_0 and an action u taken at time $k = 0$. Given the current belief state b_0, the Q-value of taking an action u is defined by $Q_H(b_0,u) = c(b_0,u) + E[J^*_{H-1}(b_1)|b_0,u]$. At time $k = 0$ the optimal policy, from Bellman's principle, can be written as $\pi^*_0(b_0) = \text{argmin}_u Q_H(b_0,u)$. The optimal policy at time k is $\pi^*_k(b_k) = \text{argmin}_u Q_{H-k}(b_k,u)$.

The second Q-function term is difficult to obtain exactly. Studies of numerous methods to approximate the Q-values have been performed [50, 73–79]. The NBO approximation method, introduced in [50] along with other guidance problem approximations and techniques, is used here.

6.2.3 NBO Approximation

Although a few approximation methods to solve POMDPs are available, we chose the NBO method because of low computationally cost relative to other POMDP approximation methods like foresight optimization, hindsight optimization, policy rollout, and Q-learning [72]. In practice, real-time implementation of a UAV guidance algorithm requires a method that is not computationally prohibitive. Our previous work [64, 67] showed acceptable computation efficiency for the NBO algorithm.

Given N_{targs} targets, the target state is represented as $\chi_k = (\chi^1_k, \chi^2_k, ..., \chi^{N_{\text{targs}}}_k)$, where the ith target is represented by χ^i_k. The track-state is $\xi_k = (\xi^1_k, \xi^2_k, ..., \xi^{N_{\text{targs}}}_k)$ and $\mathbf{P}_k = (\mathbf{P}^1_k, ..., \mathbf{P}^{N_{\text{targs}}}_k)$, where $(\xi^i_k, \mathbf{P}^i_k)$ is the ith target track-state. We use a zero-mean noise linearized target motion model for the target-state dynamics, given by $(\forall i)$

$$\chi^i_{k+1} = \mathbf{F}_k\chi^i_k + \nu^i_k, \qquad \nu^i_k \sim N(0,\mathbf{Q}_k) \tag{6.2}$$

with observations as defined in (1). The measurement error is $w^i_k \sim N(0,\mathbf{R}_k(\chi^i_k,s_k))$, and the target motion model (for all targets) is $\mathbf{F}_k$. The ith target state, χ^i_k, includes position coordinates (x_k,y_k), velocities (v^x_k,v^y_k), and accelerations (a^x_k,a^y_k) in x- and y-directions, i.e., $\chi^i_k =$

$[x_k, y_k, v_k^x, v_k^y, a_k^x, a_k^y]^T$. The observation model becomes $\mathbf{H}_k = [\mathbf{I}_{2x2}, \mathbf{0}_{4x4}]$. A constant velocity (CV) model [69, 70] is implemented for target dynamics in (6.2), which defines $\mathbf{F}_k$. The ith target belief state, with assumed Gaussian distributions, can be expressed as $b_k^{\chi^i}(\chi) = N(\chi - \xi_k^i, \mathbf{P}_k^i)$, where ξ_k^i and $\mathbf{P}_k^i$ are the ith target track-states, which evolve according to the JPDA algorithm [70, 71].

The objective function is approximated by the NBO method as $J_H(b_0) \approx \sum_{k=0}^{H-1} c(\hat{b}_k, u_k)$, where $\{\hat{b}_i\}_{i=1}^{H-1}$ is a nominal belief-state sequence over an optimized action sequence $\{u_i\}_{i=1}^{H-1}$ .The ith target nominal belief-state sequence, developed from nominal tracks $(\hat{\xi}_k^i, \hat{\mathbf{P}}_k^i)$ using a zero-noise Kalman filter [69, 70] is: $\hat{b}_k^{x^i}(\chi) = N(\chi - \hat{\xi}_k^i, \hat{\mathbf{P}}_k^i)$, where $\hat{\xi}_{k+1}^i = \mathbf{F}_k \hat{\xi}_k^i$, and

$$\hat{\mathbf{P}}_{k+1}^i = \begin{cases} [[\hat{\mathbf{P}}_{k+1|k}^i]^{-1} + \mathbf{S}_{k+1}^i]^{-1} & \text{if measurement available} \\ \hat{\mathbf{P}}_{k+1|k}^i & \text{otherwise} \end{cases} \tag{6.3}$$

where $\hat{\mathbf{P}}_{k+1|k}^i = \mathbf{F}_k \hat{\mathbf{P}}_k^i \mathbf{F}_k^T + \mathbf{Q}_k$, $\mathbf{S}_{k+1}^i = \mathbf{H}_{k+1}^T [\mathbf{R}_{k+1}(\hat{\xi}_{k+1}^i, s_{k+1})]^{-1} \mathbf{H}_{k+1}$ and $s_{k+1} = \Psi(s_k, u_k)$ (Ψ is defined in the next subsection). The nominal error covariance matrix $\hat{\mathbf{P}}_{k+1}^i$ in (6.3) is dependent upon the availability of future observations. Because of the uncertainty of future observations we can check for target observability by guessing the target location at time $k+1$ as $\hat{\xi}_{k+1}^{i,pos}$ and checking its line of sight from the sensor location (s_{k+1}^{pos}), where $\hat{\xi}_{k+1}^{i,pos}$ is the ith target nominal track-state at time $k+1$. We can write the cost function, defined as the mean-squared error between the targets and the tracks, as: $c(\hat{b}_k, u_k) = \sum_{i=1}^{N_{\text{targs}}} \text{Tr}\hat{\mathbf{P}}_{k+1}^i$. We want to find the sequence of actions $(u_0, u_1, ..., u_{H-1})$ that minimizes the cumulative cost function (truncated horizon [50]) $J_H(b_0) = \sum_{k=0}^{H-1} \sum_{i=1}^{N_{\text{targs}}} \text{Tr}\hat{\mathbf{P}}_{k+1}^i$, where $\hat{\mathbf{P}}_{k+1}^i$ represents the ith targets nominal error covariance matrix at time $k+1$. A "receding horizon control" approach is adopted, which optimizes the action sequence for **H** time-steps from the current time-step but only the action corresponding to the current time-step is implemented. At the next time-step, we perform a new action sequence optimization for **H** time-steps.

The measurement error, w_k in (1), is normally distributed $N(0, \mathbf{R}_k(\chi_k, s_k))$, where $\mathbf{R}_k$ contains q-rad angular uncertainty and $p\%$ range uncertainty. If the distance between the sensor and the

target at time k is r_k, then the standard deviations corresponding to the range ($\sigma_{\text{range}}(k)$) and the angle ($\sigma_{\text{angle}}(k)$) are $\sigma_{\text{range}}(k) = (p/100)r_k$ and $\sigma_{\text{angle}}(k) = qr_k$. The information matrix depends on the inverse of the measurement covariance matrix, which depends on the distance between the sensor and the target. Therefore, the information matrix blows up when the UAV is exactly on top of the target (i.e., when $r_k = 0$ the sensor's location overlaps with the target's location in our 2-D environment). To address this problem, we define the effective distance (r_{eff}) between the sensor and the target as follows: $r_{\text{eff}}(k) = \sqrt{r_k^2 + b^2}$, where r_k is the actual distance between the target and the sensor and b is some non-zero real value. Therefore, the standard deviations of the range and the angle are given by $\sigma_{\text{range}}(k) = (p/100)r_{\text{eff}}(k)$ and $\sigma_{\text{angle}}(k) = qr_{\text{eff}}(k)$. If α_k is the angle between the target and the sensor at time k, then $\mathbf{R}_k$ is calculated as follows:

$$\mathbf{R}_k = M_k \begin{bmatrix} \sigma^2_{\text{range}}(k) & 0 \\ 0 & \sigma^2_{\text{angle}}(k) \end{bmatrix} M_k^T \qquad \text{where}$$

$$M_k = \begin{bmatrix} \cos(\alpha_k) & -\sin(\alpha_k) \\ \sin(\alpha_k) & \cos(\alpha_k) \end{bmatrix}.$$

The eigenvalues of the matrix $\mathbf{R}_k$ are therefore $\sigma^2_{\text{range}}(k)$ and $\sigma^2_{\text{angle}}(k)$.

6.3 UAV KINEMATICS AND FIXED FOV

Simplified UAV kinematics are used for calculating the UAV motion. Using a fixed FOV sensor, the FOV can be calculated in camera axes. Using Euler angle transformations, we can determined if the target is within the sensor FOV.

6.3.1 UAV Kinematics

In this subsection we define the mapping function Ψ introduced in Section III to describe the evolution of the sensor (UAV) state given an action, i.e., $s_{k+1} = \Psi(s_k, u_k)$. The state of the ith UAV at time k is given by $s_k^i = (p_k^i, q_k^i, V_k^i, \theta_k^i)$, where (p_k^i, q_k^i) represents the position coordinates, V_k^i represents the speed, and θ_k^i represents the heading angle. Let a_k^i be the forward acceleration

(control variable) and ϕ_k^i be the bank angle (control variable) of the UAV, i.e., $u_k^i = (a_k^i, \phi_k^i)$. The mapping function Ψ can be specified as a collection of simple kinematic equations that govern the UAV motion. The kinematic equations of the UAV motion are as follows. The speed is updated according to $V_{k+1}^i = [V_k^i + a_k^i T]_{V_{\text{min}}}^{V_{\text{max}}}$, where $[v]_{V_{\text{min}}}^{V_{\text{max}}} = \max\{V_{\text{min}}, \min(V_{\text{max}}, v)\}$, where V_{min} and V_{max} are the minimum and the maximum limits on the speed of the UAVs. The heading angle is updated according to $\theta_{k+1}^i = \theta_k^i + (gT\tan(\phi_k^i)/V_k^i)$, where g is the acceleration due to gravity and T is the length of the time-step. The position coordinate are updated according to $p_{k+1}^i = p_k^i + V_k^i T\cos(\theta_k^i)$ and $q_{k+1}^i = q_k^i + V_k^i T\sin(\theta_k^i)$.

6.3.2 Fixed FOV Calculation

Given a sensor width (x_{sens}) and height (y_{sens}), along with the focal length (f) of an installed sensor, a FOV can be calculated for a given altitude (z). The angular FOV for the sensor width is $\text{FOV}_w = 2\tan^{-1}(x_{\text{sens}}/2f)$ and for the sensor height is $\text{FOV}_h = 2\tan^{-1}(y_{\text{sens}}/2f)$. Given a height above the ground, the edges of the FOV can be calculated in camera axes. We assume that the camera is installed concurrent to the body axis, for ease of calculation. The top and bottom edges of the FOV are $x_{FOV} = \pm\text{z}\ \tan(0.5\text{FOV}_h)$ and the left and right edges of the FOV are $y_{FOV} = \pm\text{z}\ \tan(0.5\text{FOV}_w)$, where z is the altitude of the sensor.

To determine if an observation is made, the position of the target must be translated and rotated into the UAV body axis. A vector η is calculated between the UAV position and the target position as $\eta = (p_{\text{UAV}} - p_{\text{target}}, q_{\text{UAV}} - q_{\text{target}}, \text{z}_{\text{UAV}} - \text{z}_{\text{target}})$. Given the UAV Euler angles, pitch (θ_{UAV}), roll (ψ_{UAV}), and heading (ϕ_{UAV}) a rotation of η can be made between the world axis and body axis. The value of η in the body axis can now be compared to the FOV for the sensor. If the position of the target, η, is within the field of view, an observation is made and the tracker will update normally with an updated observation, as specified in (1). However, if η is not in the FOV, there will be no observation and no measurement will be passed to the tracker.

6.4 Fuel Burn Cost Function

The nominal cost function often used for vehicle control is focused purely on the estimated target location error, based on the trace of the covariance P_k. This simple cost function results in significant dynamics of the UAV to try to precisely track the target for as accurate of a target location estimate as possible. However, as a result, the increased dynamics results in less endurance. There are times during missions that maintaining observations of the target is important, but the accuracy of the estimated location is less of a concern. In these cases, increasing the mission endurance of the UAV by decreasing the dynamics becomes a viable option.

Depending on the level of desired tracking accuracy versus the increased mission endurance, the cost function can be updated to include a fuel burn estimate. A fuel-burn **B** can be included in the cost function, with a set of ratio variables, λ_1 and λ_2, to vary the impact of the error compared to the fuel burn by:

$$J_H(b_0) = \sum_{k=0}^{H-1} (\sum_{i=1}^{N_{\text{targs}}} \lambda_1 \text{Tr}\hat{\mathbf{P}}^i_{k+1}) + \lambda_2 \mathbf{B} \tag{6.4}$$

The trace of the estimated covariance is calculated for all targets across the horizon of concern and the fuel burn is calculated by estimating the power usage of the UAV during the horizon.

6.5 Weighted Trace Penalty

When using a limited FOV sensor there are conditions when the UAV may not be able to observe the target for an extended period of time. As a result, the POMDP algorithm may be unable to determine an adequate path plan, due to extended non-observation across the limited horizon. This can be exacerbated when trying to limit maneuvering for fuel efficiency. To determine a sufficient path we incorporate a *Weighted Trace Penalty (WTP)* term into the calculation of the next best UAV command. The WTP term was previously developed as an estimated cost to go (ECTG) term to deal with occlusions preventing observations [50]. We can estimate this growth with a WTP term, which is a product of the current covariance trace and the minimum distance to

observability (MDO) for the non-observed target. The terminal cost or ECTG term using the WTP takes the form:

$$\hat{J}(b) = J_{\text{WTP}}(b) := \gamma D(s, \xi^{\boldsymbol{q}}) \text{Tr} P^{\mathbf{q}} \tag{6.5}$$

where γ is a positive constant, $\boldsymbol{q}$ is the index of the worst non-observed target. The MDO, $D(s, \xi)$, is the distance from the UAV to the closest point where the target becomes observable. The addition of the WTP results in a final cost function of:

$$J_H(b_0) = \sum_{k=0}^{H-1} (\sum_{i=1}^{N_{\text{targs}}} \lambda_1 \text{Tr} \hat{\mathbf{P}}_{k+1}^{i}) + \lambda_2 \mathbf{B} + \gamma D(s, \xi^{\boldsymbol{q}}) \text{Tr} \mathbf{P}^{q} \tag{6.6}$$

6.6 Experiments

Statistical analysis, based upon 500 Monte Carlo simulations per configuration, was performed to evaluate the effectiveness of our method to track the target with varying fuel efficiency. The number of targets tracked, the altitude of the UAV, and the burn ratio of the cost function were all varied for the experiment. A single UAV tracked either one or two targets. Previous research with a fixed FOV sensor configuration has shown that above 1000 meters the impact of the sensor FOV is minimal. Therefore, only three altitudes were evaluated: 200, 500, and 1000 meters. A burn ratio (tracking accuracy:fuel burn, $\lambda_1 : \lambda_2$) of 1:0, 1:1, 1:5, 1:10, and 1:15 were used for single and two target tracking. Additionally, a burn ration of 1:20 was also performed for single target to evaluate whether the burn savings had reached a minimum yet. A range uncertainty of 10 percent and an angular uncertainty of 1 percent were used for this evaluation. The sensor was fixed in a 90 degree straight down FOV configuration, relative to the UAVs nose. A horizon window of 6 steps, at 2 seconds per step, was used for the trajectory calculation. The trajectory calculation was performed every 4 seconds. A 400 second duration was used for each Monte Carlo simulation.

The sensor simulated was modeled after several commercially available sensors [65]. The sensor was assumed to have a focal plane width of 5 mm, height of 3 mm, and a lens focal length of 5.3 mm. This configuration provides a field of view of 50.5 degree wide and 31.6 degrees tall.

It is assumed at the altitudes tested, the sensor is capable of resolving the target sufficiently for tracking.

The targets moved at a constant speed of 10 m/s at a constant heading (unknown to the tracker). The UAV was allowed to vary speed between 16 and 25 m/s and a maximum bank angle of 30 degrees. To provide a consistent analysis baseline, altitude was only considered for variation of sensor field of view; otherwise the problem was treated as a 2-D problem. Tracking errors were not considered as a function of altitude, only of x-y distance variations to enable FOV comparison. Previous efforts have shown that the average location error was 2-3 meters for single target and 5-6 meters for multiple target tracking in similar scenarios [64].

A basic fuel-burn calculation based upon actual power usage of a small UAV was used [80]. Measurements of real-time energy usage (in watt-hr) from a fixed-speed commercially available small UAV were taken while the vehicle was flying constant altitude and either wings level or in a full bank. Data showed that the power usage between wings level and full bank were essentially linear. For simplicity, it is assumed that the aircraft is symmetric such that left or right bank angle require the same energy. The burn was estimated across the speed range to extend from just a fixed speed aircraft. Table 6.1 provides simple lookup values for interpolating fuel burn.

Table 6.1: Fuel Burn Values

Velocity (m/s)	Wings Level (Watt-hr)	30 deg Bank (Watt-hr)
14	95	110
20	105	120
25	112	135
30	119	145
35	130	160

6.6.1 Results

Overall, the algorithm showed reasonable ability to track moving targets while providing fuel-efficient options. As previously seen, at lower altitudes, the tracking errors are generally higher due to reduced observations. The modified cost function provided the ability to increase endurance by 6-9% while still providing tracking ability. The increase in tracking errors due to the change in fuel burn were still reasonable for general tracking of a moving target that doesn't require high quality accuracy.

Based on the fuel usage model of the UAV, if the vehicle were to fly straight and level (most efficient possible) at a speed of 14 m/s for a 400 second window, the UAV would use 10.56 W, or 95 W for an hour. At 35 m/s, that would result in 14.4 W for 400 seconds, or 130 W for an hour. However, because the vehicle must maneuver, an increase in power usage is experienced.

The average tracking errors and fuel burn were calculated for each altitude and burn ratio configuration. Figure 6.1 provides the comparison of mean tracking error and mean fuel burn for a 400 second simulation for a single target. Figure 6.2 shows the mean percent of observations for each configuration compared to the mean location error. The symbols indicate which burn ratio was used, and the line connected to the given symbol indicate the altitude for the simulation. The trace:burn ratio provides the values of $\lambda_1 : \lambda_2$. As can be seen, as the fuel-burn ratio is increased, the fuel consumption is reduced while the tracking error is increased. As can be seen at all three altitudes, the fuel-burn ratio reduces the overall fuel use while showing an increase in tracking errors, as expected.

Figure 6.3 shows the mean tracking error versus mean fuel burn for two target tracking. Figure 6.4 shows the mean percent observations versus mean tracking error. The mean location errors and mean observation percentages are the average of all 500 runs for both targets. Only 500 and 1000 meter configurations are shown as the 200 meter data showed inadequate performance of tracking the second target. Over half of the runs at 200 meters indicated the UAV never made an observation of the second target. As can be seen, an increase in tracking error, corresponding to

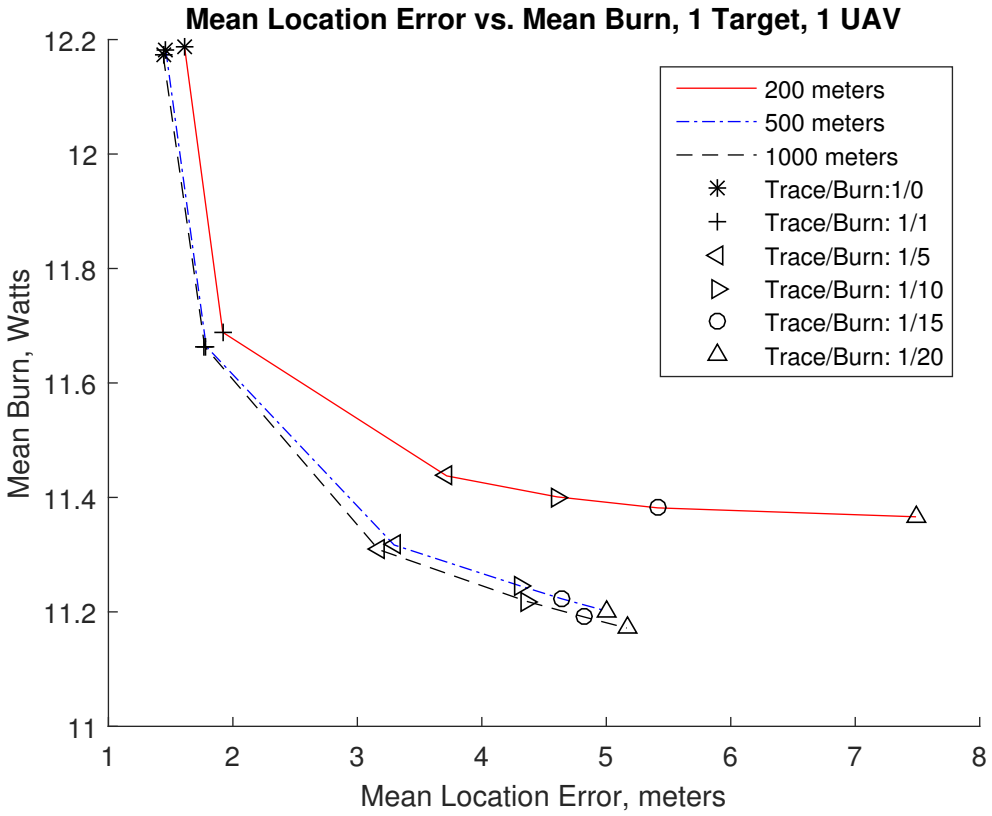


Figure 6.1: Average Location Error vs Fuel Burn, 1 UAV, 1 Target

reduced observations, can be seen as we increase the fuel burn. At 1000 meters it can be noted that the change in fuel burn has a minimal impact on mean tracking errors.

While the 500 and 1000 meter tracking showed good performance, the 200 meter errors with two targets was inadequate. One source of the poor performance is due to the MATLAB fmincon optimization function finding a local minimum and not the global minimum. This generally forces the UAV to stay closer to the first target. Additional minimization options and heuristics will be investigated for future development. Finally, due to the limited FOV of the sensor, the best way to improve the performance would be to either incorporate a sensor with a larger FOV or incorporate a gimbaled control of the sensor.

As discussed earlier, the fuel burn flying a pure straight and level 400 second flight would be 10.56 W. Tables 6.2 and 6.3 show the burns for single and two target tracking for each burn configuration. As can be seen, for single target tracking, an approximately 7-9% increase in fuel

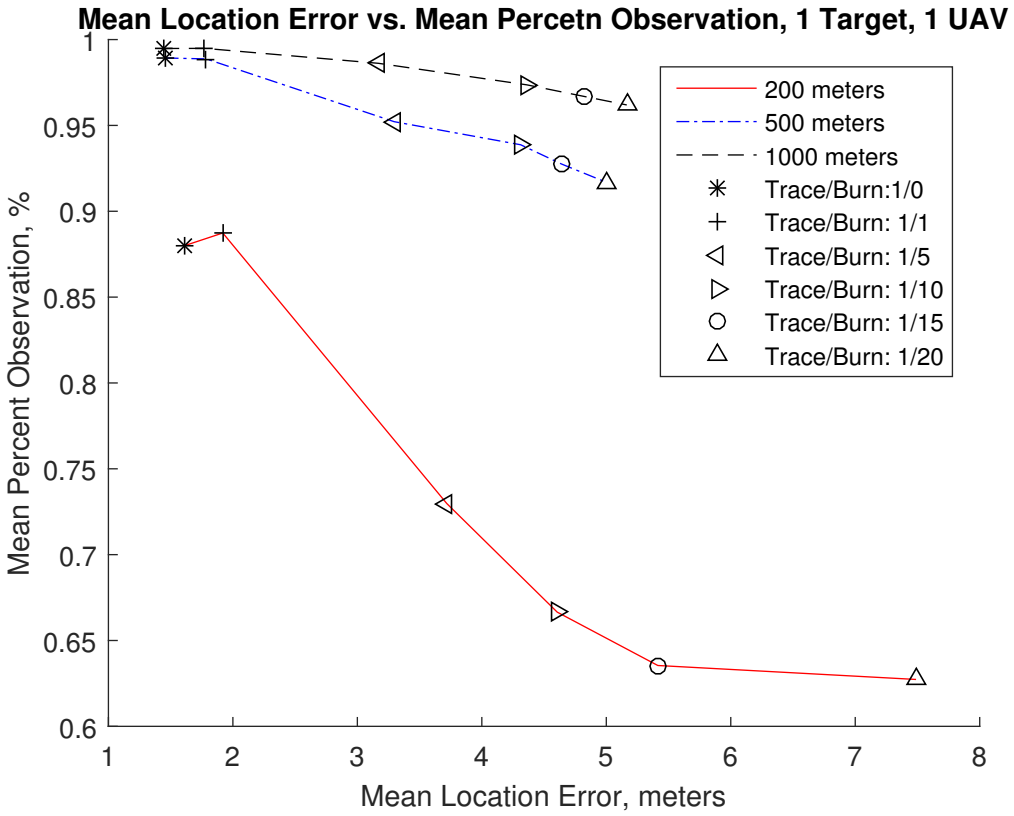


Figure 6.2: Average Location Error vs. Percent Observation, 1 UAV, 1 Target

Table 6.2: Average Fuel Burn, 1 target

$Ratio$	$200\ m$	$500\ m$	$1000\ m$
$1:0$	12.19	12.18	12.17
$1:20$	11.37	11.2	11.17
$\%gain$	7.2%	8.75%	8.95%

efficiency is noted from the 1:0 configuration as compared to the 1:20 configuration. As the altitude increases, the savings increase due to the increased sensor footprint allows reduced maneuvering while still gaining observations. For two target tracking, an approximately 6-8% increase in fuel efficiency is noted from the 1:0 configuration as compared to the 1:15 configuration. As with single target, increased altitude increases the fuel burn due to sensor footprint.

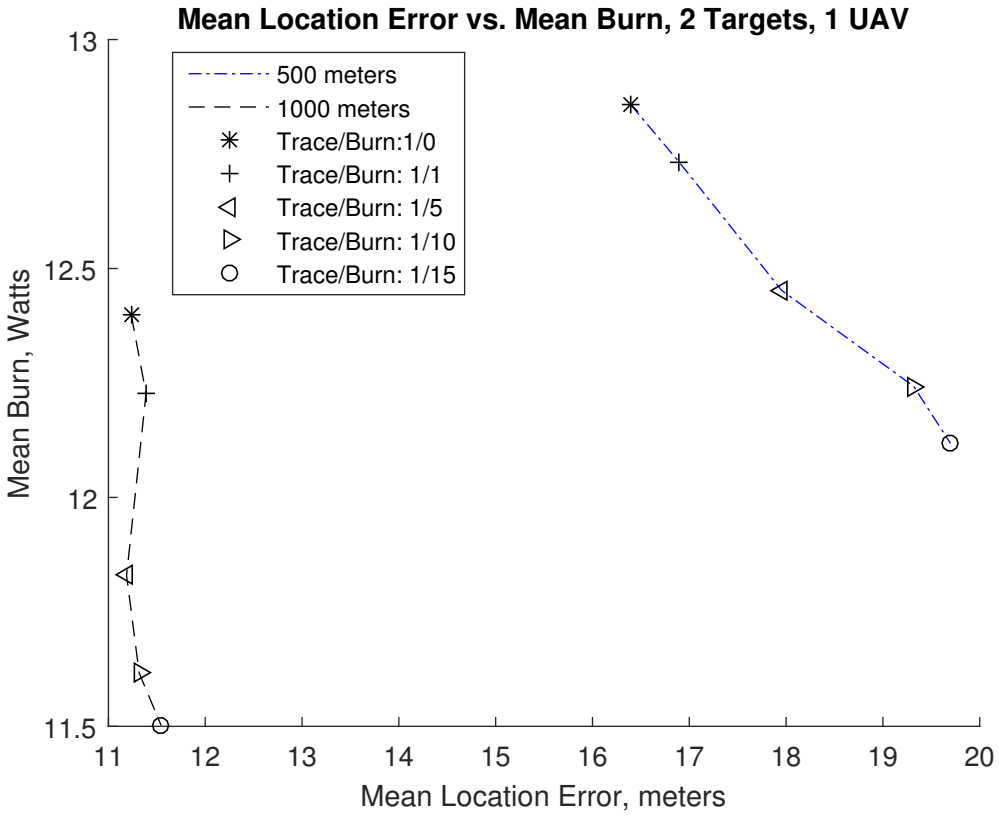


Figure 6.3: Average Location Error vs Fuel Burn, 1 UAV, 2 Targets

Table 6.3: Average Fuel Burn, 2 targets

$Ratio$	$500\ m$	$1000\ m$
$1:0$	$12.86W$	$12.40W$
$1:15$	$12.12W$	$11.5W$
$\%gain$	6.1%	7.8%

6.7 Conclusions

The fuel efficient POMDP presented provides a robust tracking solution that enables fuel efficient options. Highly accurate tracking with a limited FOV sensor flying at low altitude is a difficult use case for UAVs. Significant maneuvering is required for a UAV to provide high quality tracking with minimal errors but leads to reduced endurance. Using a fuel-burn component in the cost function of the POMDP, alongside a Weighted Trace Penalty provides a highly capable tracker with improved endurance opportunities. Endurance increases of 6-9% were experienced while still

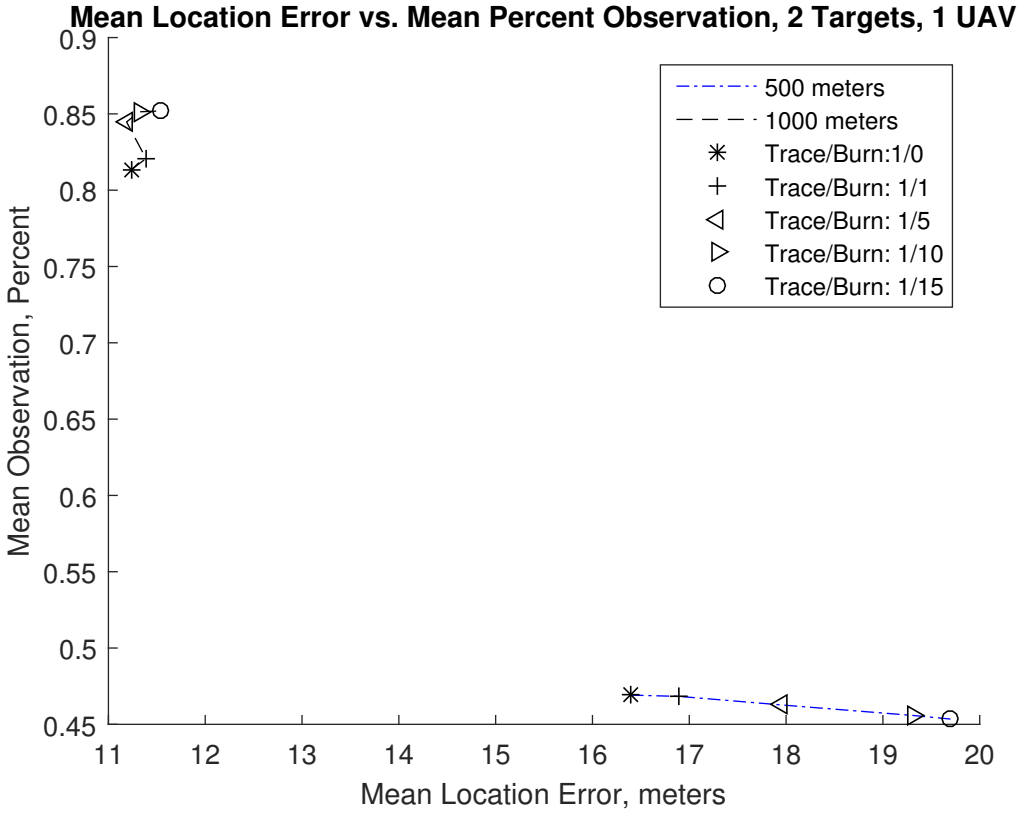


Figure 6.4: Average Location Error vs. Percent Observation, 1 UAV, 2 Targets

providing useful tracks of moving targets. Improvements in lower-altitude tracking of multiple targets may be seen by incorporating a gimbaled sensor.

Chapter 7

Increasing Fuel Efficiency for Target Tracking via Lookahead Control and Camera Articulation

7.1 Introduction

This chapter focuses primarily on the unification of Chapters 5 and 6. The formulations in these chapters are highly related; UAV's equipped with camera sensor platforms are employed to track multiple targets. Both control perspectives are also related through the application of the NBO approximation. The lookahead capabilities offer advantages in Chapter 5 with respect to the decreased gimbal motion capabilities and in Chapter 6 by accounting for the future impact of fuel consumption. Our specific integration of these two investigations loosens the restriction on the gimbal movement, placing the primary impact of the lookahead capabilities in the realm of the fuel consumption of the UAV platforms.

During the research conducted for Chapter 6 it was noted that the target tracking performance was lacking at low altitudes for multiple target cases. This is directly related to the reduced size of the camera FOV. The reduced FOV forces the UAV paths to incorporate higher speeds, tighter and more frequent turns in order to obtain even a limited number of target observations. The fuel efficiency and tracking are both negatively impacted. This is exacerbated when the targets outnumber the sensor platforms and are widely separated. Although adding a steerable gimbal to the camera sensor does not directly increase the area coverage of the camera FOV, it does allow quicker maneuverability of the focus of the camera. Lacking a gimbal, a UAV must spend more time traveling between locations at which each target is observable. The addition of the gimbal mounted camera alleviates this shortcoming through the added degrees of freedom, allowing the camera FOV to move from target to target with a reduced impact on the UAV flight path (i.e., reduced fuel consumption) and increasing the observation frequency.

Our goal is to employ the combined technologies of the previous two chapters to track widely separated targets at various altitudes. This serves as a litmus test to overcome the obstacles realized in Chapter 6. Furthermore, we investigate the impact of lookahead control on steerable cameras with quicker steering times than those from Chapter 5. Lastly, we intend to identify the approximate limits at which the balancing of target tracking and fuel efficiency reach diminishing returns.

The remainder of the chapter is structured as follows. Section 7.2 fuses the two POMDP formulations to include the camera articulation controls for the gimbal platform and formally includes the fuel efficiency cost. Section 7.3 moves beyond the simple melding of Chapters 5 and 6, discussing the specifics of the NBO cost function. This includes the modifications of the HECTG with respect to improving UAV fuel efficiency, and augmentations to the NBO cost function to produce a more informative function manifold for action selection optimization. Finally, Section 7.4 presents an application involving two UAVs tracking three widely distributed targets and the corresponding performance analysis.

7.2 POMDP

The POMDP formulations from Chapters 5 and 6, reveal obvious commonalities. The large degree of overlap exposes the minimal modifications required for their composition. The vast majority of the POMDP component definitions are shared (i.e., UAV state, observation, and observation law etc.). For completeness and continuity we detail the entire POMDP, highlighting the incorporation the gimbal control into the fuel-efficient POMDP formulation from Section 6.2.1.

State

The POMDP state is $x_k = (S_k, \chi_k, \mathcal{F}_k)$. S_k and χ_k represent the UAV state and the target state, respectively, while $\mathcal{F}_k$ is the filter state. The UAV state can be further divided into two parts, the airframe and camera, letting $S_k = (s_k, G_k)$. The elements describing the current airframe positions, velocities, and Euler angles are characterized in s_k. The camera state, G_k encompasses the gimbal azimuth, A_k and elevation, E_k, angles, i.e., $G_k = (A_k, E_k)$.

Actions

The action, u_k retains the control input vectors for UAV acceleration, a_k, and bank angle adjustment, ϕ_k. Following from the addition of the camera gimbal angles to the state, the corresponding controls are added to the action. As in Section 5.1, α_k is the azimuth slew rate and ϵ_k is the elevation slew rate for the gimbal. Then the complete action input at time k can then be described as the set of vectors $u_k = \{a_k, \phi_k, \alpha_k, \epsilon_k\}$. Each member of the four-tuple u_k is a vector of length N_s and every element of the vectors is uniformly bounded according to the physical constraints of the airframe and gimbal. This is explicitly described, at time k, as

$$
\begin{aligned}
a_k &= \left[a_k^{(1)} \dots a_k^{(N_s)}\right]^\top, & a_k^{(i)} &\in [-a^*, a^*] \\
\phi_k &= \left[\phi_k^{(1)} \dots \phi_k^{(N_s)}\right]^\top, & \phi_k^{(i)} &\in [-\phi^*, \phi^*] \\
\alpha_k &= \left[\alpha_k^{(1)} \dots \alpha_k^{(N_s)}\right]^\top, & \alpha_k^{(i)} &\in [-\alpha^*, \alpha^*] \\
\epsilon_k &= \left[\epsilon_k^{(1)} \dots \epsilon_k^{(N_s)}\right]^\top, & \epsilon_k^{(i)} &\in [-\epsilon^*, \epsilon^*],
\end{aligned}
$$

where N_s is the number of UAVs.

Observation and Observation Law

Within the POMDP state the UAV state (airframe and gimbal), and the filter state are completely observable. Therefore, their respective observation laws are simply the identity functions. The targets' states, however, can only be measured when its position, χ_k^{pos}, is included within the camera FOV ground projection. An observation is the set of all target measurements collected at time k, $Z_k = \{z_k^1, \dots, z_k^{N_m}\}$, where N_m is the total number of measurements. Each measurement is generated according to the observation law,

$$
z_k^\chi = \begin{cases} \chi_k^{pos} + w_k, & \text{if target is visible} \\ \text{no measurement}, & \text{otherwise} \end{cases}, \tag{7.1}
$$

where χ_k^{pos} is the true target position and $w_k \sim \mathcal{N}\left(0, \mathbf{R}\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right)\right)$ is random measurement noise and

$$\mathbf{R}\left(s_k^{\text{pos}}, \chi_k^{\text{pos}}\right) = \mathbf{G}(\rho_k) \begin{bmatrix} .1 \times r_k & 0 \\ 0 & .1\pi \times r_k \end{bmatrix} \mathbf{G}(\rho_k)^\top. \tag{7.2}$$

The variables s_k^{pos}, r_k, and $\mathbf{G}\left(\rho_k\right)$ are the two-dimensional sensor position, the range from the sensor to the target location and the counter-clockwise rotation matrix of angle ρ, respectively.

State Transition Law

The UAV state progresses according to the mapping

$$s_{k+1} = \Psi(s_k, u_k), \tag{7.3}$$

which describes the UAV kinematics given the control inputs defined in Section 6.3.1. The target movement is assumed to be that of a nearly constant velocity (NCV) motion model [63]. For the jth target,

$$\chi_{k+1}^j = f(\chi_k^j) + \nu_k, \tag{7.4}$$

where $\nu_k \sim \mathcal{N}\left(0, \mathbf{Q}_k\right)$ is independent and identically distributed random noise. The camera gimbal angles are updated by two additive equations; for the ith camera gimbal azimuth progresses as $A_{k+1}^{(i)} = A_k^{(i)} + \Delta_k \alpha_k^{(i)}$ and the elevation for time $k+1$ is similarly updated by $E_{k+1}^{(i)} = E_k^{(i)} + \Delta_k \epsilon_k^{(i)}$, where Δ_k is the length of the kth time-step. Lastly, the filter state is updated according to $\mathcal{F}_{k+1} = K\left(\mathcal{F}_k, z_{k+1}\right)$ such that $K\left(\cdot,\cdot\right)$ describes the chosen "tracking" filter. Here $K\left(\cdot,\cdot\right)$ is the standard Kalman filter and thus the filter state can be succinctly described by the means and covariance matrices, i.e., $\mathcal{F}_k = (\xi_k, \mathbf{P}_k)$.

One-step Cost

Following the nature of our study, the one-step cost is composed of the expected mean squared tracking error and the fuel consumption term, the *burn cost*, $B\left(s_k, u_k\right)$. These two components are

combined to form the one-step cost via the weighted sum

$$C(x_k, u_k) = \lambda_1 E_{\nu_k, w_{k+1}}[\|\chi_{k+1} - \xi_{k+1}\|^2 | x_k, u_k] + \lambda_2 B\left(s_k, u_k\right). \tag{7.5}$$

The variables λ_1 and λ_2 are constants defining the respective weights. The burn cost is not included inside the expectation as it is calculated with the UAV state and current actions choice which do not include any randomness.

7.2.1 Belief State

The standard approach to solving a problem posed as a POMDP is to formulate the corresponding belief-state Markov decision process. This is done by defining the belief state as the posterior distribution of the underly states conditioned on the action and observation history,

$$b_k(X) = P_{X_k}(X|\mathbf{Z}_0, ..., \mathbf{Z}_k; u_0, ..., u_{k-1}).$$

The belief state can be defined in terms of the original POMDP state components, $b_k(X) = (b_k^S, b_k^\chi, b_k^\mathcal{F})$. Since the UAV and filter states are fully observable their corresponding belief state components (distributions) are simple delta functions, $b_k^S = \delta(S - S_k)$ and $b_k^\mathcal{F} = \delta(\mathcal{F} - \mathcal{F}_k)$. These fully observable component then simply follow the state updates of the POMPD. The target belief state, $b_k^\chi = P_{\chi_k}(\chi|\mathbf{Z}_0, ..., \mathbf{Z}_k; u_0, ..., u_{k-1})$, is updated according to Bayes' law. For the ith target we assume that $b_k^{\chi^{(i)}} \sim \mathcal{N}\left(\xi_k, \mathbf{P}_k\right)$. This Gaussian assumption along with the assumption of perfect knowledge of observation to target associations implies that the Bayesian update required for the target belief state is simply the Kalman filter.

The actions, observations, and observation laws remain unaffected from the POMDP formulation. The one-step cost function must now be posed in terms of the belief state, opposed to the POMDP state. The transition to one-step *belief-state cost* is accomplished by the following integration,

$$c(b_k, u_k) = \int C(X, u_k) b_k(X) dX.$$

Once the conversion of the POMDP to a belief-state Markov decision process has taken place, standard MDP solution theory can be applied.

7.3 Nominal Belief-state Optimization

As Section 5.2 details the NBO motivation and theory, only a brief recap is included here for completeness. Recall that finding a "solution" to the POMDP equates to finding a mapping from the belief states to actions which minimizes the cumulative cost,

$$V_H(b_0) = E\left[\sum_{k=0}^{H-1} c\left(b_k, u_k\right) \mid b_0\right], \tag{7.6}$$

over some horizon H. This mapping is the *optimal policy*. Bellman's principle poses the problem equivalently as,

$$V_H^*\left(b_0\right) = \min_u \left\{c\left(b_0, u\right) + \mathrm{E}\left[V_{H-1}^*\left(b_1\right) | b_0, u\right]\right\},$$

exemplifying the implicit recursive nature of the problem; optimize the current action cost plus the expected cost-to-go. Now the optimal cumulative cost, $V_H^*\left(b_0\right)$ and the optimal policy, $\pi^*\left(b_0\right)$, can be expressed concisely as,

$$\begin{aligned} V_H^*\left(b_0\right) &= \min_u Q_H(b_0, u) \\ \pi_k^*\left(b_k\right) &= \arg\min_u Q_H\left(b_k, u\right), \end{aligned}$$

where $Q_H(b_0, u) = C(b_0, u) + \mathrm{E}\left[V_{H-1}^*\left(b_1\right) \mid b_0, u\right]$ is the Q-value over horizon H. Fixing horizon H as a constant and solving for $\pi^*\left(b\right)$ given the current belief state b is *receding horizon control*. This is a convenient way to reduce computations by assuming that the impact of a current action is negligible after some time horizon H.

While the truncated horizon from the receding horizon control approach saves on computational efforts, the propagation of a posterior distribution. Also, as mentioned in the preceding section, our assumptions lead directly the use of the Kalman filter for the belief-state update calcu-

lations. But, the calculations of the belief-state cost (7.6) still require significant computations due to the expectation. NBO lessens this burden through the approximation of the posterior distribution by defining a *nominal belief state* as a single sample of the distribution. Under our assumptions we choose this sample to be the MAP estimate provided by the nominal filter state, $\left(\hat{\xi}_k, \hat{\mathbf{P}}_k\right)$. The nominal belief state then progresses according to the Kalman filter equations with exactly zero noise. The corresponding equations are seen in Section 5.3.1.

7.3.1 NBO Cost function

Given the nominal belief state and update definitions, then the cumulative belief state cost can be approximated as

$$V_H\left(b_0\right) \approx \sum_{k=1}^{H} c\left(\hat{b}_k, u_k\right),$$

eliminating the expectation by inserting the nominal belief state for the belief state in (7.6). The optimization for any time k can then be posed as

$$V^*\left(b_k\right) = \min_{(u_k)_j} \sum_{j=1}^{H} = c\left(\hat{b}_{k+j}, u_{k+j}\right)$$

for the fixed horizon length of H optimizing over the set of actions $\{u_i\}_{i=k}^{H+k}$.

The addition of the gimbal control elements to the POMDP in Section 7.2 constitute corresponding modifications to the solution approach. The overall NBO theory and application remain the same, but there are adjustments and additions to the calculations involved in the NBO cost. We build from the canonical form of NBO cost,

$$V_H(b_0) = \sum_{k=0}^{H-1}(\sum_{i=1}^{N_{\text{targs}}} \lambda_1 \text{Tr}\hat{\mathbf{P}}_{k+1}^{i}) + \lambda_2 \mathbf{B},$$

which accounts for the expected mean-squared tracking error and the burn cost, mirroring one-step cost defined in (7.5). The expected mean-squared tracking error is equivalently $\text{Tr}\hat{\mathbf{P}}_{k+1}^{i}$ in terms of

the belief state and **B** represents the analytically calculated burn cost based on the nominal belief state sequence.

The following subsections blueprint the remaining heuristics and details of the final NBO cost function. The first augmentation to the NBO cost is the heuristic expected cost-to-go (HECTG), $\mathcal{T}'(b_H)$. The HECTG is a mildly modified version from Section 5.3.3. Next, the NBO cost is further extended to accommodate hard physical bounds in both the airframe and the gimbal. Lastly, a cost component which represents the positioning of the target in the camera FOV is included to compensate for the variances inherent to the belief-state estimates. This *pixel distance penalty* pertains only to the gimbal camera platform.

7.3.2 HECTG

From our previous research included in Chapters 5 and 6, specifically the results in Section 5.4.2, it is clear that simple truncation of the lookahead horizon lacks complete information of the long-term impact of a current action selection. Insight can be further garnered by the addition of a heuristic cost-to-go term. Chapter 6 employs a cost function based solely on the minimum distance to observation (MDO). Chapter 5 implements a more custom-fit approach based on airframe the combined flight time, turning time, and gimbal-adjust time for a UAV to reach a point of observation to maximally diminish the expected target tracking error. We propose a similar HECTG for the fuel-efficient target tracking with gimbaled cameras.

Section 7.1 speaks to how the relaxation of gimbal slew rate restrictions directly benefit the fuel-efficiency of a UAV tracking multiple targets. To this point, we eliminate the gimbal-steering time from the HECTG in Section 5.3.3. Camera gimbals with higher slew rates are capable of switching between targets with less lead time for adjustment. Furthermore, the increased steering speed nearly eliminates the impact of the gimbal component in the previous HECTG. This leads to an HECTG

$$\tilde{V}(b_H) = \gamma \mathcal{T}'(\hat{b}_H) \text{Tr}\, \hat{\mathbf{P}}_H^q$$

where the q is the target index who has the worst tracking error of all unobserved targets at the end of the truncated lookahead horizon H. The constant γ is a scaling factor and $\mathcal{T}'\left(\hat{b}_H\right) = T_1 + T_2$. The linear flight time is

$$T_1 = \frac{\|\hat{s}_H^{pos} - e_{xy}^q\|_2}{\mathcal{V}^+}$$

and the UAV heading correction time is

$$T_2 = \frac{\left|\tan^{-1}\left(\frac{\hat{s}_{\dot{y}}}{\hat{s}_{\dot{x}}}\right) - \tan^{-1}\left(\frac{e_y^q - \hat{s}_y}{e_x^q - \hat{s}_x}\right)\right|}{\mathcal{V}_R^+}$$

based on the navigation from the sensor position $\hat{s}_H^{\text{pos}} = (s_x, s_y)$ to the *optimal point of observation* of an unobserved target; the desired location for observation, $e_{xy}^q = (e_x^q, e_y^q)$ is a point a fixed distance from the targets nominal belief state mean, $\hat{\xi}_H^q$ lying on the eigenvector of $\hat{\mathbf{P}}_H^q$ with the maximum eigenvalue. The denominators V^+ and V_R^+ are the maximum UAV linear velocity and radial velocity, respectively. Extending the NBO cost to include this HECTG results in

$$V_H(b_0) = \sum_{k=0}^{H-1}\Big(\sum_{i=1}^{N_{\text{targs}}} \lambda_1 \text{Tr}\hat{\mathbf{P}}_{k+1}^i\Big) + \lambda_2\mathbf{B} + \gamma\mathcal{T}'(b_H)\text{Tr}\hat{\mathbf{P}}^q. \tag{7.7}$$

7.3.3 Optimizing with Secondary Constraints

The current NBO cost in (7.7) includes for tracking error, burn cost and a HECTG, but lacks explicit accountability for the gimbal mounted camera and airframe constraints beyond those of the action bounds, $\{a^*, \theta^*, \alpha^*, \epsilon^*\}$. There are secondary constraints which need to be respected, UAV speed and gimbal elevation angle. The NBO cost must reflect these fixed values in order to provide a meaningful action selection. The actions are selected by numerical optimization based on the NBO cost. Thus, the NBO cost must include thresholds for these hard bounds. When such control inputs are issued, the corresponding values of the physical UAV platform simply remain at their limits. For example, if the UAV is flying at maximum speed V^+ and an acceleration input is issued which requests the UAV to increase its speed, the UAV is incapable of complying and remains at

V^+. The UAV's obey the same thresholding for a minimum speed, V^-. A similar situation arises with the gimbal elevation angles which have hard limits, $[-\mathcal{E}, \mathcal{E}] = [-90°, 90°]$.

Implementing a simple threshold within the nominal belief state propagation only prevents the control selection from issuing invalid commands, but does not prevent the numerical search from exploring these areas of the optimization manifold created by the cost function in (7.7). This is computationally inefficient. Moreover, as with any numerical optimization, it could fall prey to local minima during this exploration, perhaps repeatedly. To avert this pitfall a penalty function, $\mathcal{P}(\hat{V}, \hat{E}) = \omega \left(\mathcal{P}_{V^+}(\hat{V}) + \mathcal{P}_{\mathcal{E}}(\hat{E}) \right)$, is imposed. Let $\hat{V} = \|[\hat{s}_{\dot{x}}, \hat{s}_{\dot{y}}]^\top\|$ be the current UAV speed and $\hat{E}$ be the current gimbal elevation based on the action u being evaluated for the NBO cost minimization. The UAV speed penalty,

$$\mathcal{P}_{V^+}(\hat{V}) = \left| \min\left(\hat{V} - V^-, 0\right) + \min\left(V^+ - \hat{V}, 0\right) \right|,$$

and the gimbal elevation penalty,

$$\mathcal{P}_{\mathcal{E}}(\hat{E}) = \left| \min\left(\hat{E} + \mathcal{E}, 0\right) + \max\left(\hat{E} - \mathcal{E}, 0\right) \right|),$$

are weighted by the constant ω. Note, the penalty factors scale according to the magnitude of the violations in an attempt to maintain smoothness in the respective dimensions of the cost function for the numerical search.

7.3.4 Pixel Distance Heuristic

We recognize that the nominal belief states are an estimate of the underlying state. Accordingly, the larger the expected tracking error, $\mathrm{Tr}\mathbf{P}$, the less accurate the location estimate $\hat{\xi}$ is prone to be. This is, however, the only information NBO is provided for action selection. This discrepancy between the real-world state the nominal belief state can severely impact tracking performance due to the limited FOV and camera aiming. For instance, if the action selection places nominal belief state target position near the boundary of the camera FOV it is plausible that the action

will result in a missed detection (null observation) if the true target location lies just beyond the projection of the camera FOV. The likelihood of this instance increases with the value of $\mathrm{Tr}\mathbf{P}$. For this reason, a *pixel distance penalty* is imposed as an additive term to the NBO cost. Through the standard series of rotational matrix multiplications and linear translations [62], the nominal target position is projected into the camera frame of reference in terms of pixel location. Given the pixel coordinate location of the ith target with respect to the jth UAV camera center of FOV, $\hat{\mathcal{S}}^{(i,j)}$, the pixel distance penalty is

$$\mathcal{P}_{\mathcal{X}}\left(\hat{\mathcal{S}}^{(i,j)}\right) = \omega_{\mathcal{X}} \min\left(\|\mathcal{S}^{(i,j)}\|_2, \mathcal{M}\right) \mathrm{Tr}\hat{\mathbf{P}}^i,$$

where $\mathcal{M}$ is the maximum distance (in pixels) at which a target is visible within the camera FOV, i.e., the distance from the center of FOV to the corner of the FOV. Again, $\hat{\cdot}$ implies values generated from the nominal belief state. Multiplying the minimum by $\mathrm{Tr}\mathbf{P}$ scales the penalty directly proportionate to expected tracking error. Note, the pixel distance penalty is only included in the steerable camera platforms in Section 7.4. The reduced degrees of freedom of a fixed camera platform would result in the pixel distance penalty favoring UAV maneuvers, conflicting the burn cost. In fact, the weight $\omega_{\mathcal{X}}$ is orders of magnitude lower than the burn-weight λ_2 to alleviate the effect, even on the gimbal equipped UAVs.

7.3.5 Culmination of Cost

Based on the attributes of our UAV target tracking scenario, we formulated the HECTG term arriving at (7.7). Beyond the HECTG, two penalty terms strengthen the informativeness of the NBO cost function. The penalty function $\mathcal{P}(\hat{V}, \hat{E})$ strives to restrict the search across the action space which results in the violation airframe or gimbal operational constraints. Secondly, the pixel distance penalty, $\mathcal{P}\left(\hat{\mathcal{S}}^{(i,j)}\right)$, motivates the aiming of the camera towards centering the FOV on the estimated target location. Then the NBO cost encompassing the supplementary heuristics is

$$V_H(b_0) = \sum_{k=0}^{H-1} \Bigg[\sum_{i=1}^{N_{\text{targs}}} \left(\lambda_1 \text{Tr}\hat{\mathbf{P}}^i_{k+1} + \sum_{j=1}^{N_{\text{UAV}}} \mathcal{P}_{\mathcal{X}} \left(\hat{\mathcal{S}}^{(i,j)}_{k+1} \right) \right) + \sum_{j=1}^{N_{\text{UAV}}} \left(\mathcal{P} \left(\hat{V}^j_{k+1}, E^j_{k+1} \right) + \lambda_2 \mathbf{B}^j_{k+1} \right) \Bigg] + \gamma \mathcal{T}'(\hat{b}_H) \text{Tr}\hat{\mathbf{P}}^q. \tag{7.8}$$

The superscripts i and j refer to the target and UAV indices, respectively. This final cost function increases the quality of the action rankings.

7.4 Empirical Study

Our experiments center around a scenario involving two UAVs tracking three targets. The target trajectories are those generated by a single random sample of an NCV model. For ease of comparison, we chose to utilize the same target trajectories from Section 5.4.4 and illustrated in Figure 5.2. The results compare track-loss percentage, tracking error, and fuel cost across fixed camera and gimbal mounted camera UAV platforms. Over the simulations presented in the remainder of this section, several aspects are explored through the variation of parameters: two different UAV altitudes, and six trace/burn ratios ($\lambda_1 : \lambda_2$) for both the fixed and gimballed camera. Furthermore, we explore two different gimbal slew-rate limits to clarify the impact of the gimbal capabilities. For each of these parameter combinations, 150 Monte Carlo simulations are evaluated to compile statistically relevant results.

7.4.1 Simulation Details

The details and parameter settings of these experiments bear close similarities to those of Chapters 5 and 6. To provide an easy reference for the reader, the aspects directly related to the ensuing results and analysis may include some repetition.

UAV The two UAV's in each simulation are considered identical platforms, having homogeneous parameters. The UAV's have upper and lower limits on their speed of 14m/s and 25m/s, respectively. The action inputs which manipulate the airframe and the gimbal are each bounded. The

longitudinal acceleration is limited to $a^* = 3\text{m/s}^2$, and the bank angle is limited to $\phi^* = 30^\circ$. The gimbal slew rates are considered equal for both the azimuth and elevation, $\alpha^* = \epsilon^*$. We examine both a slew rate of 85°/s and 60°/s. The final limit applicable to the UAV state the gimbal elevation angle limit set at $\mathcal{E} = 90^\circ$.

Camera Our simulations model the camera sensor after the Sony FCB-EV7500 Block Camera [65] with a CCD of dimensions $5\text{mm} \times 3\text{mm}$ and a focal length of 5.3mm. This results in an angular FOV with a width of 50.5° and a height of 31.6°. The camera noise model is classified by the covariance matrix in (7.2), defining a 10% uncertainty in range and .1-radian angular uncertainty.

Target Models The three targets start at $(0, 0), (0, 250)$ and $(200, 200)$ and their final locations are $(1548, 35), (1780, 446)$ and $(699, 1083)$, respectively. The target movements are randomly generated by the NCV model in (4.2). These trajectories imply that the targets travel at average speeds of 7.5m/s, 8.9m/s, and 5.1m/s. Two of the three targets travel in a nearly parallel manner while the third target diverges, creating an increasingly large target separation.

7.4.2 Performance Evaluation

The results in the following section evaluate the performance in terms of fuel efficiency and tracking error. This focus can be deduced from the POMDP one-step cost in (7.6). The tracking error is a post-processing calculation based on the recorded target location estimates over the simulation and the targets' ground-truth locations; a simple two-dimensional Euclidean distance is computed. The fuel efficiency is computed in watts based on the UAV state components, speed, and bank angle. Table 6.1 sets forth the sample values used to (linearly) interpolate the watts consumed at consecutive time steps.

Although the reduction of target tracking error and fuel consumption are the core ambitions of this study, we present another relevant tracking statistic, track-loss percentage. A track is considered lost when the current estimate of the tracking error, $\text{Tr}(\mathbf{P})$, exceeds some set threshold. Analyzing the simulation data in terms of tracking error by utilizing the true location of the targets

is a valid way to appraise the tracking competence, after the fact. In contrast, track loss is capable of providing real-time feedback about the quality of the tracking estimates. This measure is easily extracted from the filter state alongside the location estimate. From an end user's point of view, if $\mathrm{Tr}(\mathbf{P})$ grows excessively large, less credence should be given to the accuracy of the estimate. For this reason, we exercise track loss as a filter of the results before examining tracking errors. In other words, when $\mathrm{Tr}(\mathbf{P}) > \Omega$ for any target during a simulation we consider this track to be lost and therefore exclude that Monte Carlo sample from the tracking error statistics. This follows logically, as NBO operates solely on the *estimated* target locations. Once this estimate is of sufficiently poor quality, the resulting action selections have no bases to determine how to observe (i.e., track) a target due to the limited camera FOV. In fact, track loss is even used in state-machine style UAV control for target tracking, [81], to switch from target tracking to search a surveillance algorithm to relocate targets. Our study does not encompass target re-acquisition. Thus, we discard the associated tracking error data. The formal definition of track loss is found in Section 5.4.1.

Note, in the track-loss percentage results, $\Omega = 10000$ which correlates to an expected tracking error of 100m.

7.4.3 Results

We present the outcomes of the thirty-six scenarios in three tables corresponding to the track-loss percentage, Table 7.1, the average fuel consumption, Table 7.2, and the average tracking error, Table 7.3. The results are also visualized through figures contrasting the track-loss percentage and tracking error to the fuel consumption, Figure 7.1 and Figure 7.2, respectively.

Before discussing the results, it is necessary to highlight the impact of filtering the results based on track loss. The sample size of the respective averages (in the tables and figures) is the number of Monte Carlo simulations which experience zero track loss. At higher burn weights the number of samples over which the average is computed is severely decreased; the corresponding confidence intervals are large. It is with this fact in mind the reader should interpret the following results.

Some general trends emerge while examining the tables. The higher altitudes and larger slew rates provide a reduction in track loss, seen in Table 7.1. As the burn-weight, λ_2, increases there is an inverse impact on, not only track loss but also on the tracking error. The comparison between the two gimbal slew rates at either altitude shows that when λ_2 is low, the increased camera maneuverability does not play a significant role in the tracking error. When $\lambda_2 \geq 20$, the impact of the slew rate is more pronounced.

Table 7.1: Track-loss Percentage

	300m			400m		
$\lambda_1 : \lambda_2$	Fixed	Gimbal $60°$	Gimbal $85°$	Fixed	Gimbal $60°$	Gimbal $85°$
$1:0$	51.2	15.5	12.5	22.0	6.5	7.1
$1:1$	53.6	18.5	14.9	22.0	8.3	3.0
$1:10$	83.9	36.3	33.9	50.6	22.0	22.6
$1:20$	83.3	50.6	49.4	60.1	35.7	33.3
$1:30$	89.3	60.1	50.6	64.3	35.7	39.3
$1:40$	89.9	66.7	65.5	66.7	56.5	44.0

Table 7.2: Average Fuel Burn (W)

	300m			400m		
$\lambda_1 : \lambda_2$	Fixed	Gimbal $60°$	Gimbal $85°$	Fixed	Gimbal $60°$	Gimbal $85°$
$1:0$	9.32	9.35	9.38	9.37	9.40	9.45
$1:1$	8.74	8.76	8.78	8.73	8.74	8.74
$1:10$	8.48	8.38	8.34	8.46	8.31	8.31
$1:20$	8.43	8.27	8.28	8.42	8.24	8.23
$1:30$	8.41	8.26	8.26	8.38	8.20	8.20
$1:40$	8.32	8.19	8.20	8.37	8.19	8.18

Inspecting the figures reinforces the trends seen in the tables and accents the limitations of balancing the tracking error and fuel efficiency. Once $\lambda_2 > 20$ the reduction in fuel burn begins to level off. While the burn cost approaches 8.1 watts the only significant effect is the increase in both track-loss percentage and tracking error. This point of attrition is directly related to the target separation, defining the extended travel distances required to obtain target observations.

Table 7.3: Average Tracking Error (m)

	300m			400m		
$\lambda_1 : \lambda_2$	Fixed	Gimbal 60°	Gimbal 85°	Fixed	Gimbal 60°	Gimbal 85°
1 : 0	16.1	14.3	14.8	12.7	12.3	12.8
1 : 1	19.4	16.1	17.0	15.9	14.0	14.7
1 : 10	26.2	25.3	25.6	22.3	22.6	25.8
1 : 20	28.9	31.3	35.1	28.7	30.9	29.0
1 : 30	30.7	36.9	34.5	34.1	31.6	32.5
1 : 40	35.7	38.7	37.9	34.1	36.3	35.3

To quantify the benefits of the gimballed camera in terms of fuel efficiency we again reference the data associated with $\lambda_2 = 20$; the tracking error remains viable at this level. Given the fixed sensor performance with $\lambda_2 = 0$ as a basis for comparison, there is an 11% efficiency improvement at 300m altitude and a 12% reduction at the 400m. These gains are with respect to a UAV equipped with a steerable camera capable of $60°$/s adjustments.

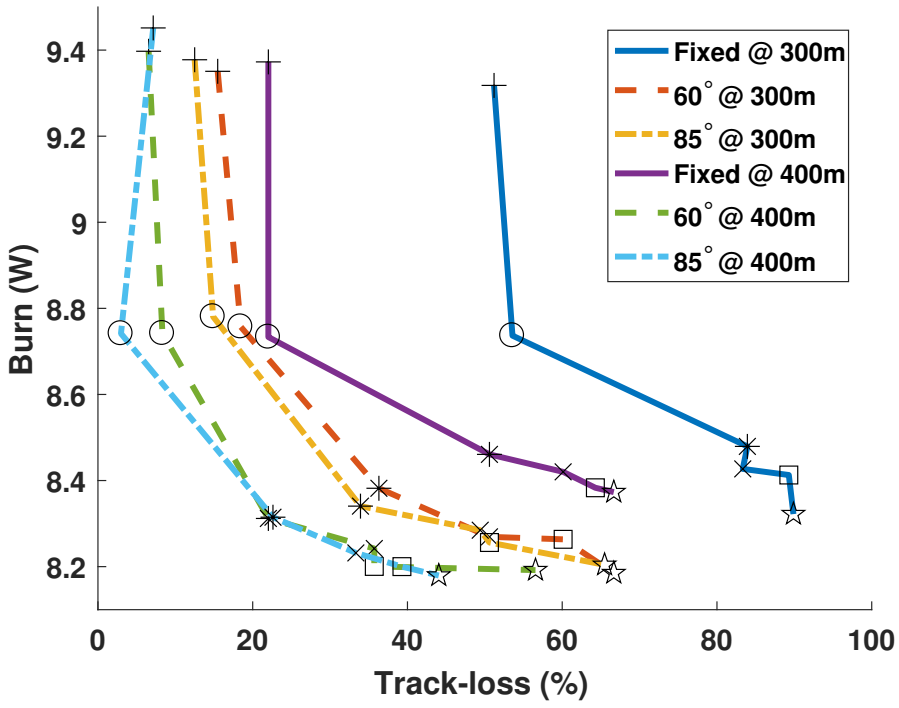


Figure 7.1: Depicts track loss percentage versus fuel consumption in watts. The symbols represent different λ_2 values; $\{+, \circ, *, \times, \square, \star\}$ correspond to $\lambda_2 = \{0, 1, 10, 20, 30, 40\}$, respectively and $\lambda_1 = 1$ for all simulations.

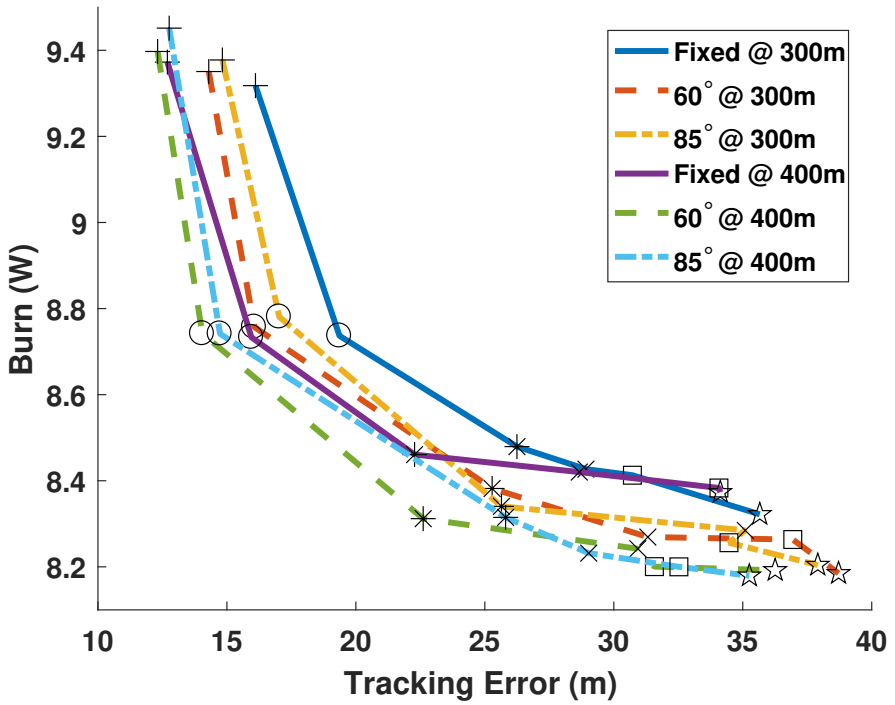


Figure 7.2: Depicts average tracking error in meters versus average fuel consumption in watts. These plots are based on only the samples with *no* lost tracks. The symbols represent different λ_2 values; $\{+, \circ, *, \times, \square, \star\}$ correspond to $\lambda_2 = \{0, 1, 10, 20, 30, 40\}$, respectively and $\lambda_1 = 1$ for all simulations.

7.5 Conclusions

The research in this chapter was heavily inspired by the results of Chapter 5 and Chapter 6. Decision control through lookahead optimization provided performance gains for both camera gimbal control and fuel efficiency. For this reason, this chapter has combined the POMDP formulations in order to extract the benefits of the steerable camera for fuel efficiency and target tracking. In response to the expanded POMDP, a custom cost function was constructed for use with nominal belief state optimization. The cost function was augmented with a HECTG term and two additional heuristics to better enable the numerical optimization in relation to the underlying states. The POMDP formulation and NBO were applied to a scenario involving two UAVs tracking three targets. The analysis of the simulation results showed decreases in tracking error and fuel efficiency gains up to 12%.

Chapter 8

Conclusions and Recommendations

We have shown the adaptability of non-myopic sensor resource management by way of POMDP theory. Chapters 2 and 3 present sensor resource management from the perspective of scheduling wave forms to maximize information gain. Chapter 2 introduces the problem of time-varying sparse-signal estimation under the impediments of noise and erasure. The time varying signal leads to a rapid increase in the dimensions of the belief state. In order co combat the impending computational issues we applied two data association algorithms from the target tracking paradigm (JPDA and MHT). This unique approach produced quality results across all four simulation scenarios. Chapter 3 extended the adaptive estimation from Chapter 2 posing a problem with increased difficulty. The action selection process includes a mandated covertness requirement, preventing repeated measurement waveforms. The non-myopic Rollout solution approached out performed greedy action selections under these constraints when combined with MHT.

Chapters 4, 5, 6, and 7 delved into another outlook of sensor resource management. These remaining chapters surveyed UAV sensor platform control for target tracking. Like the previous chapters each investigation was based on a POMDP, but applied the NBO approximation for solutions. The primary objective in each study was to minimize the mean-squared tracking error. However, competing auxiliary objectives, persistent surveillance and fuel efficiency, were also concerns in Chapters 4 and 6, respectively. Chapters 5 and 7 escalated the control complexity through the addition of steerable gimbal mounted cameras. The included results again showed the advantages of non-myopic decision control. Chapter 4 implements a GM-PHD filter, which allowed the foresight of the NBO to optimize the action selections for concurrent persistent surveillance *and* target tracking performance. Chapter 5 exemplified the advantages of longer NBO horizons when camera gimbal movements are tightly constrained due to hardware limitations. Chapter 6 augmented the NBO cost function by including UAV fuel-efficiency as an action selection directive. Fuel usage and target-tracking error were balanced though the resulting path planning choices. Chapter

7 melds Chapters 5 and 6 to benefit from the flexibility of gimbal mounted cameras to aid in the reduction of UAV fuel consumption while tracking multiple targets at low altitudes.

Sparse signal estimation through compressive sensing and target tracking with UAVs are contrasting ventures, but both ball under the purview of sensor resource management. Through rigorous POMDP formulations and the application of approximate solution techniques, namely Rollout and NBO, based on Bellman's equation, we have shown that both benefit from lookahead control in several diverse scenarios.

8.1 Recommendations

While the work presented in this dissertation has culminated in showing viable results based on lookahead action selection there are further avenues for exploration to extend the work. The two specific areas of interest are the extension of Chapter 4 to consider a more complex sensor model and to pursue the real-world implementation of the developments from Chapters 5, 6, and 7.

Incorporating a pan-tilt-zoom camera sensor model into the work of Chapter 4 could further aid the disparate goals of target tracking and simultaneous persistent surveillance. As seen in Chapter 7, the increased degrees-of-freedom provided by a gimbal mounted camera alleviated the conflict between the competing cost components; a similar effect should result from a steerable camera in the formulation of Chapter 4. Furthermore, the addition of a zoom capable camera model, while increasing the control complexity, could provide an action space well suited to the switching between the local target (increased zoom) tracking focus and the global surveillance focus (wide angle view).

Moving toward a real-wold implementation of the study from Chapters 5-7 would require an intermediary step, porting the algorithms to a higher fidelity simulation platform. This step may sound mundane, but a simulation platform such as the open source Aerospace Multi-agent Simulation Environment (OpenAMASE) developed by the Air Force Research laboratories [82] would highlight practical implementation issues which directly impact the control concerns. For example, situations such as gimbal-lock and servo-jitter concerns that would need addressed before

implementation on a physical platform. The OpenAMASE not only supplies a proven testbed for identifying the issues but serves an a stepping stone for transitioning our algorithms to any of the UAV platforms represented in the included libraries.

Bibliography

[1] R. Bellman. *Dynamic programming*. Princeton University Press, NJ, 1957.

[2] E. J. Candés and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[3] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, April 2006.

[4] E. J. Candés, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, August 2006.

[5] R. G. Baraniuk. Compressive sensing. *IEEE Signal Processing Magazine*, 24(4):118–121, July 2007.

[6] Y. C. Eldar and G. Kutyniok. *Compressed Sensing: Theory and Applications*, volume 95. Cambridge University Press, Cambridge, UK, 2012.

[7] R. A. DeVore. Deterministic constructions of compressed sensing matrices. *Journal of Complexity*, 23(4):918–925, December 2007.

[8] P. Indyk. Explicit constructions for compressed sensing of sparse signals. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, pages 30–33, San Francisco, CA, January 20-22, 2008. Society for Industrial and Applied Mathematics (SIAM).

[9] W. U. Bajwa, J. D. Haupt, G. M. Raz, S. J. Wright, and R. D. Nowak. Toeplitz-structured compressed sensing matrices. In *Proceedings of the 14th IEEE/SP Workshop on Statistical Signal Processing (SSP)*, pages 294–298, Madison, WI, August 26-29, 2007.

[10] X. Weiyu and B. Hassibi. Compressed sensing over the Grassmann manifold: A unified analytical framework. In *Proceedings of the 46th Annual Allerton Conference on Communi-*

cation, Control, and Computing (Allerton), pages 562–567, Monticello, IL, September 23-26, 2008.

[11] R. Calderbank, S. Howard, and S. Jafarpour. Construction of a large class of deterministic sensing matrices that satisfy a statistical isometry property. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):358–374, April 2010.

[12] S. D. Howard, A. R. Calderbank, and S. J. Searle. A fast reconstruction algorithm for deterministic compressive sensing using second order Reed-Muller codes. In *Proceedings of the 42nd IEEE Annual Conference on Information Sciences and Systems (CISS)*, pages 11–15, Princeton, NJ, March 19-21, 2008.

[13] M. A. Davenport, P. T. Boufounos, M. B. Wakin, and R. G. Baraniuk. Signal processing with compressive measurements. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):445–460, April 2010.

[14] Z. Ben-Haim and Y. C. Eldar. The Cramér-Rao bound for estimating a sparse parameter vector. *IEEE Transactions on Signal Processing*, 58(6):3384–3389, June 2010.

[15] A. Eftekhari, J. Romberg, and M. Wakin. Matched filtering from limited frequency samples. *IEEE Transactions on Information Theory*, to appear.

[16] E. Arias-Castro, E. J. Candès, and M. A. Davenport. On the fundamental limits of adaptive sensing. *IEEE Transactions on Information Theory*, 59(1):472–481, January 2013.

[17] S. Ji, Y. Xue, and L. Carin. Bayesian compressive sensing. *IEEE Transactions on Signal Processing*, 56(6):2346–2356, June 2008.

[18] E. Bashan, R. Raich, and A. O. Hero. Optimal two-stage search for sparse targets using convex criteria. *IEEE Transactions on Signal Processing*, 56(11):5389–5402, November 2008.

[19] R. M. Castro, J. Haupt, R. Nowak, and G. M. Raz. Finding needles in noisy haystacks. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5133–5136, Las Vegas, NV, March 30-April 4, 2008.

[20] D. Wei and A. O. Hero III. Multistage adaptive estimation of sparse signals. *IEEE Journal of Selected Topics in Signal Processing*, to appear.

[21] J. D. Haupt, R. G. Baraniuk, R. Castro, and R. D. Nowak. Compressive distilled sensing: Sparse recovery using adaptivity in compressive measurements. In *Conference Records of the 43rd IEEE Asilomar Conference on Signals, Systems and Computers*, pages 1551–1555, Pacific Grove, CA, November 1-4, 2009.

[22] D. Sejdinovic, C. Andrieu, and R. Piechocki. Bayesian sequential compressed sensing in sparse dynamical systems. In *Proceedings of the 48th IEEE Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1730–1736, Monticello, IL, September 29-October 1, 2010.

[23] C. Qiu, W. Lu, and N. Vaswani. Real-time dynamic MR image reconstruction using Kalman filtered compressed sensing. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 393–396, Taipei, Taiwan, April 19-24, 2009.

[24] W. Dai, D. Sejdinovic, and O. Milenkovic. Gaussian dynamic compressive sensing. In *Proceedings of International Conference on Sampling Theory and Applications (SampTA)*, Singapore, May 2-6, 2011.

[25] M. S. Asif, D. Reddy, P. T. Boufounos, and A. Veeraraghavan. Streaming compressive sensing for high-speed periodic videos. In *Proceedings of the 17th IEEE International Conference on Image Processing (ICIP)*, pages 3373–3376, Hong Kong, September 26-29, 2010.

[26] W. Li and J. C. Preisig. Estimation of rapidly time-varying sparse channels. *IEEE Journal of Oceanic Engineering*, 32(4):927–939, October 2007.

[27] M. L. Littman. A tutorial on partially observable Markov decision processes. *Journal of Mathematical Psychology*, 53(3):119–125, June 2009.

[28] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.

[29] R. Zahedi, L. W. Krakow, E. K. P. Chong, and A. Pezeshki. Adaptive compressive sampling using partially observable Markov decision processes. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5269–5272, Kyoto, Japan, March 25-30, 2012.

[30] R. Zahedi, L. W. Krakow, E. K. P. Chong, and A. Pezeshki. Adaptive compressive measurement design using approximate dynamic programming. In *Proceedings of the 2013 American Control Conference (ACC)*, Washington, DC, June 17-19, 2013.

[31] J. Vermaak, S. J. Godsill, and P. Perez. Monte Carlo filtering for multi-target tracking and data association. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):309–332, January 2005.

[32] S. Oh, S. Russell, and S. Sastry. Markov chain Monte Carlo data association for general multiple-target tracking problems. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC)*, volume 1, pages 735–742, Atlantis, Paradise Island, Bahamas, December 14-17, 2004.

[33] R. Karlsson and F. Gustafsson. Monte Carlo data association for multiple target tracking. In *Proceedings of Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE*, volume 1, pages 13/1–13/5, October 16-17, 2001.

[34] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM*, 47(4):681–720, July 2000.

[35] V. D. Blondel and J. N. Tsitsiklis. A survey of computational complexity results in systems and control. *Automatica*, 36(9):1249–1274, September 2000.

[36] T. E. Fortmann, Y. Bar-Shalom, and M. Scheffe. Multi-target tracking using joint probabilistic data association. In *Proceedings of the 19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes (CDC)*, volume 19, pages 807–812, Albuquerque, NM, December 1980.

[37] Y. Bar-Shalom, F. Daum, and J. Huang. The probabilistic data association filter. *Control Systems, IEEE*, 29(6):82–100, December 2009.

[38] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *Aerospace and Electronic Systems Magazine, IEEE*, 19(1):5–18, January 2004.

[39] G. W. Pulford. Taxonomy of multiple target tracking methods. *IEE Proceedings Radar, Sonar and Navigation*, 152(5):291–304, October 2005.

[40] Edwin K. P. Chong, Christopher M. Kreucher, and Alfred O. Hero. Partially observable Markov decision process approximations for adaptive sensing. *Discrete Event Dynamic Systems*, 19(3):377–422, 2009.

[41] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, May 1998.

[42] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 3rd edition, January 2007.

[43] D. P. Bertsekas. Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334, 2005.

[44] J. H. Conway, R. H. Hardin, and N. J. A. Sloane. Packing lines, planes, etc.: Packings in Grassmannian spaces. *Experimental Mathematics*, 5(2):139–159, April 1996.

[45] R. Zahedi, A. Pezeshki, and E. K. P. Chong. Measurement design for detecting sparse signals. *Physical Communication*, 5(2):64–75, June 2012.

[46] H. Park and C. Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2, Part 2):3336–3341, March 2009.

[47] N. Nigam. The multiple unmanned air vehicle persistent surveillance problem: A review. *Machines*, 2(1):13–72, 2014.

[48] G. L. Plett, P. DeLima, and D. J. Pack. Target localization using multiple UAVs with sensor fusion via sigma-point Kalman filtering. *Proceedings of the 2007 AIAA*, 2007.

[49] M. L. Littman. A tutorial on partially observable markov decision processes. *Journal of Mathematical Psychology*, 53(3):119–125, 2009.

[50] S. A. Miller, Z. A. Harris, and E. K. P. Chong. A POMDP framework for coordinated guidance of autonomous UAVs for multitarget tracking. *EURASIP Journal on Advances in Signal Processing*, 2009(1):1–17, 2009.

[51] A. O. Hero, D. Castanon, D. Cochran, and K. Kastella. *Foundations and applications of sensor management*. Springer Science & Business Media, 2007.

[52] B. Ristic, B. Vo, and D. Clark. A note on the reward function for PHD filters with sensor control. *Aerospace and Electronic Systems, IEEE Transactions on*, 47(2):1521–1529, 2011.

[53] M. Jiang, W. Yi, and L. Kong. Multi-sensor control for multi-target tracking using Cauchy-Schwarz divergence. *CoRR*, abs/1603.08355, 2016.

[54] S. Ragi and E. K. P. Chong. UAV path planning in a dynamic environment via partially observable Markov decision process. *Aerospace and Electronic Systems, IEEE Transactions on*, 49(4):2397–2412, 2013.

[55] B. Vo and W. Ma. The Gaussian mixture probability hypothesis density filter. *Signal Processing, IEEE Transactions on*, 54(11):4091–4104, 2006.

[56] K. W. Lee, B. Kalyan, S. Wijesoma, M. Adams, F. S. Hover, and N. M. Patrikalakis. Tracking random finite objects using 3D-LIDAR in marine environments. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1282–1287. ACM, 2010.

[57] D. E. Clark, K. Panta, and B. Vo. The GM-PHD filter multiple target tracker. In *Information Fusion, 2006 9th International Conference on*, pages 1–8. IEEE, 2006.

[58] K. Granström, C. Lundquist, and O. Orguner. Extended target tracking using a Gaussian-mixture PHD filter. *Aerospace and Electronic Systems, IEEE Transactions on*, 48(4):3268–3286, 2012.

[59] E. K. P. Chong, C. M. Kreucher, and A. O. Hero. POMDP approximation using simulation and heuristics. In *Foundations and Applications of Sensor Management*, pages 95–119. Springer, 2008.

[60] B. Ristic, Ba-Ngu Vo, D. Clark, and Ba-Tuong Vo. A metric for performance evaluation of multi-target tracking algorithms. *Signal Processing, IEEE Transactions on*, 59(7):3452–3457, 2011.

[61] L. W. Krakow and E. K. P. Chong. Autonomous UAV control: Balancing target tracking and persistent surveillance. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1524–1529, Aug 2017.

[62] R. Sharma and D. Pack. Cooperative sensor resource management to aid multi-target geolocalization using a team of small fixed-wing unmanned aerial vehicles. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 4706, 2013.

[63] S. S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House radar library. Artech House, 1999.

[64] C. M. Eaton, E. K. P. Chong, and A. A. Maciejewski. Robust UAV path planning using POMDP with limited FOV sensor. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1530–1535, Aug 2017.

[65] Sony Electronics Inc. FCB-EV7500 Camera block specifications, 2005.

[66] C. M. Eaton, E. K. P. Chong, and A. A. Maciejewski. Multiple-scenario unmanned aerial system control: A systems engineering approach and review of existing control methods. *Aerospace*, 3(1):1, 2016.

[67] S. Ragi and E. K. P. Chong. UAV guidance algorithms via partially observable markov decision processes. In *Handbook of Unmanned Aerial Vehicles*, pages 1775–1810. Springer, 2015.

[68] L. W. Krakow and E. K. P. Chong. Autonomous UAV control: Balancing target tracking and persistent surveillance. In *Control Technology and Applications (CCTA), 2017 IEEE Conference on*, pages 1524–1529. IEEE, 2017.

[69] Y. Bar-Shalom, X Rong Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: Theory algorithms and software*. John Wiley & Sons, NJ, 2004.

[70] S. S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House radar library. Artech House, MA, 1999.

[71] Y. Bar-Shalom, T. E. Fortmann, and P. G. Cable. Tracking and data association. *The Journal of the Acoustical Society of America*, 87(2):918–919, 1990.

[72] E. K. P. Chong, C. M. Kreucher, and A. O. Hero. Partially observable Markov decision process approximations for adaptive sensing. *Discrete Event Dynamic Systems*, 19(3):377–422, 2009.

[73] C. Kreucher, A. O. Hero, K. Kastella, and D. Chang. Efficient methods of non-myopic sensor management for multitarget tracking. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 722–727. IEEE, 2004.

[74] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1st edition, 1996.

[75] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, MA, 1998.

[76] D. P. Bertsekas and D. A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

[77] E. K. P. Chong, R. L. Givan, and Hyeong Soo Chang. A framework for simulation-based network control via hindsight optimization. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 2, pages 1433–1438. IEEE, 2000.

[78] G. Wu, E. K. P. Chong, and R. Givan. Burst-level congestion control using hindsight optimization. *IEEE Transactions on Automatic Control*, 47(6):979–991, 2002.

[79] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, Belmont, MA, 3rd edition, 2007.

[80] Birdseyeview Aerobatics FireFLY6 Pro. https://www.birdseyeview.aero/. Accessed: 2018-01-30.

[81] Daniel J Pack, Pedro DeLima, Gregory J Toussaint, and George York. Cooperative control of uavs for localization of intermittently emitting mobile targets. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(4):959–970, 2009.

[82] Steven Rasmussen, Derek Kingston, and Laura Humphrey. A brief introduction to unmanned systems autonomy services (uxas). In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 257–268. IEEE, 2018.