AN INFORMATION-BASED APPROACH TO SENSOR RESOURCE ALLOCATION

by

Christopher M. Kreucher

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering : Systems)
in The University of Michigan
2005

Doctoral Committee:

Professor Alfred O. Hero III, Chair
Professor Jeffrey A. Fessler
Professor Susan A. Murphy
Assistant Professor Sandeep P. Sadanandarao

To the two great men of science that I have had the good to fortune to have known,
Ronald James Kreucher and Father Thomas O'Brien.

ACKNOWLEDGEMENTS

I would like to acknowledge the significant contributions to this work made by my advisors Dr. Keith Kastella and Professor Alfred O. Hero III. They have played equally important and complimentary roles in shaping the work reported in this thesis.

Dr. Kastella's early work in the development of the joint multitarget probability and Kullback-Leibler methods for sensor management provided the starting point for the work reported in the first two chapters of this thesis. Furthermore, his guidance and advice throughout the development of my extensions was instrumental in making this work a success. I also want to acknowledge his role as program manager at General Dynamics in setting up an environment where I was able to meet the requirements of the University while simultaneously remaining employed at General Dynamics.

Professor Hero's work using the Rényi Divergence in other problem domains provided the impetus for its adoption here. His guidance and advice on information based sensor scheduling and the multistage extensions was crucial to the work reported in the final two chapters of this thesis. I also want to acknowledge his graciousness and flexibility as my academic and dissertation advisor in allowing me to meet the demands of my workplace while simultaneously remaining a student at the University.

I also appreciate the interaction with my other committee members, Professors

Fessler, Murphy, and Pradhan. The comments, questions, and advice given by the members at my Quals II talk, my preliminary meeting, my final defense, and during personal interactions have significantly improved the dissertation.

Finally, I would like to acknowledge the contributions of my colleagues at General Dynamics, specifically Dan Chang and Karen Heinze. Both provided snippets of code or simulation results that contributed to work reported here.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

SUMMARY OF NOTATION

Notation	Meaning
$\mathbf{z}^k$	A measurement made at time k (scalar, vector, or matrix)
$\mathbf{Z}^k$	The collection of measurements made up to and including time k
$\mathbf{x}^k$	The state of a target at time k (e.g., $\mathbf{x} = [x, \dot{x}, y, \dot{y}]$)
$p(\mathbf{x}^k \vert \mathbf{Z}^{k-1})$	The prior estimate of single target state $\mathbf{x}$ given $\mathbf{Z}$ up to time $k-1$
$p(\mathbf{x}^k \vert \mathbf{Z}^k)$	The posterior estimate of single target state $\mathbf{x}$ given $\mathbf{Z}$ up to time k
$p(\mathbf{x}^k \vert \mathbf{x}^{k-1})$	The model of single target kinematics
$p(\mathbf{z}^k \vert \mathbf{x}^k)$	The single target sensor model
$\mathbf{x}_p^k$	Target state estimate provided by particle p at time k
w_p^k	The weight of particle p at time k
$\mathbf{X}^k$	The multitarget state at time k, i.e. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_{T-1}, \mathbf{x}_T]$
T^k	The number of targets at time k
$p(\mathbf{X}^k, T^k \vert \mathbf{Z}^{k-1})$	The prior estimate of multitarget state $\mathbf{X}$ and target number T
$p(\mathbf{X}^k, T^k \vert \mathbf{Z}^k)$	The posterior estimate of multitarget state $\mathbf{X}$ and target number T
$p(\mathbf{X}^k, T^k \vert \mathbf{X}^{k-1}, T^{k-1})$	The multitarget model of target kinematics
$p(\mathbf{z}^k \vert \mathbf{X}^k, T^k)$	The multitarget sensor model
$\mathbf{X}_p^k$	Multitarget state estimate provided by particle p at time k
$\mathbf{X}_{p,t}^k$	State estimate of target t provided by particle p at time k
T_p^k	Target number estimate of particle p at time k (implicit in $\mathbf{X}_p$)

ABSTRACT

AN INFORMATION-BASED APPROACH TO SENSOR RESOURCE ALLOCATION

by

Christopher M. Kreucher

Chair: Alfred O. Hero III

This work addresses the problem of scheduling the resources of agile sensors. We advocate an information-based approach, where sensor tasking decisions are made based on the principle that actions should be chosen to maximize the information expected to be extracted from the scene. This approach provides a single metric able to automatically capture the complex tradeoffs involved when choosing between possible sensor allocations.

We apply this principle to the problem of tracking multiple moving ground targets from an airborne sensor. The aim is to task the sensor to most efficiently estimate both the number of targets and the state of each target simultaneously. The state of a target includes kinematic quantities like position and velocity and also discrete variables such as target class and target mode (e.g., "turning" or "stopped"). In many experiments presented herein, target motion is taken from real recorded vehicle histories.

The information-based approach to sensor management involves the development

of three interrelated elements.

First, we form the joint multitarget probability density (JMPD), which is the fundamental entity capturing knowledge about the number of targets and the states of the individual targets. Unlike traditional methods, the JMPD does not assume any independence, but instead explicitly models coupling in uncertainty between target states, between targets, and between target state and the number of targets. Furthermore, the JMPD is not assumed to be of some parametric form (e.g., Gaussian). Because of this generality, the JMPD must be estimated using sophisticated numerical techniques. Our representation of the JMPD is via a novel multitarget particle filter with an adaptive sampling scheme.

Second, we use the estimate of the JMPD to perform (myopic) sensor resource allocation. The philosophy is to choose actions that are expected to maximize information extracted from the scene. This metric trades automatically between allocations that provide different types of information (e.g., actions that provide information about position versus actions that provide information about target class) without adhoc assumptions as to the relative utility of each.

Finally, we extend the information-based paradigm to non-myopic sensor scheduling. This extension is computationally challenging due to an exponential growth in action sequences with horizon time. We investigate two approximate methods to address this complexity. First, we directly approximate Bellman's equation by replacing the value-to-go function with an easily computed function of the ability to gain information in the future. Second, we apply reinforcement learning as a means of learning a non-myopic policy from a set of example episodes.

CHAPTER I

Introduction

In this work, we develop an information based approach to the problem of automatically allocating the resources of agile sensors. The techniques developed here are based on the principle that sensors should be tasked to take actions that maximize the expected amount of information extracted from the scene. We demonstrate herein that the information based technique has advantages over other strategies in that it automatically captures the complex tradeoffs involved when choosing between different sensing actions, requiring no additional ad hoc assumptions.

The method is applied to the problem of tracking multiple moving ground targets from airborne sensors. The aim in this situation is to task the sensor so as to most efficiently estimate both the number of targets and the state of each target simultaneously. In many experiments presented herein, target motion is taken from real, recorded vehicle histories and sensor measurements were simulated using a model of realistic sensors.

Our approach requires the development of three interrelated elements.

- *Multitarget Tracking Using a Particle Filter Representation of the Joint Multitarget Probability Density*: First, we construct a high fidelity nonparametric probabilistic model that captures the uncertainty inherent in the multitarget

tracking problem. We do this via the joint multitarget probability density (JMPD), which is a single entity that probabilistically describes the knowledge of the states (e.g., position and velocity in 2 dimensions plus identification) of each target as well as the number of targets. Due to the nature of the target tracking problem, it is essential to capture the correlations in uncertainty between the states of different targets as well as the coupling between the uncertainty about the number of targets and their individual states. The JMPD captures these couplings precisely as it makes no inherent factorization, independence, or parametric form assumptions about the density. Due to the high dimensionality and non-parametric nature of the density, advanced numerical methods are necessary to estimate the density in a computationally tractable manner. To this end, we have developed a novel multitarget particle filter with an adaptive sampling scheme.

- *Information Based Sensor Mangement*: Second, we use the estimate of the JMPD to make (myopic) sensor resource allocation decisions. We take an information-based approach, where the fundamental paradigm is to make sensor tasking decisions that maximize the expected amount of information gained about the scenario. This unifying metric allows us to automatically trade between sensor allocations that provide different types of information (e.g., actions that provide information about position versus actions that provide information about identification) without any ad hoc assumptions as to the relative utility of each.

- *Non-myopic Information Based Sensor Management*: Third, we extend the information-based sensor resource allocation paradigm to long-term (non-myopic) sensor scheduling. This extension allows the consideration of long-term infor-

mation gaining capability when making decisions about current actions. It is particularly important when the sensor has time-varying target response characteristics due to sensor motion, the behavior of the vehicles being tracked, or dynamic terrain features. We develop numerically efficient methods of approximating the long-term solution in situations where long-term scheduling is important.

We apply the method to the problem of ground target tracking from an airborne sensor. We illustrate performance through a series of experiments which use real collected target motion data. First, we look at the problem of detecting and tracking a set of moving ground targets using a pixelated sensor that returns energy (or merely thresholded detections) in a set of cells. Next, we extend the experiment to include target identification by simulating a multiple modality sensor capable of switching between a target indication mode and a target identification mode. We then extend the realism further by considering a moving platform and the presence of terrain. This creates regions that are obscured to the sensor and therefore multistage planning is required for optimal performance. Finally, we consider a situation where targets are "smart" in that they react to sensing actions by obscuring their whereabouts. We apply the multistage scheduling algorithm to this scenario using a multiple modality sensor.

We next review the history of the problem and our contribution to each of the aforementioned three elements of the algorithm.

1.1 Overview and Literature Review for Multitarget Tracking Using a Particle Filter Representation of the Joint Multitarget Probability Density

The problem of tracking a **single** maneuvering target in a cluttered environment is a very well studied area [1]. Normally, the objective is to predict the state of an object based on a set of noisy and ambiguous measurements. There are a wide range of applications in which the target tracking problem arises, including vehicle collision warning and avoidance [2][3], mobile robotics [4], human-computer interaction [5], speaker localization [6], animal tracking [7], tracking a person [8], and tracking a military target such as a ship, aircraft, or ground vehicle [9].

The single target tracking problem can be formulated and solved in a Bayesian setting by representing the target state probabilistically and incorporating statistical models for the sensing action and the target state transition. The standard tool is the Kalman Filter [10], applicable and optimal when the measurement and state dynamics are Gaussian and linear.

In a more general setting where nonlinear target motions, non-Gaussian densities, or non-linear measurement to target couplings are involved, more sophisticated nonlinear filtering techniques are necessary [11]. Standard nonlinear filtering techniques involve modifications to the Kalman Filter such as the Extended Kalman Filter [12], the Unscented Kalman Filter [13], and Gaussian Sum Approximations [14], all of which relax some of the linearity assumptions present in the Kalman Filter. However, these techniques do not accurately model all of the salient features of the density, which limits their applicability to scenarios where the target state posterior density is well approximated by a multivariate Gaussian density. To address this deficiency, others have studied grid-based approaches [15][16], which utilize a

discrete representation of the entire single target density. In this setup, no assumptions on the form of the density are required, so arbitrarily complicated densities may be accommodated. However, fixed grid approaches are computationally intractable except in the case of very low state space dimensionality [17].

Recently, the interest of the tracking community has turned to the set of Monte Carlo techniques known as Particle Filtering [18]. A particle filter approximates a probability density on a set of discrete points, where the points are chosen dynamically via importance sampling. Particle filtering techniques have the advantage that they provide computational tractability [19], have provable convergence properties [20], and are applicable under the most general of circumstances, as there is no assumption made on the form of the density, the noise process, the measurement to state coupling, or the state evolution [21]. Indeed, particle filter based approaches have been used successfully in areas where grid based [22] or Extended or Unscented Kalman Filter-based [23][24] filters have previously been employed.

The **multitarget** tracking problem has been traditionally addressed with techniques such as multiple hypothesis tracking (MHT) and joint probabilistic data association (JPDA) [1][9][25]. Both techniques work by translating a measurement of the surveillance area into a set of detections by thresholding a likelihood ratio. The detections are then either associated with existing tracks, used to create new tracks, or deemed false alarms. Typically, Kalman-filter type algorithms are used to update the existing tracks with the new measurements after association. This leads to what is known as the data association problem, which is the primary computational challenge of these methods. As the number of targets and measurements grow, there are exponentially many possible associations between the existing targets and the measurements. It is also important to note that these techniques do not explicitly

model the multitarget density, but merely a collection of single target densities – i.e., correlations between target states are ignored.

Others have approached the problem from a fully Bayesian perspective. Stone [26] develops a mathematical theory of multiple target tracking from a Bayesian point of view. Mori [27], Srivistava, Miller [28], Kastella [29] and Mahler [30] did early work in this area. For the same reasons as the single target case, use of Extended and Unscented Kalman Filter approaches are typically inappropriate. Furthermore, due to the explosion in dimensionality of the state space, fixed grid approaches are computationally intractable.

Recently, some researchers have applied particle filter based strategies to the problem of multitarget tracking. In [31], Hue and Le Cadre introduce the probabilistic multiple hypothesis tracker (PMHT), which is a blend between the traditional MHT and particle filtering. Considerable attention is given to dealing with the measurement to target association issue. Others have done work which amounts to a blend between JPDA and particle filtering [32][33][34].

The BraMBLe [35] system, the independent partition particle filter (IPPF) of Orton and Fitzgerald [36], the work of Maskell [37] and Tao [38] all consider multitarget tracking from a Bayesian perspective via particle filtering. Measurement-to-target association is not done explicitly; it is implicit within the Bayesian framework. In short, these works focus on a tractable implementation of ideas in [26]. All of these efforts represent excellent advances, but each suffers from some deficiencies that limit widespread use. For example, [35] employs only the simplest of sampling methods requiring tens of thousands of particles to track four or five targets. The methods of [36] and [37], while both giving non-trivial sampling schemes, do not account for unknown target number. Furthermore, the method of [37] does not model the multi-

target state vector, but proposes merely a combination of single target state vectors. Finally, none of these works explicitly recognize and lay out the connection between the particle filter and the underlying probability density being estimated (i.e., the JMPD).

The main contribution of our work in the area of multitarget tracking is the development of a Bayesian multiple target tracker that is designed explicitly to estimate the joint multitarget probability density. The estimation is done using particle filtering methods with a carefully designed adaptive importance sampling density. Among the benefits of this approach are that target number is estimated simultaneously with the individual target states, and no thresholding of measurements is required.

These features distinguish the particle filter based JMPD approach from traditional approaches of MHT and JPDA as well as the approaches of Hue [31][39] and others [32][40][41], which require thresholded measurements (detections) and a measurement-to-track association procedure. Additionally, as our method utilizes an adaptive sampling scheme, deals with unknown target number, and explicitly models the multitarget state, it generalizes the efforts of [35], [36], [37], and [38].

In the simulation experiments of Chapter II, we demonstrate that the particle filter implementation of JMPD provides a natural way to track a collection of targets, is computationally tractable, and performs well under difficult conditions such as target crossing, convoy movement, and low SNR.

1.2 Overview and Literature Review for Information Based Sensor Mangement

Sensor resource allocation, or sensor management, refers to the problem of determining the best way to task a sensor or group of sensors when each sensor may have many modes and search patterns. Typically, the sensors are used to gain information

about the kinematic state (e.g., position and velocity) and identification of a group of targets as well as the number of targets. Applications of sensor management are often military in nature [42], but also include things such as wireless networking [43] and robot path planning [44]. One of the main issues with robust sensor management is that there are many objectives that the sensor manager may be tuned to meet, e.g., minimization of track loss, probability of new target detection, minimization of track error/covariance, and identification accuracy. Each of these different objectives taken alone may lead to a different sensor allocation strategy [42][45].

Sensor scheduling strategies may be myopic (single-stage) or non-myopic (multi-stage). In the myopic case, sensing actions are taken so as to maximize the immediate reward, and as such are greedy. Myopic methods have the advantage that they are more computationally tractable than non-myopic methods. In this section, we discuss our approach to myopic sensor management and then extend this to non-myopic sensor management in the following section.

Some researchers have proposed using information measures as a general means of sensor management. Information theory has the benefit that it provides a single metric[1] that is useful for scheduling sensors across different objectives. In the context of Bayesian estimation, a good measure of the quality of a sensing action is the reduction in entropy of the posterior distribution that the measurement is expected to create. Therefore, information theoretic methodologies strive to take sensing actions that maximize expected gain in information. The possible sensing actions are enumerated, the expected gain for each measurement is calculated, and the action that yields the maximal expected gain is chosen.

Much of the work involving information-theoretic ideas in this context is for track-

[1] As we will discuss, there are many information theoretic metrics that have been used in the literature, including the Rényi Divergence, Kullback-Leibler Divergence, and Mutual Information. However, any particular choice of information measure represents a single metric able to balance different types of information.

ing moving targets. Hintz et al. [46][47] did early work focusing on using the expected change in Shannon entropy when tracking a single target moving in one dimension with Kalman Filters. A related approach uses discrimination gain based on a measure of relative entropy, the Kullback-Leibler (KL) divergence. Schmaedeke and Kastella [48] use the KL divergence to determine sensor-to-target taskings. Kastella [49][50] uses KL divergence to manage a sensor between tracking and identification mode in the multitarget scenario. Mahler [51][30] uses the KL divergence as a metric for "optimal" multisensor multitarget sensor allocation. Zhao [52] compares several approaches, including simple heuristics, and information-based techniques based on entropy and relative entropy (KL).

Information theoretic measures have also been used by the machine learning community in techniques with the names "active learning" [53], "learning by query" [54], "relevance feedback" [55][56], and "stepwise uncertainty reduction" [57]. A specific example is the interactive search of a database of imagery for a desired image, also called content based image retrieval (CBIR). Geman [57] studied the situation where a user has a specific image in mind and the system steps through a sequence of images presented to the user under a binary forced-choice protocol. A pair of images is chosen by the system at each time and the user is asked to choose the one that is most similar to the specific image in mind. The image pairs are chosen in such a way that the posterior density of the unknown image has the lowest resulting Shannon entropy after the user responds. Similarly, Cox et al. [56] uses a set of psychophysical experiments to model human notion of closeness for a robust CBIR system.

Information-based adaptivity measures such as mutual information (related to the KL divergence) and entropy reduction have also been used by the computer vision community in techniques with the names "active object recognition" [58], "active

computer vision" [59], and "active sensing" [60]. These techniques are iterative procedures wherein the system has the ability to change sensor parameters to make the learning task easier. The ultimate goal is to learn something about the environment, e.g., the class of an object, the orientation of a robot's tool, or the location of the robot within an area.

In the context of multitarget tracking, we employ information theoretic methods to allocate sensor resources so as to estimate the number of targets present in the surveillance region as well as the states of the individual targets. Since the number and states of the targets is a dynamic process that evolves over time, we use the particle filter based multitarget tracking algorithm discussed earlier to recursively estimate the JMPD. At each iteration of the algorithm, we use an information measure to decide on which sensing action to take. The decision as to how to use a sensor then becomes one of determining which sensing action will maximize the expected information gain between the current JMPD and the JMPD after a measurement has been made. That is, sensing actions are driven by the ability of the sensor to reduce uncertainty in the JMPD. Actions that are anticipated to maximize the relative entropy between the prior and posterior JMPD are favored.

In this work, we consider a quite general information measure called the Rényi Information Divergence [61] (also known as the α-divergence), which reduces to the KL divergence for a certain value of the parameter α. The Rényi divergence has additional flexibility in that it allows for emphasis to be placed on specific portions of the densities under comparison.

The main contribution of this work in the area of myopic sensor management is the development of an approach where the Rényi divergence is to estimate the utility of taking different sensing actions, where the underlying multitarget density

is approximated by a multitarget particle filter. To the best of our knowledge, this is the first time the Rényi Divergence has been used in this context, and the first time a particle filter has been used as the basis for a multitarget sensor management algorithm.

We show that the information theoretic approach provides a unified method for sensor management able to task sensors by choosing both mode (e.g., SAR mode or GMTI mode) and pointing direction for the purposes of detecting, tracking and identifying multiple moving targets. In particular, we demonstrate order-of-magnitude type gains in sensor efficiency when compared to no scheduling (i.e., periodic scan). In addition, we show that the information measures outperform other intuitive management algorithms predicated on pointing the sensor near where the targets are expected to be. Under certain conditions, we also show the algorithm provides a computationally tractable method of performing sensor management and tracking for tens of targets.

1.3 Overview and Literature Review for Non-myopic Information Based Sensor Management

To maximally extract information about a surveillance region, scheduling decisions must consider the impact of the actions on the ability to gather information in the future. This is referred to as non-myopic or long-term sensor scheduling. Situations where the sensor has a time varying effectiveness benefit particularly from long-term scheduling. An example is the situation where target and/or sensor platforms are moving, which causes the visibility of a target from a sensor changes with time as terrain features (e.g., mountains, trees) block the direct path from the sensor to the target. Planning ahead by taking certain actions at the current time step that do not maximize immediate gain in information may lead to improvements of long term

gain in information.

Many researchers have approached the non-myopic sensor scheduling problem with a Markov decision process (MDP) strategy. However, a complete long-term scheduling solution suffers from combinatorial explosion when solving practical problems of even moderate size. Researchers have thus worked to develop approximate solution techniques.

For example, Krishnamurthy [62][63] uses a multi-arm bandit formulation involving hidden Markov models. In [62], an optimal algorithm is formulated to track multiple targets with an electronically scanned array that has a single steerable beam. Since the optimal approach has prohibitive computational complexity, several suboptimal approximate methods are given and some simple numerical examples involving a small number of targets moving among a small number of discrete states are presented. Even with the proposed suboptimal solutions, the problem is still very challenging numerically. In [63], the problem is reversed, and a single target is observed by a single sensor from a collection of sensors. Again, approximate methods are formulated due to the intractability of the globally optimal solution.

Bertsekas and Castañon [64] formulate heuristics for the solution of a stochastic scheduling problem corresponding to sensor scheduling. They implement a rollout algorithm based on their heuristics to approximate the solution of the stochastic dynamic programming algorithm. Additionally, Castañon [65][66] formulates the problem of classifying a large number of stationary objects with a multi-mode sensor based on a combination of stochastic dynamic programming and optimization techniques. In [67] Malhotra proposes using reinforcement learning as an approximate approach to dynamic programming.

Chhetri [68] approaches the long-term scheduling problem for a single target using

particle filters and the unscented transform. The method involves drawing samples from the predicted future distribution and minimizing expected future costs. This requires enumeration of the exponentially growing number of possible sensing actions, a very computationally demanding procedure. This is combined with branch and bound techniques which require some restrictive assumptions on additivity of costs. In a series of works, Zhao [43][69][52] et al. investigate sensor management in the setting of a wireless ad hoc network, which involves some long term considerations such as power management.

The main contribution of this work in the area of non-myopic sensor management is the combination of information theoretic methods with strategies for approximating the non-myopic scheduling problem. These strategies are distinguished from earlier efforts in that the cost (reward) function for deciding on what sensing action to take is given by the expected gain in information (as measured by the Rényi Divergence) for that action. This method is in keeping with the philosophy of taking actions that maximize information (minimize entropy) of the resulting posterior density on target number and target state.

The methods we investigate here include sparse sampling techniques, direct approximation of Bellman's equation, and reinforcement learning techniques. We illustrate the tradeoffs between computation and performance of the strategies in our setting.

1.4 Advancements Made and Conclusions Drawn by this Dissertation

As has been outlined in the preceding subsections and will be further clarified in the body of this thesis, the advancements made to the field of target tracking and sensor management by this work include

- The development of a tractable particle filter based multitarget tracker to recursively estimate the joint multitarget probability density (JMPD) (see Chapter II). This approach simultaneously addresses estimation of target number and the state of each individual target, is nonparametric, and makes no assumptions of linearity or Gaussianity.
- The development of the Rényi Divergence metric for resource allocation in the multitarget tracking scenario (see Chapter III). This method chooses sensor taskings in a manner that automatically trades between detection information, kinematic information, and identification information. The metric is general enough so that additional knowledge about the priority of each task can be incorporated.
- The extension of the information based sensor scheduling approach to multistage decision making through direct approximation and learning techniques (see Chapter IV).

As a result of this work, we can draw following broad conclusions about the problem domain and the utility of our work.

- By appropriate design of importance density, it is possible to construct a tractable particle filter based multitarget tracker capable of estimating both the number of targets and the individual states of each in situations involving tens of targets (see Chapter II.)
- The Rényi Divergence framework for resource allocation is theoretically grounded and provides a natural method for trading the effects of different sensing actions (see Chapter III).

  - The particle filter estimation and Rényi Divergence resource allocation algorithm are robust in the face of model mismatch (see Section 3.4.5).
  - Through marginalization and weighting, the Rényi Divergence can be used as a surrogate for task specific metrics (see Section 3.3).
  - In the case of discrete action spaces, this method provides a tractable method of resource allocation (see Section 3.4.6).
  - This method outperforms heuristic methods designed with domain knowledge (see Section 3.4.2.)

- Multistage planning results in significant performance gain in situations where the system dynamics are changing rapidly (see Chapter IV).
  - Simple approximations to the MDP can provide good approximations to the multistage solution in many common scenarios (see Section 4.2.4).
  - Reinforcement learning methods are broadly applicable and can be used to address the multistage scheduling problem when training data and computational resources are available (see Section 4.2.5).

1.5 Outline of Dissertation

This dissertation is organized as follows.

In Chapter II, we introduce and describe the mathematics behind the joint multitarget probability density (JMPD) and show how the rules of Bayesian Filtering are applied to produce a recursive filtering procedure. We detail our particle filter based representation of the JMPD, which provides a numerically tractable method for tracking a few tens of targets. The tractability comes from a novel importance density design, which combines an automatic factorization of the JMPD when targets

are behaving independently, and a measurement-directed proposal for both persistent targets and targets that come and go with time. We furthermore detail the permutation symmetry issue (present in all multitarget tracking algorithms) and its manifestation in our particle filter estimation of the JMPD. We conclude with simulation results detailing the performance of the particle-filter-based multitarget tracker in situations stressful to traditional trackers, including unknown target count, many targets, and low signal to noise ratio (SNR).

In Chapter III, we detail the (myopic) information based sensor management scheme. The strategy employs the Rényi divergence as a metric, and is predicated on making measurements that are expected to attain maximum immediate gain in information about the surveillance region. We provide mathematical and computational details of how the Rényi divergence is estimated in the context of a particle filter representation of the JMPD. We show therein how the information based sensor management strategy is able to automatically balance complex tradeoffs when selecting both sensor mode and pointing angle. We furthermore provide a performance analysis of the tracker using sensor management on several model problems and discuss computational tractability with large numbers of targets. We include comparisons to a non-managed (periodic) scheme and two other sensor management techniques.

In Chapter IV, we present non-myopic extensions of the information based sensor management algorithm introduced in Chapter III. Sensor allocations are made based on maximizing long-term gain in information at each sensor selection time. We first provide a motivating example of a scenario in which non-myopic sensor management provides benefit. We then show by example the full optimal multistage scheduling approach, and note the intractability for problems with long time-scales or large

action spaces. We develop several approximate methods for scheduling sensors using information theoretic measures. First, we detail an information-directed method of selectively searching through sets of actions sequences. Second, we detail an approximate technique for solving the Bellman equation which replaces the value-to-go with a function that approximates the long-term value of an action. This technique has computational cost on the order of the myopic scheme. Third, we investigate Q-learning as a method of learning the optimal sensor allocation strategy and the environment by embedded simulations. Finally, we provide simulation results comparing the myopic, non-myopic, and approximate techniques in terms of track error and computational burden.

We conclude in Chapter V with some summary remarks.

1.6 List of Relevant Publications

The following publications are related to this dissertation.

1. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Sensor Management Using An Active Sensing Approach", *Signal Processing*, vol. 85, no. 3, pp. 607-624, 2005.
2. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Multitarget Tracking using a Particle Filter Representation of the Joint Multitarget Probability Density", to appear in *IEEE Transactions on Aerospace and Electronic Systems*.
3. C. M. Kreucher, D. Blatt, A. O. Hero III, and K. Kastella, "Adaptive Multimodality Sensor Scheduling for Detection and Tracking of Smart Targets", Proceedings of the 2004 *Defense Applications of Signal Processing (DASP) Workshop*.

4. C. M. Kreucher, D. Blatt, A. O. Hero III, and K. Kastella, "Adaptive Multimodality Sensor Scheduling for Detection and Tracking of Smart Targets", to appear in *Digital Signal Processing*.

5. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Multiplatform Information-based Sensor Management", to appear in *The SPIE International Symposium on Defense and Security*, Orlando, FL, 28 March - 1 April, 2005.

6. C. M. Kreucher and A. O. Hero III, "Non-myopic Approaches to Scheduling Agile Sensors for Multitarget Detection, Tracking, and Identification", to appear in *The 2005 IEEE Conference on Acoustics, Speech, and Signal Processing Special Section on Advances in Waveform Agile Sensor Processing*, Philadelphia, PA, March 18-23, 2005 (Invited Paper).

7. M. Morelande and C. M. Kreucher, "Multiple Target Tracking with a Pixelated Sensor", to appear in *The 2005 IEEE Conference on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, March 18-23, 2005.

8. C. M. Kreucher, M. Morelande, A. O. Hero III, and K. Kastella, "Particle Filtering for Multitarget Detection and Tracking", to appear in *The 2005 IEEE Aerospace Conference*, Big Sky, Montana, March 5-12, 2005 (Invited Paper).

9. C. M. Kreucher, A. O. Hero III, K. Kastella, and D. Chang, "Efficient Methods of Non-myopic Sensor Management for Multitarget Tracking", *The 43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, December 2004.

10. C. M. Kreucher, A. O. Hero III, and K. Kastella, "Multiple Model Particle Filtering for Multitarget Tracking", *The Twelfth Annual Workshop on Adaptive Sensor Array Processing*, Lexington, Mass, March 2004.

11. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Tracking Multiple Targets Using a Particle Filter Representation of the Joint Multitarget Probability Density", *The SPIE International Symposium on Optical Science and Technology*, San Diego, California, August 2003.

12. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Information-based sensor management for multitarget Tracking", *The SPIE International Symposium on Optical Science and Technology*, San Diego, California, August 2003.

13. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Particle Filtering and Information Prediction for Sensor Management", *2003 Defense Applications of Data Fusion Workshop*, Adelaide, Australia, July 2003.

14. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Information Based Sensor Management for Multitarget Tracking", *Proceedings of the Workshop on Multiple Hypothesis Tracking: A Tribute to Samuel S. Blackman*, San Diego, CA, May 30, 2003.

15. C. M. Kreucher, Kastella K., and A. O. Hero III, "A Bayesian Method for Integrated Multitarget Tracking and Sensor Management", *The 6th International Conference on Information Fusion*, Cairns, Australia, July 2003.

16. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Tracking Multiple Targets Using a Particle Filter Representation of the Joint Multitarget Probability Density", *Sixth ONR/GTRI Workshop on Target Tracking and Sensor Fusion*, San Diego, CA, May 28-29, 2003.

17. C. M. Kreucher, K. Kastella, and A. O. Hero III, "Multi-target Sensor Management Using Alpha Divergence Measures", *The 2nd International Conference on Information Processing in Sensor Networks*, Palo Alto, California, April 2003

(General Dynamics Medal Paper Award Winner, 2003).

CHAPTER II

Multitarget Tracking Using a Particle Filter Representation of the Joint Multitarget Probability Density

In this chapter, we give the details of our method of generating a probabilistic estimate of the state of a multitarget system. In general the "state" of the system refers to the number of targets and the individual states of each target – which typically includes kinematic values such as position, velocity and acceleration, but may also include discrete components like mode (e.g., a target is moving or a target is turning) and identification (e.g., tank, or jeep). To describe this method, this chapter first provides a comprehensive framework for multitarget detection, tracking, and identification that includes unknown and time varying target number and follows by detailing the particle filter based implementation.

This chapter proceeds as follows. In Section 2.1, we give the details of a nonlinear filtering methodology based on recursively estimating the joint multitarget probability density (JMPD). The discussion is an expanded and generalized version of the discussion given by Kastella in [70]. In Section 2.2, we give our multitarget particle filter implementation of the JMPD method. In that section, we proceed by first introducing the notion of particle filtering, followed by multitarget particle filtering and concluding with our method based on an adaptive measurement directed sampling scheme for computational tractability. Finally, we conclude in Section 2.3 with a

detailed set of simulations involving unknown target number, varying signal to noise ratio, and crossing targets. As will be described therein, many of the simulations are performed using target motion data taken from real, recorded target trajectories.

2.1 The Joint Multitarget Probability Density

In this section, we give the details of a Bayesian method of multitarget tracking predicated on recursive estimation of the Joint Multitarget Probability Density (JMPD). Many others have studied Bayesian methods for tracking multiple targets, e.g., [26][28][71]. In particular, Mahler [51][72][73] advocates a related approach based on random sets which he calls "finite-set statistics" (FISST). Since FISST and the JMPD approach attack some of the same problems, many of the concepts that appear here such as multitarget motion models and multitarget measurement models also appear in the work of Mahler et. al. [72]. FISST is a theoretical framework for unifying most techniques for reasoning under uncertainty (e.g., Dempster Shafer, fuzzy, Bayes, rules) in a common structure based on random sets. The JMPD method can be derived in the FISST framework and both strategies can both be traced back to the EAMLE work of Kastella [73][74][29][49]. The EAMLE work builds on early multitarget tracking work such as [27][75][76] and others. The JMPD technique does not require the random set formalism of FISST; in particular, in contrast to the random set approach, the JMPD technique adopts the view that likelihoods and the joint multitarget probability density are conventional Bayesian objects to be manipulated by the usual rules of probability and statistics. Therefore, the JMPD approach described here makes no appeal to random sets or related concepts such as Radon-Nikodym derivatives.

The concept of JMPD was discussed in [73][49][70], where a method of tracking

multiple targets that moved between discrete cells on a line was presented. We generalize the discussion here to deal with targets that have N-dimensional continuous valued state vectors and arbitrary kinematics. In many of the model problems, we are interested in tracking the position (x, y) and velocity $(\dot{x}, \dot{y})$ of multiple targets and so we describe each target by the four dimensional state vector $[x, \dot{x}, y, \dot{y}]'$. At other times, we may be interested in estimating the identity or mode of each target and so will augment the state vector appropriately.

A simple schematic showing three targets (Targets A, B, and C) moving through a surveillance area is given in Figure 2.1. There are two target crossings, a challenging scenario for multitarget trackers.

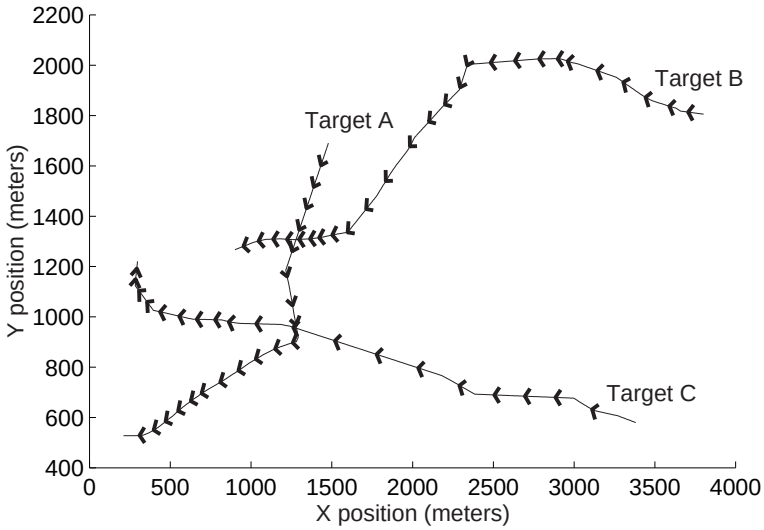


Figure 2.1: An example of the trajectories of a group of three moving targets. The target trajectories come from real target motion collected as part of a military training exercise and recorded by GPS. The target paths are indicated by the lines, and direction of travel by the arrows. There are two instances where the target paths cross (i.e. are at the same sensor resolution cell at the same time).

Recursive estimation of the JMPD provides a means for tracking an unknown number of targets in a Bayesian setting. The statistical model employed uses the

joint multitarget conditional probability density

$$p(\mathbf{x}_1^k, \mathbf{x}_2^k, ...\mathbf{x}_{T-1}^k, \mathbf{x}_T^k, T^k|\mathbf{Z}^k) = \tag{2.1}$$
$$p(\mathbf{x}_1^k, \mathbf{x}_2^k, ...\mathbf{x}_{T-1}^k, \mathbf{x}_T^k|T^k, \mathbf{Z}^k)p(T^k|\mathbf{Z}^k)$$

as the probability density for exactly T^k targets with states $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_{T-1}, \mathbf{x}_T$ at time k based on a set of past observations $\mathbf{Z}^k$. We abuse terminology by calling $p(\mathbf{x}_1^k, \mathbf{x}_2^k, ...\mathbf{x}_{T-1}^k, \mathbf{x}_T^k, T^k|\mathbf{Z}^k)$ a density since T^k is a discrete valued random variable. In fact, as (2.1) shows, the JMPD is a continuous discrete hybrid as it is a product of the probability mass function $p(T^k|\mathbf{Z}^k)$ and the probability density function $p(\mathbf{x}_1^k, \mathbf{x}_2^k, ...\mathbf{x}_{T-1}^k, \mathbf{x}_T^k|T^k, \mathbf{Z}^k)$.

The number of targets at time k, T^k, is a variable to be estimated simultaneously with the states of the T^k targets. The JMPD is defined for all T^k, $T^k = 0 \cdots \infty$. The observation set $\mathbf{Z}^k$ refers to the collection of measurements up to and including time k, i.e. $\mathbf{Z}^k = \{\mathbf{z}^1, \mathbf{z}^2, ...\mathbf{z}^k\}$, where each of the $\mathbf{z}^i$ may be a single measurement or a vector of measurements made at time i.

Each of the $\mathbf{x}_t$ in the density $p(\mathbf{x}_1^k, \mathbf{x}_2^k, ...\mathbf{x}_{T-1}^k, \mathbf{x}_T^k|T^k, \mathbf{Z}^k)$ is a vector quantity and may (for example) be of the form $[x, \dot{x}, y, \dot{y}]$. We refer to each of the T target state vectors $\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_{T-1}, \mathbf{x}_T$ as a partition of the multitarget state $\mathbf{X}$. For convenience, the density will be written more compactly in the traditional manner as $p(\mathbf{X}^k|T^k, \mathbf{Z}^k)$, which implies that the state-vector $\mathbf{X}^k$ represents a variable number of targets each possessing their own state vector. We will drop the time superscript k for notational simplicity when no confusion will arise.

As an illustration, some examples illustrating the sample space of p are

- $p(\emptyset, T = 0|\mathbf{Z})$, the posterior probability that there are no targets in the surveillance area.

- $p(\mathbf{x}_1, T = 1|\mathbf{Z})$, the posterior probability that there is one target in the surveillance area with state $\mathbf{x}_1$.
- $p(\mathbf{x}_1, \mathbf{x}_2, T = 2|\mathbf{Z})$, the posterior probability that there are two targets in the surveillance area with states $\mathbf{x}_1$ and $\mathbf{x}_2$.

The likelihoods $p(\mathbf{z}|\mathbf{X}, T)$ and the joint multitarget probability density $p(\mathbf{X}, T|\mathbf{Z})$ are conventional Bayesian objects manipulated by the usual rules of probability and statistics. Thus, a multitarget system has state $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$ with probability distribution $p(\mathbf{x}_1, \cdots, \mathbf{x}_T, T|\mathbf{Z})$. This can be viewed as a hybrid stochastic system where the discrete random variable T governs the dimensionality of $\mathbf{X}$. Therefore the normalization condition that the JMPD must satisfy is

$$\sum_{T=0}^{\infty} \int d\mathbf{x}_1 \cdots d\mathbf{x}_T p(\mathbf{x}_1, \cdots, \mathbf{x}_T, T|\mathbf{Z}) = 1 \ . \tag{2.2}$$

where the single integral sign is used to denote the T integrations required.

Quantities of interest can be deduced from the JMPD. For example, the probability that there are exactly T targets present in the surveillance area is given by the marginal distribution

$$p(T|\mathbf{Z}) = \int d\mathbf{x}_1 \cdots d\mathbf{x}_T p(\mathbf{x}_1, \cdots, \mathbf{x}_T, T|\mathbf{Z}) \ . \tag{2.3}$$

An important factor that is often overlooked in multitarget tracking algorithms is that the JMPD is symmetric under permutation of the target indices. This symmetry is a fundamental property of the JMPD which exists because of the physics of the problem and not because of mathematical construction. Specifically, the multitarget states $\mathbf{X} = \big(\mathbf{x}_1, \mathbf{x}_2\big)$ and $\mathbf{X}' = \big(\mathbf{x}_2, \mathbf{x}_1\big)$ refer to the same event, namely that there are two targets in the surveillance area – one with state $\mathbf{x}_1$ and one with state $\mathbf{x}_2$. This is true regardless of the makeup of the single target state vector. For example, the

single target state vector may include target ID or even a target serial number and the permutation symmetry remains. Therefore, all algorithms designed to implement the JMPD are permutation invariant.

If targets are widely separated in the sensor's measurement space, each target's measurements can be uniquely associated with it, and the joint multitarget posterior density approximately factors. In this case, the problem may be treated as a collection of single target problems. The characterizing feature of multitarget tracking is that in general some of the measurements have ambiguous associations, and therefore the conditional density does not simply factor into a product of single target densities.

The temporal update of the posterior likelihood proceeds according to the usual rules of Bayesian filtering. The model of how the JMPD evolves over time is given by $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$ and will be referred to as the kinematic prior (KP). The kinematic prior includes models of target motion, target birth and death, and any additional prior information that may exist such as terrain and roadway maps. In the case where target identification is part of the state being estimated, different kinematic models may be used for different target types. The time-updated (prediction) density is computed via the *model update* equation as

$$p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1}) = \sum_{T^{k-1}=0}^{\infty} \int_{\mathbf{X^{k-1}}} d\mathbf{X}^{k-1} p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1}) p(\mathbf{X}^{k-1}, T^{k-1}|\mathbf{Z}^{k-1}) \ . \tag{2.4}$$

Note that this formulation of the time evolution of the JMPD makes several assumptions. First, as is commonly done, we assume that state evolution is Markov. Furthermore, we assume the action at time $k-1$ does not influence state evolution, i.e., if the sensing action performed at time $k-1$ is denoted m^{k-1} then by assumption $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1}, m^{k-1}) = p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$. In some situations this

assumption is not valid, including the "smart" target problem (see Section 4.3.4). If either of these assumptions is invalid in a particular setting, (2.4) would be generalized appropriately.

The *measurement update* equation uses Bayes' rule to update the posterior density with a new measurement $\mathbf{z}^k$ as

$$p(\mathbf{X}^k, T^k|\mathbf{Z}^k) = \frac{p(\mathbf{z}^k|\mathbf{X}^k, T^k)p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})}{p(\mathbf{z}^k|\mathbf{Z}^{k-1})} \quad . \tag{2.5}$$

This formulation allows JMPD to avoid altogether the problem of measurement-to-track association which is the fundamental computational issue in conventional multitarget tracking algorithms such as MHT and JPDA. There is no need to identify which target is associated with which measurement because the Bayesian framework keeps track of the entire joint multitarget density. This property, of course, introduces a different but related computational challenge which will be addressed later. Furthermore, there is no need for thresholded measurements (detections) to be used at all. A tractable sensor model merely requires the ability to compute the likelihood $p(\mathbf{z}|\mathbf{X}, T)$ for each measurement $\mathbf{z}$ received. This property allows the JMPD technique to generalize and outperform other multitarget tracking algorithms particularly in low SNR environments.

In the following subsections, we describe the specific models used in this work for target kinematics and the sensor, respectively.

2.1.1 Kinematic Modeling : The Model $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$

The Bayesian framework outlined above requires a model of how the system state evolves, $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$. This includes both how the number of targets changes with time (i.e. T^k versus T^{k-1}) and how individual targets that persist over time evolve (i.e. $\mathbf{x}^k$ versus $\mathbf{x}^{k-1}$). In general, this model is chosen using the physics of

the particular system under consideration. The target motions in the simulation studies presented in this thesis come from a set of real ground targets recorded during a military battle exercise. Therefore, here we specialize the state models to this application. More general models (or simply different models) are possible and can be implemented similarly if warranted by the physics of the situation.

To specify the state model, we need to generate an expression for how the state of the system evolves, $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$, which can be evaluated for any set of multitarget states and target counts $\{\mathbf{X}^k, T^k, \mathbf{X}^{k-1}, T^{k-1}\}$.

We first define a set of spatially varying priors on target arrival and departure from the surveillance region. We refer to these two events as target birth and death. Let $\alpha^k(\mathbf{x})$ denote the *a priori* probability that a target will arrive (birth) at location $\mathbf{x}$ at time k. Similarly, let the *a priori* probability that a target in location $\mathbf{x}$ will leave the surveillance region (death) be denoted by $\beta^k(\mathbf{x})$. Target arrival may indicate a target actually passing into the surveillance region (e.g., along the border of the region) or may indicate the target has taken an action that first makes it detectible to the sensor (e.g., motion by a target when the only sensor is a moving target indicator). Target death may indicate a target actually leaving the surveillance region or becoming permanently extinguished. These model parameters specify how target number changes with time.

The birth/death model can be precisely written as follows. The model is that birth and death represent a Markov process with spatially varying rates $\alpha(\mathbf{x}^k)$ and $\beta(\mathbf{x}^k)$. Assuming a pixelated sensor with constant birth/death rates in each cell (see Section 2.1.2), these rates can be denoted α_i^k and β_i^k, where i represents the pixel that $\mathbf{x}$ maps into. Using t_i^k to represent the number of targets in cell i, the model

can then be written

$$
\begin{aligned}
p(t_i^{k+1} = 0 | t_i^k = 1) &= \beta_i^k \\
p(t_i^{k+1} = 0 | t_i^k = 0) &= 1 - \alpha_i^k \\
p(t_i^{k+1} = 1 | t_i^k = 1) &= 1 - \beta_i^k \\
p(t_i^{k+1} = 1 | t_i^k = 0) &= \alpha_i^k \ .
\end{aligned} \tag{2.6}
$$

For targets that persist over a time step (those targets that are present at both time step k and $k+1$), we model the target motion as linear and independent for each target. Using $\mathbf{x} = (x, \dot{x}, y, \dot{y})$ to denote the state vector of an individual target, the model is

$$
\mathbf{x}_i^k = \mathbf{F}\mathbf{x}_i^{k-1} + \mathbf{w}_i^k \ , \tag{2.7}
$$

where

$$
\mathbf{F} = \begin{pmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{pmatrix} \ . \tag{2.8}
$$

Here $\mathbf{w}_i^k$ is 0-mean Gaussian noise with covariance $\mathbf{Q} = diag(20, .2, 20, .2)$, which was selected based on an empirical fit to the data. In the cases where we are interested in tracking the mode or identity of targets, the state vector and $\mathbf{F}$ and $\mathbf{Q}$ matrices are augmented appropriately. If appropriate, a multiple hidden Markov model formulation of the kinematic model is possible [77].

We emphasize here that Linear/Gaussian models are not a requirement of the formulation, but are used as they have been found to perform well in simulation studies with the real data. More complicated models of target motion can be inserted where appropriate without directly effecting computations in the algorithm.

2.1.2 Sensor Modeling : The Model $p(\mathbf{z}^k|\mathbf{X}^k, T^k)$

To implement Bayes′ Formula (2.5), we must compute the measurement likelihood $p(\mathbf{z}|\mathbf{X}, T)$. This likelihood is derived from the physical nature of the sensor. There are two approaches to modeling the likelihood, which we refer to as the "associated measurement" model and the "association-free" model. In both models, the sensor produces a sequence of scans at discrete instants in time. Each scan is a set of measurements produced at the same instant. The difference between the models lies in the structure of the scans.

In the associated measurement model, an observation vector consists of M measurements, denoted $\mathbf{z} = (z_1, \ldots, z_M)$. $\mathbf{z}$ is composed of threshold exceedances, i.e. valid detections and false alarms. Each valid measurement is generated by a single target and is related (possibly non-linearly) to the target state. False alarms have a known distribution independent of the targets (usually taken as uniform over the observation space) and the targets have known detection probability P_d (usually constant for all targets). The origin of each measurement is unknown. If measurement m is generated by target t, then it is a realization of the random process $z_m \sim H_t(\mathbf{x}_t, w_t)$.

In its usual formulation, the associated measurement model *precludes* the possibility of two different targets contributing to a single measurement. This model predominates most current tracking, data fusion and sensor management work. The practical advantage of this model is that it breaks the tracking problem into two disjoint sub-problems: data association and filtering. The filtering problem is usually treated using some kind of Kalman filter. The disadvantages are that a restricted sensor model is required and the combinatorial problem of associating observations to filters is difficult. The associated measurement model was initially conceived in order to cast the problem into a form in which the Kalman filter can be applied, which

is understandable in light of the enormous success the Kalman filter has enjoyed.

In contrast, nonlinear filtering methods allow much greater flexibility regarding the way measurements are modeled. As a result, we are free to employ an association-free sensor model in the work presented here. Note that we could employ an associated measurement model if desired, but it leads to poorer performance for the reasons outlined above. This type of model has been used in track-before-detect algorithms, in the "Unified Data Fusion" work of Stone et. al [26] and in the grid-based sensor management work of [49]. There are several advantages to the association-free method. First, it requires less idealization of the sensor physics and can readily accommodate issues such as merged measurements, side-lobe interference amongst targets and velocity aliasing. Second, it eliminates the combinatorial bottleneck of the associated-measurement approach. Finally, it allows processing of pre-thresholded measurements to enable improved tracking at lower target SNR.

In this work, we consider models associated with sensors commonly used in tracking and surveillance applications. As motivation, we briefly review some typical examples of surveillance sensors here.

An imaging sensor may observe a collection of unresolved point objects. The imager returns a collection of 1- or 2-dimensional pixel intensities. The output of each pixel is related to the integrated photon count in that pixel which is in turn determined by the background rate and how many targets are present within the pixel during the integration interval, and their locations within the pixel. This is represented numerically as either a positive integer or real number. Depending on the nature of the optics and their impulse response function, one or more pixels may respond to a target. Furthermore, multiple targets can contribute to the output of a single pixel, violating the assumptions of the associated measurement model.

Another commonly used sensor type is radar. In a ground moving target indicator (GMTI) radar, a collection of pulses is emitted, their returns are collected and integrated over some coherent processing interval (CPI) [78]. The output of successive CPIs may also be averaged non-coherently. During the integration interval, the radar antenna is directed at some fixed or slowly varying bearing. The integrated pulse data is processed to obtain the reflectivity as a function of range and range-rate at that average bearing. Depending on the nature of the integration process, the return amplitude may be envelope detected or it may be available in complex form. Given the ubiquity of modern digital signal processing, radar data is usually available somewhere within the radar system as a data array indexed by discrete range, range-rate and bearing values.

With this as background motivation, we present the association-free model adopted in this work. We compute the measurement likelihood $p(\mathbf{z}|\mathbf{X}, T)$, which describes how sensor output depends on the state of all of the targets in the surveillance region. A sensor scan consists of M pixels, and a measurement $\mathbf{z}$ consists of the pixel output vector $\mathbf{z} = [z_1, \ldots, z_M]$, where z_i is the output of pixel i. In general, z_i can be an integer, real, or complex valued scalar, a vector or even a matrix, depending on the sensor. If the data are thresholded, then each z_i will be either a 0 or 1. Note that for thresholded data, $\mathbf{z}$ consists of both threshold exceedances and non-exceedances. The failure to detect a target at a given location can have as great an impact on the posterior distribution as a detection.

We model both measurements of spatially separated pixels and measurements of the same pixel at different time instants as conditionally independent given the state, i.e.

$$p(\mathbf{z}|\mathbf{X}, T) = \prod_i p(z_i|\mathbf{X}, T) \ . \tag{2.9}$$

Independence between the measurements given the state is often approximately true, and modeling as such often provides a nice simplification. However, conditional independence amongst the measurements is not a necessary part of this framework. Occasions where the physics of the situation imply sensor returns are dependent warrant a more general sensor model. This will not change the framework given here, only the implementation of the likelihood $p(\mathbf{z}|\mathbf{X}, T)$.

Let $\chi_i(\mathbf{x}_t)$ denote the indicator function for pixel i, defined as $\chi_i(\mathbf{x}_t) = 1$ when a (single) target in state $\mathbf{x}_t$ projects into sensor pixel i (i.e. couples to pixel i) and $\chi_i(\mathbf{x}_t) = 0$ when it does not. Observe a pixel can couple to multiple targets and single target can contribute to the output of multiple pixels, say, by coupling through side-lobe responses. The indicator function for the joint multitarget state is constructed as the logical disjunction

$$\chi_i(\mathbf{X}, T) = \bigvee_{t=1}^{T} \chi_i(\mathbf{x}_t) \ . \tag{2.10}$$

The set of pixels that couple to $\mathbf{X}$ is

$$i_{\mathbf{X}} = \{i|\chi_i(\mathbf{X}, T) = 1\} \ . \tag{2.11}$$

For the pixels that do not couple to $\mathbf{X}$, the measurements are characterized by the background distribution, denoted $p_0(z_i)$. With this, (2.9) can be written as

$$p(\mathbf{z}|\mathbf{X}, T) = \prod_{i \in i_{\mathbf{X}}} p(z_i|\mathbf{X}, T) \prod_{i \notin i_{\mathbf{X}}} p_0(z_i) \propto \prod_{i \in i_{\mathbf{X}}} \frac{p(z_i|\mathbf{X}, T)}{p_0(z_i)} \ . \tag{2.12}$$

Equation (2.12) allows for fairly general modeling of a pixelated sensor response to a collection of targets including non-linear effects due to multiple targets contributing to a single pixel. One limitation is aggregations of targets only couple to the union of pixels that the individual targets couple to. If a pair of targets have some type of nonlinear coupling that results in a contribution to a pixel that they do not couple

to individually, this is not captured in the model. This is likely to be a very small effect in most situations, so we choose to ignore it here.

We further idealize the sensor as having a box-car resolution cell in position coordinates. The x- and y- ground-plane projection of each pixel is Δ_x and Δ_y. The sensor response within pixel i is uniform for targets in i and vanishes for targets outside pixel i. It is convenient to define the occupation number $n_i(\mathbf{X})$ for pixel i as the number of targets in $\mathbf{X}$ that lie in i. The single target signal-noise-ratio (SNR), assumed constant across all targets, is denoted λ. We assume that when multiple targets lie within the same pixel their amplitudes add non-coherently (this will be an accurate model for unresolved optical targets and radar targets not moving as a rigid body). Then the effective SNR when there are n targets in a pixel is $\lambda_n = n\lambda$ and we use $p_n(z_i)$ to denote the pixel measurement distribution (note that the background distribution is obtained by setting $n = 0$).

With these modeling assumptions, the measurement distribution in pixel i depends only on its occupation number and (2.12) becomes

$$p(\mathbf{z}|\mathbf{X}, T) \propto \prod_{i \in i_{\mathbf{X}}} \frac{p_{n_i(\mathbf{X}),T}(z_i)}{p_0(z_i)} \quad . \tag{2.13}$$

To complete the specification of the sensor model, we must give its dependence on SNR. Many models are plausible, depending on the detailed nature of the sensor physics. In this work, we have elected to use Rayleigh-distributed measurements. This distribution corresponds to envelope detected signals under a complex Gaussian radar return model, and has been used, for example, to model interfering targets in a monopulse radar system [25][79] and to model clutter and target returns in turbulent environments [80]. Rayleigh models are also often used for diffuse fading channels.

In the pre-thresholded case, this implies

$$p_n(z) = \frac{z}{1+n\lambda} \exp\left(-\frac{z^2}{2(1+n\lambda)}\right) \quad . \tag{2.14}$$

When the tracker only has access only to thresholded measurements, we use a constant false-alarm rate (CFAR) model for the sensor. If the background false alarm rate is set at P_f, then the detection probability when there are n targets in a pixel is

$$P_{d,n} = P_f^{\frac{1}{1+n\lambda}} \quad . \tag{2.15}$$

This extends the usual relation $P_d = P_f^{\frac{1}{1+\lambda}}$ for thresholded Rayleigh random variables at SNR λ [1].

2.2 The Particle Filter Implementation of JMPD

We now turn to the development of a particle filter approximation to the Joint Multitarget Probability Density (JMPD) [81][82]. Even for modest problems, the sample space of the JMPD is enormous as it contains all possible configurations of state vectors $\mathbf{x}_t$ for all possible values of T. Specifically, if the state of an individual target is given by the 4-tuple $\left(x, \dot{x}, y, \dot{y}\right)$, the sample space of the JMPD then contains vectors of length $4N$ for all positive finite N. Earlier implementations of JMPD given by Kastella [49] approximated the density by discretizing on a grid. The computational burden in this scenario makes evaluating realistic problems intractable, even when using the simple model of targets moving between discrete locations in one-dimension. In fact, for a fixed approximation error, the number grid cells needed grows as L^T, where L is the number of discrete locations the targets may occupy and T is the number of targets.

Thus, to estimate the JMPD in a computationally tractable manner, a more sophisticated approximation method is required. We find that a particle filter (PF)

based implementation of JMPD breaks the computational logjam and allows us to investigate more realistic problems.

2.2.1 The Single Target Particle Filter

Before describing the particle filter implementation of the JMPD, we first review standard single target particle filtering. Particle filtering is a method of approximately solving the prediction and update equations by simulation [11][18], where samples from the target density are used to represent the density and are propagated through time. As the number of particles tends towards infinity, the approximation converges to the true posterior being approximated [21].

To implement a single-target particle filter, the single-target density of interest, $p(\mathbf{x}|\mathbf{Z})$, is approximated by a set of N_{part} weighted point masses (particles):

$$p(\mathbf{x}|\mathbf{Z}) \approx \sum_{p=1}^{N_{part}} w_p \delta_D(\mathbf{x} - \mathbf{x}_p) \ , \tag{2.16}$$

where δ_D represents the usual Dirac impulse. The model update (2.4) and the measurement update (2.5) are simulated by the following three step recursion.

First, the particle locations at time k are generated using the particle locations at time $k-1$ and the current measurements by sampling from an importance density, denoted $q(\mathbf{x}^k|\mathbf{x}^{k-1}, \mathbf{z}^k)$. The design of the importance density is a well studied area [21], as the choice of the importance density can have a dramatic effect on the efficiency of the particle filter algorithm. It is known that the optimal importance density (OID) is given by $p(\mathbf{x}^k|\mathbf{x}^{k-1}, \mathbf{z}^k)$, but this density is typically prohibitively difficult to sample from. In practice, oftentimes the importance density is chosen just to be the kinematic prior $p(\mathbf{x}^k|\mathbf{x}^{k-1})$. However, more sophisticated choices of importance density lead to better results for a fixed number of particles. As we will see in the multitarget case, approximating the OID (rather than simply using the

Table 2.1: The SIR single target particle filter.

1. For each particle p, $p = 1, \cdots, N_{part}$,
   (a) Sample $\mathbf{x}_p^k \sim q(\mathbf{x}^k|\mathbf{x}^{k-1}, \mathbf{z}^k) = p(\mathbf{x}|\mathbf{x}_p^{k-1})$
   (b) Compute $w_p^k = w_p^{k-1} * p(\mathbf{z}|\mathbf{x}_p)$ for each p
2. Normalize w_p to sum to 1, $w_p \leftarrow w_p / \sum_{i=1}^{N_{part}} w_i$.
3. Resample N_{part} particles with replacement from $\mathbf{x}_p$ based on the distribution defined by w_p

kinematic prior) becomes crucial as problem dimension increases.

Second, particle weights are updated according to the weight equation (2.17), which involves the likelihood, the kinematic model, and the importance density [11].

$$w_p^k = w_p^{k-1} \frac{p(\mathbf{z}^k|\mathbf{x}_p^k) p(\mathbf{x}_p^k|\mathbf{x}_p^{k-1})}{q(\mathbf{x}_p^k|\mathbf{x}_p^{k-1}, \mathbf{z}^k)} \quad . \tag{2.17}$$

When using the kinematic prior as the importance density, the weight equation reduces to simply $w_p^k = w_p^{k-1} * p(\mathbf{z}^k|\mathbf{x}_p^k)$.

Finally, a resampling step is used to combat particle degeneracy. Without resampling, the variance of the particle weights increases with time, yielding a single particle with all the weight after a small number of iterations [21]. Resampling may be done on a fixed schedule or based on the weight variance.

The particle filter algorithm that uses the kinematic prior as the importance density and resamples at each time step is called sampling importance resampling (SIR) in the literature. The algorithm is summarized in Table 2.1.

2.2.2 SIR Multitarget Particle Filtering

To implement the JMPD recursions via a particle filter, we similarly approximate the joint multitarget probability density $p(\mathbf{X}, T|\mathbf{Z})$ by a set of N_{part} weighted samples. A particle now becomes more than just the estimate of the state of a target; it

incorporates both an estimate of the states of all of the targets as well as an estimate of the number of targets.

As we write the multitarget state vector for T targets as

$$\mathbf{X} = [\mathbf{x}_1,\ \mathbf{x}_2,\ ...,\ \mathbf{x}_{T-1},\ \mathbf{x}_T]\ , \tag{2.18}$$

the particle state vector will be written as

$$\mathbf{X}_p = [\mathbf{x}_{p,1},\ \mathbf{x}_{p,2},\ \ \ldots\ \ \mathbf{x}_{p,T_p}]\ . \tag{2.19}$$

Implicit in a particle $\mathbf{X}_p$ is the number of targets T_p, where T_p can be any non-negative integer. With δ_D denoting the Dirac delta, we define

$$\delta(\mathbf{X} - \mathbf{X}_p) = \begin{cases} 0 & T \neq T_p \\ \delta_D(\mathbf{X} - \mathbf{X}_p) & \text{otherwise} \end{cases} . \tag{2.20}$$

Then the particle filter approximation to the JMPD is given by a set of particles and corresponding weights as

$$p(\mathbf{X}, T|\mathbf{Z}) \approx \sum_{p=1}^{N_{part}} w_p \delta(\mathbf{X} - \mathbf{X}_p)\ , \tag{2.21}$$

where $\sum_{p=1}^{N_{part}} w_p = 1$.

The joint multitarget probability density $p(\mathbf{X}, T|\mathbf{Z})$ is defined for all possible numbers of targets, $T = 0, 1, 2, \cdots$. As each of the particles $\mathbf{X}_p$, $p = 1...N_{part}$ is a sample drawn from the JMPD $p(\mathbf{X}, T|\mathbf{Z})$, a particle $\mathbf{X}_p$ may have $0, 1, 2, \cdots$ partitions, each partition corresponding to the state vector of a different target. Different particles in the approximation may correspond to different estimates of the number of targets in the surveillance region.

We will denote the t^{th} partition of particle p by $\mathbf{X}_{p,t}$, i.e. $\mathbf{X}_{p,t}$ refers to the estimate of the t^{th} target state made by particle p. Since a partition corresponds to a target,

Table 2.2: The SIR multitarget particle filter.

1. For each particle p, $p = 1, \cdots, N_{part}$,
   (a) Draw a sample of the multitarget state and target number $(\mathbf{X}_p, T_p)$ at time k from $q(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1}, \mathbf{z}^k) = p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$
   (b) Compute $w_p^k = w_p^{k-1} * p(\mathbf{z}|\mathbf{X}_p)$ for each p
2. Normalize w_p to sum to 1, $w_p \leftarrow w_p / \sum_{i=1}^{N_{part}} w_i$.
3. Resample N_{part} particles with replacement from $\mathbf{X}_p$ based on w_p

the number of partitions that a particle has is that particle's estimate of the number of targets in the surveillance area.

With these definitions, the SIR particle filter extends directly to JMPD filtering, as shown in Table 2.2. Particles at time k are drawn using the particles at time $k-1$ and an importance density $q(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$. In its simplest form, the target kinematics are used for proposal, i.e. $q(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1}) = p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$.

The weight equation is

$$w_p^k = w_p^{k-1} \frac{p(\mathbf{z}^k|\mathbf{X}_p^k) p(\mathbf{X}_p^k|\mathbf{X}_p^{k-1})}{q(\mathbf{X}_p^k|\mathbf{X}_p^{k-1}, \mathbf{z}^k)} \quad . \tag{2.22}$$

As in the single target case, since the model of target kinematics is used to propose particles, the weight equation (2.17) simplifies to become the measurement likelihood, $p(\mathbf{z}|\mathbf{X}_p)$.

Targets entering or leaving the surveillance region are taken into account as the proposed particle $\mathbf{X}_p^k$ may have either fewer targets or more targets than $\mathbf{X}_p^{k-1}$ (i.e. $T_p^k = T_p^{k-1} - 1$ or $T_p^k = T_p^{k-1} + 1$).

2.2.3 The Inefficiency of the SIR Method

The SIR multitarget particle filter has the benefit of being simple to describe and easy to implement. These benefits, however, are erased by computational complexity

concerns since using the kinematics requires an enormous amount of particles for successful tracking (as illustrated later in Figure 2.3). In fact, SIR is so numerically inefficient that problems of any realistic size are intractable with this method of sampling.

Assume for discussion that the sensor is pixelated, returning energy in one of C sensor cells as discussed in Section 2.1.2. Target birth may occur in one or many of the cells at any time step. Target death may occur in any occupied cell at any time step (or in multiple cells). One method of handling this would be to have a very large number of particles, capable of encoding all possibilities of the next state, i.e. no new target, one new target (in each of the possible unoccupied cells), or one less target (in each of the occupied cells) and all the combinations of multiple additions and removals. Since the state space contains many possible locations for the new target (e.g., a 100x100 sensor grid), the straightforward method would require an enormous number of particles to include the possible permutations of targets removed and added.

Furthermore, even in the (artificial) case where there is no birth and death, target proposals using the kinematics are too inefficient to be useful on realistic problems. Consider the case where there a M targets in the surveillance region. In order for a particle to be a good estimate of the multitarget state, all M partitions must be proposed to good locations. Without knowledge of the measurements, the probability of an individual target being proposed to a good location is much less than 1. Therefore, as the number of targets grows, the number of particles required to have some particles have a good estimate of all the targets simultaneously (with high probability) grows exponentially.

Both of these problems can be remedied by using an importance density that

more closely approximates the optimal importance density. Specifically, using the current measurements to direct particle proposals to higher likelihood multitarget states will provide significant efficiency gains. Therefore, rather than using the model of kinematics $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$ for proposal, we advocate a carefully designed importance density that more closely approximates the optimal density $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1}, \mathbf{z}^k)$. The following subsections describe this design. The discussion is broken into two parts, one dealing with target birth and death and one dealing with targets that remain from time step k to time step $k+1$, i.e., persistent targets. The analysis is unified after discussion of the components individually.

2.2.4 Importance Density Design for Target Birth and Death

To reach the efficiency required for tractable detection of multiple targets, we advocate a measurement directed sampling scheme for target birth and death. Specifically, we keep an existence grid (separate from the particles and tied to the sensor grid) which contains the probability for t targets in cell i at time k given the measurements $\mathbf{Z}^k$, $p_i(t^k|\mathbf{Z}^k)$. We consider only two possible values for t: 0 (no target in cell i) and 1 (a target exists in cell i). Therefore, the existence grid is merely a single vector of floating point numbers, one for each sensor cell. Note that this implicitly assumes non-interacting targets at initialization. If target existence in one cell depends on targets in other cells, the joint probability $p(t_1^{k+1}, t_2^{k+1}, \cdots t_N^{k+1}|\mathbf{Z}^k)$ would be required.

The existence grid cells are initialized with a prior probability, $p_i(t^0|\emptyset)$, which may be spatially varying. The probability of target existence in each cell is propagated forward in time via

$$p_i(t^{k+1}|\mathbf{Z}^k) = \int p_i(t^{k+1}|t^k) p_i(t^k|\mathbf{Z}^k) dt^k \quad . \tag{2.23}$$

where $p_i(t^{k+1}|t^k)$ is the model of time evolution of target number. Specifically, it encapsulates the probability of 1 target at time $k+1$ given there were 0 targets at time k, the probability of 1 target at time $k+1$ given there was 1 target at time k, and so on. According to the model discussed earlier, new targets arrive in cell $\mathbf{x}$ at time k at the rate $\alpha(\mathbf{x}^k)$ and leave cell $\mathbf{x}$ at time k at the rate of $\beta(\mathbf{x}^k)$. This model completely specifies the transition density. Since t can only take on one of two values, this integral becomes a simple summation that is easily computable.

The existence grid is updated according to Bayes' rule when new measurements, $\mathbf{z}^{k+1}$, come in as

$$p_i(t^{k+1}|\mathbf{Z}^{k+1}) = \frac{p_i(t^{k+1}|\mathbf{Z}^k)p_i(\mathbf{z}^{k+1}|t^{k+1})}{p_i(\mathbf{z}^{k+1})} \quad . \tag{2.24}$$

These update procedures result in an existence grid that is separate from the particles and which contains a probability of target existence.

To handle target birth, new targets are preferentially added in locations according to the rate dictated by $p_i(\textit{target}^k|\mathbf{Z}^k)$ rather than at the prior rate given by $\alpha(\mathbf{x}^k)$. This is a bias which will be removed during the weight update process so that the Bayesian recursions are still exactly implemented. This implementational technique allows particles to be used more efficiently as new targets are only added in highly probable areas. Similarly, to handle target death, targets are preferentially removed at the rate dictated by $p_i(\textit{no target}^k|\mathbf{Z}^k)$ rather than the prior rate given by $\beta(\mathbf{x}^k)$.

This bias is removed during the weight update so that the Bayesian recursions are still exactly implemented. Specifically, the particle weights are modified by a function which relates the true physical arrival rate $\alpha(\mathbf{x}^k)$ to the arrival rate actually used to add new targets. Similarly, the particle weight must be modified by a function which relates the true removal rate $\beta(\mathbf{x}^k)$ to the removal rate actually used by the filter.

For the purposes of this work, we assume the model is that the arrival and re-

moval rates are spatially and temporally constant and thus $\alpha(\mathbf{x}^k)$ and $\beta(\mathbf{x}^k)$ are constants equal to α and β. The generalization to temporally and spatially varying rates is straightforward to implement but significantly complicates notation. This generalization is pursued in [83].

We know specify the exact manner in which bias is removed. Let the number of possible locations where a target could be added in the surveillance region at time k be denoted e^k. Furthermore, for notational convenience denote the existence-grid weight in cell i at time k by g_i^k. Finally, for a set of integers Z let $\mu_m^Z(1,j),\ldots,\mu_m^Z(m,j)$, $j \in \{1,\ldots,\binom{|Z|}{m}\}$ denote the jth combination of m integers from Z. Then, if particle p adds a targets preferentially to cells given in the set j_1 and removes b targets preferentially from cells given in the set j_2, the bias correction factor (to be used in particle weight update) is given by

$$m_p = \frac{\lambda_a \rho_b}{\nu_{a,j_1}\kappa_{b,j_2}} \tag{2.25}$$

where λ and ρ are defined using the prior birth and death probabilities as

$$\lambda_a = (1-\alpha)^{e_k}\Big(\frac{\alpha}{1-\alpha}\Big)^a \tag{2.26}$$

and

$$\rho_b = (1-\beta)^{T^{k-1}}\Big(\frac{\beta}{1-\beta}\Big)^b \ . \tag{2.27}$$

The quantities ν and κ are defined from the addition and removal rates used by the filter as

$$\nu_{a,j} = \prod_{i\in A^k}(1-g_i^k)\prod_{l=1}^{a} g^k_{\mu_a^{A^k}(l,j)} \Big/ \Big(1-g^k_{\mu_a^{A^k}(l,j)}\Big) \tag{2.28}$$

and

$$\kappa_{b,j} = \prod_{i=1}^{T^{k-1}}\big(1-\tau_i^k\big)\prod_{l=1}^{b} \tau^k_{\mu_b^{T^{k-1}}(l,j)} \Big/ \Big(1-\tau^k_{\mu_b^{T^{k-1}}(l,j)}\Big) \tag{2.29}$$

where

$$\tau_i^k = \beta \frac{1 - g^k_{v_i^k}}{1/T^{k-1} \sum_{l=1}^{T^{k-1}} (1 - g^k_{v_i^k})} \quad . \tag{2.30}$$

2.2.5 Importance Density Design for Persistent Targets

The drawback to using the kinematic prior for persistent targets is that the method does not explicitly take advantage of the fact that the state vector represents many targets. Targets that are far apart in measurement space behave independently and should be treated as such. Furthermore, similarly to the uninformed birth/death proposal, the current measurements are not used when proposing new particles. These two considerations taken together result in an inefficient use of particles and therefore to be successful at target tracking, methods using the kinematics as the importance density require large numbers of particles.

To overcome these deficiencies, we propose a particle proposal technique which biases proposed particles toward the measurements and allows for factorization of the multi-target state when permissible. These strategies propose each partition in a particle separately, and form new particles as the combination of the proposed partitions. We describe two methods here, the independent partitions (IP) method of [36] and the coupled partitions (CP) method. The basic idea behind both CP and IP is to construct particle proposals at the partition level, incorporating the measurements so as to propose in a manner that is close to the optimal importance density. We show that each method has benefits and drawbacks and propose an adaptive partition (AP) method which automatically switches between the two as appropriate.

Specifically, the CP method proposes particles in a permutation invariant manner. However, it has the drawback of being computationally demanding. also, when used

on all partitions individually, the IP method is not permutation invariant. However, it is significantly more computationally efficient than CP. The solution we advocate is to perform an analysis of the particle set at each time step to determine which partitions require the CP algorithm, and which partitions may be proposed via the IP method. This analysis leads to the AP method of proposal which is permutation invariant.

The Independent-Partition (IP) Method

The independent partition (IP) method given by Orton [36] is a convenient way to propose particles when the JMPD factors. In its simplest form, a partition corresponds directly to a target, and that is how we describe it here. This assumption is relaxed later. The Independent-Partition (IP) method proposes a new partition as follows. For a partition t, each particle at time $k-1$ has it's t^{th} partition proposed via the kinematic prior and weighted by the measurements. From this set of N_{part} weighted estimates of the state of the t^{th} target, we select N_{part} samples with replacement to form the t^{th} partition of the particles at time k.

When using IP, we are implying an importance density $q(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$ that is no longer simply the model of target kinematics $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$ as in the SIR multitarget particle filter. Therefore, the weight given by (2.17) does not simply become the likelihood $p(\mathbf{z}^k|\mathbf{X}^k, T^k)$. There is a bias which leads to preferring to select partitions in accordance with the likelihood of the partition. To account for this sampling scheme, the biases corresponding to each particle for each partition, $b_{p,t}$, are retained to use in conjunction with the likelihood $p(\mathbf{z}^k|\mathbf{X}^k, T^k)$ when computing particle weights. Furthermore, note that some particles will estimate a particular target does not exist. This estimate is treated exactly as other estimates. This algorithm is summarized in Table 2.3.

Table 2.3: The independent partition particle filter.

1. For each partition, $t = 1 \cdots T_{max}$, propose partition t via Independent Partition Subroutine
2. Remove or add targets to each particle as in Section 2.2.4 resulting in bias term m_p
3. Compute $w_p^k = w_p^{k-1} * \frac{p(\mathbf{z}|\mathbf{X}_p)}{m_p \prod_{t=1}^{T_{max}} b_{p,t}}$

Independent Partition Subroutine for Target t:

1. For each particle $p = 1, ..., N_{part}$,
   (a) Sample $\mathbf{X}_{p,t}^* \sim p(\mathbf{X}_{p,t}^k | \mathbf{X}_{p,t}^{k-1})$
   (b) Compute $w_p = p(\mathbf{z}|\mathbf{X}_{p,t}^*)$
2. Normalize w_p to sum to 1, $w_p \leftarrow w_p / \sum_{i=1}^{N_{part}} w_i$.
3. For each particle $p = 1, ..., N_{part}$,
   (a) Sample an index j from the distribution defined by w
   (b) Set $\mathbf{X}_{p,t} = \mathbf{X}_{j,t}^*$
   (c) Retain bias of sample, $b_{p,t} = w_j$

It is important to carefully account for the permutation symmetry issue discussed in Section 2.2 here. The IP method makes the critical assumption that partition t in each particle corresponds to the same target. Therefore, the partitions in each particle must be aligned before this method can be applied. If IP is applied to particles that have different orderings of partitions, multiple targets will be grouped together within the same partition and erroneously used to propose the location of a single target. In Section 2.2.6, a method of sorting that lines partitions up across all of the particles and allows IP to be used more often. When the assumption of target/partition correspondence is valid, IP is an effective sampling strategy because it combines results for each partition across particles, resulting in improved numerical efficiency.

In the case of well separated targets, this method allows many targets to be tracked with the same number of particles needed to track a single target. Indeed,

as mentioned earlier, in the case of well separated targets, the multitarget tracking problem breaks down into many single-target problems. The IP method is useful for just this case, as it allows the targets to be treated independently when their relative spacing is sufficiently large to decouple the JMPD. Note, however, that this method is not applicable when there is any measurement-to-target association ambiguity. Therefore, when targets are close together in sensor space, an alternative method must be used.

The Coupled Partition (CP) Proposal Method

When the posterior distributions on target position begin to overlap, we say that the corresponding partitions are coupled. In these instances, the IP method is no longer applicable, and another method of particle proposal such as Coupled Partitions (CP) must be used. An alternative method would be to use the IP strategy on groups of partitions as suggested in [36]. We suggest instead the CP method which recognizes the increased difficulty of the problem when overlap happens and combats it via increased sampling.

We apply the coupled partitions method as follows. To propose partition t of particle p, the CP method proposes R possible realizations of the future state using the kinematic prior. The R proposed futures are then given weights according to the current measurements and a single representative is selected. This process is repeated for each particle until the t^{th} partition for all particles has been formed. This can interpreted as an auxiliary particle filter [84] where the multiplicity R plays the role of the auxiliary variable. It can also be interpreted as a Monte Carlo approximation to the OID for a partition. As in the IP method, the final particle weights must be adjusted for this biased sampling. The algorithm is summarized in Table 2.4.

Table 2.4: The coupled partition particle filter.

1. For each partition, $t = 1 \cdots T_{max}$, propose partition t via Coupled Partition Subroutine
2. Remove or add partitions as in Section 2.2.4 resulting in bias term m_p
3. Compute $w_p^k = w_p^{k-1} * \frac{p(\mathbf{z}|\mathbf{X}_p)}{m_p \prod_{t=1}^{T_p} b_{p,t}}$

Coupled Partition Subroutine for Target t

1. For each particle $p = 1, ..., N_{part}$,
   (a) For each proposal $r = 1, ..., R$
      i. Sample $\mathbf{X}_{p,t}^*(r) \sim p(\mathbf{X}_{p,t}^k | \mathbf{X}_{p,t}^{k-1})$
      ii. Compute $w_r = p(\mathbf{z}|\mathbf{X}_{p,t}^*(r))$
   (b) Normalize w_r to sum to 1, $w_r \leftarrow w_r / \sum_{i=1}^{R} w_i$.
   (c) Sample an index j from the distribution defined by w
   (d) Set $\mathbf{X}_{p,t} = \mathbf{X}_{p,t}^*(j)$
   (e) Retain bias of sample, $b_{p,t} = w_j$

This algorithm is a modified version of the traditional SIR technique that operates on partitions individually, i.e., uses IP. It improves tracking performance over SIR at the expense of additional computations.

Adaptive Particle Proposal Method

To mitigate the problem of additional computational cost of the CP method, and the problems with the IP method when targets are close together, we propose a hybrid solution, called the Adaptive-Partition (AP) method. The adaptive-partition method again considers each partition separately. Those partitions that are sufficiently well separated according to a given metric (see below) from all other partitions are treated as independent and proposed using the IP method. When targets are not sufficiently distant, the CP method is used.

To determine when targets are sufficiently separated, we threshold based on distance in sensor space between the estimated state of the i^{th} partition and the j^{th}

Table 2.5: The adaptive proposal method.

1. For each partition $t = 1 : T_{max}$
   (a) $d(t) = \min_{j \neq t} ||\hat{\mathbf{x}}_t - \hat{\mathbf{x}}_j||$
   (b) if $d(t) > \tau$
   Propose partition t using IP method
   (c) else
   Propose partition t using CP method
2. Remove or add partitions as in Section 2.2.4 resulting in bias term m_p
3. For each particle $p = 1, ..., N_{part}$
   $w_p^k = w_p^{k-1} * \frac{p(\mathbf{z}|\mathbf{X}_p)}{m_p \prod_{t=1}^{T_p} b_{p,t}}$

partition. Denote by $\hat{\mathbf{x}}_i$ the estimated x and y positions of the i^{th} partition (2.41). Notice that only the spatial states are used (i.e. velocities are neglected), as our model assumes that no velocity information is available from the sensor. A more general model (which would be driven by a more capable sensor) would lead to a more general distance metric. We have computed the distance between two partitions using a Euclidian metric between the estimated centers, and the Mahalanobis metric (2.31), where $\hat{\Sigma}_j$ is the covariance associated with the estimate of the j^{th} partition (see 2.42).

$$r^2 = (\hat{x}_i - \hat{x}_j)' \hat{\Sigma}_j^{-1} (\hat{x}_i - \hat{x}_j) \ . \tag{2.31}$$

We have additionally used a nearest neighbor type criteria, where partitions are considered coupled if any sample from partition i is closer to the center of partition j then any sample from partition j. In practice, it is found that simply using the Euclidian distance between estimated states is sufficient and less computationally burdensome. The adaptive proposal method is summarized in Table 2.5.

It turns out that a refinement to the CP method (given by Morelande [83]) improves performance even further. In this method, those partitions that are deemed

Table 2.6: The modified adaptive proposal method.

1. Cluster targets into C groups
2. For each group $c = 1 : C$
   (a) if group C has one entry,
   Propose group c using IP method
   (b) else
   Propose group c using CP method
3. Remove or add partitions from selected particles as in Section 2.2.4
4. For each particle $p = 1, ..., N_{part}$
   $w_p^k = w_p^{k-1} * \frac{p(\mathbf{z}|\mathbf{X}_p)}{m_p \prod_{c=1}^{C} b_{p,c}}$

to be coupled are clustered according to the method of Section 2.2.6. This results in "partitions" that contain multiple targets – some with 2 targets, some with 3 targets, etc. Then instead of proposing each target individually, the clustered pairs (pairs, triplets, etc.) of targets are proposed all at once. This method is summarized in Table 2.6. Note that the idea of a partition containing multiple targets is also present in the work of Orton [36], although adaptively deciding partition boundaries and partition clustering is new to this work.

2.2.6 Permutation Symmetry and Partition Sorting

As discussed throughout the preceding sections, the permutation symmetry associated with the JMPD discussed in Section 2.1 is directly inherited by the particle filter representation of the JMPD. Each particle contains many partitions (as many as the number of targets it estimates exist in the surveillance region) and the permutation symmetry of JMPD is visible through the fact that the relative ordering of targets may change from particle to particle. We refer to the permutation symmetry in this context as partition swapping.

The fact that partitions are in different orders from particle to particle is of no

consequence when the object of interest is an estimate of the joint multitarget density. Each particle contributes the correct amount of mass in the correct location to the multitarget density irrespective of the ordering of its partitions.

However, the IP scheme requires particles be identically ordered. Furthermore, estimating the multitarget states from the particle filter representation of JMPD must also be done in a way that is invariant to permutations of the particles. Therefore, before estimating target states, we permute the particles so that each particle has the targets in the same order. We use the K-means algorithm [85] to cluster the partitions of each particle, where the optimization is done across permutations of the particles. This is a very light computational burden in practice for two reasons. First, those partitions that are not coupled are already consistently ordered and need not be involved in the clustering procedure. Second, since this re-ordering occurs at each time step, those partitions that are coupled are nearly ordered already, and so one iteration of the K-means algorithm is typically enough to find the best permutation.

We now give the details of the K-means algorithm applied to our setting. First, we state the notion of permutation symmetry precisely. Suppose a particle has T_p partitions labeled $t = 1 \cdots T_p$. A permutation π_p is a reshuffling of the labels, $\pi_p : i \rightarrow \pi_p(i)$. So a particle is defined

$$\mathbf{X}_p = [\mathbf{x}_{p,1}, \mathbf{x}_{p,2}, \cdots, \mathbf{x}_{p,T_p}] \ . \tag{2.32}$$

Under the permutation π_p is reordered to

$$\mathbf{X}_p = [\mathbf{x}_{p,\pi_p(1)}, \mathbf{x}_{p,\pi_p(2)}, \cdots, \mathbf{x}_{p,\pi_p(T_p)}] \ . \tag{2.33}$$

Denote by π a set of permutations for each particle, $\pi_p, p = 1 \cdots N_{part}$. We define the mean of the tth partition under the permutation π as

$$\bar{\mathbf{X}}_t(\pi) = \sum_{p=1}^{N_{part}} w_p \mathbf{X}_{p,\pi_p(t)} \ , \tag{2.34}$$

Table 2.7: The K-means algorithm for partition sorting.

1. Initialize with π = current ordering of partitions
2. Compute $\bar{\mathbf{X}}_t(\pi)$ for $t = 1 \cdots T_{max}$ using (2.34)
3. For each particle p, permute the particle (update π_p) to yield
$$\pi_p \leftarrow \arg\min_{\pi_p} \sum_{t=1}^{T_p} (\mathbf{X}_{p,\pi_p(t)} - \bar{\mathbf{X}}_t(\pi_p))^2$$
4. If no particles have changed permutation from π, quit.
Otherwise set $\pi = (\pi_1, \cdots, \pi_p, \cdots, \pi_{N_{part}})$ and go to 2

where it is understood that the summation is taken over only those particles that have partition t, and the weights are appropriately normalized to this subset. Further, define the χ^2 statistic as

$$\chi^2(\pi) = \sum_{p=1}^{N_{part}} \sum_{t=1}^{T_p} w_p (\mathbf{X}_{p,\pi_p(t)} - \bar{\mathbf{X}}_t(\pi_p))^2 \ . \tag{2.35}$$

To reorder the particles, the goal is to find the set of permutations π that minimize χ^2, i.e.

$$\hat{\pi} = \min_{\pi} \chi^2(\pi) \ . \tag{2.36}$$

The K-means algorithm is a well known method of approximately solving problems of this type. An initial permutation π is assumed and perturbations about that value are made to descend and find the locally optimal π. As mentioned earlier, re-ordering is done at each iteration of the algorithm, so the initial ordering is typically very close to the globally optimal ordering. Therefore, the K-means algorithm typically converges to the global optimum after a very small number of iterations (often 1). The algorithm is given in Table 2.7.

Notice that if the K-means algorithm fails to return the globally best reshuffling, this is not a serious problem. The main effect is that the CP algorithm will be used

more than is minimally necessary. This results in increased computation but no performance degradation. A secondary effect is that target estimates will be slightly incorrect. There will be only a minor error relative to the sensor resolution because a local minimum will at worst mix partitions that are very close together in sensor space.

2.2.7 State Estimation

The particles that approximate the JMPD can be used to generate estimates of the number of targets in the surveillance area and the states of the individual targets.

Equation (2.3) gives the expression for computing the probability that there are exactly T targets in the surveillance volume from the JMPD. To extract this estimate from the particle filter approximation, first define the indicator variable $I_p(T)$ for $p = 1...N_{part}$,

$$I_p(T) = \begin{cases} 1 & \text{if } T_p = T \\ 0 & \text{otherwise} \end{cases} . \tag{2.37}$$

Then the probability that there are T targets in the surveillance volume, $p(T|\mathbf{Z})$, is given by

$$p(T|\mathbf{Z}) \approx \sum_{p=1}^{N_{part}} I_p(T) w_p \ . \tag{2.38}$$

Hence, the estimate of the probability that there are T targets in the surveillance volume is merely the sum of the weights of the particles that have T partitions.

To compute the estimated state and covariance of target i, we first define a second indicator variable $\tilde{I}_p(i)$ that indicates if particle p has a partition corresponding to target i. This is necessary as each particle is a sample drawn from the JMPD and hence may have a different number of partitions (targets):

$$\tilde{I}_p(i) = \begin{cases} 1 & \text{if partition } i \text{ exists in particle } p \\ 0 & \text{otherwise} \end{cases} . \tag{2.39}$$

Note that the sorting procedure of Section 2.2.6 has identified an ordering of particles to allow $\tilde{I}_p(i)$ to be determined. This is largely a bookkeeping issue; particles are sorted so the partition tracking target i is in the same position in each particle. If a particular particle does not predict that target i exists, it's corresponding partition is "empty" and $\tilde{I}_p(i) = 0$. Defining the normalized weights to be

$$\hat{w}_p = \frac{w_p \tilde{I}_p(i)}{\sum_{l=1}^{N_{part}} \tilde{I}_l(i) w_l} \ , \tag{2.40}$$

so $\hat{w}_p$ is the relative weight of particle p, with respect to all particles containing a partition corresponding to target i. Also, $\hat{w}_p$ is zero for particles not tracking target i. Then the estimate of the state of target i is given by

$$\hat{\mathbf{X}}(i) = E[\mathbf{X}(i)] = \sum_{p=1}^{N_{part}} \hat{w}_p \mathbf{X}_{p,i} \ . \tag{2.41}$$

which is simply the weighted summation of the position estimates from those particles that are tracking target i. The covariance is given similarly as

$$\hat{\mathbf{\Lambda}}(i) = \sum_{p=1}^{N_{part}} \hat{w}_p (\mathbf{X}_{p,i} - \hat{\mathbf{X}(i)})(\mathbf{X}_{p,i} - \hat{\mathbf{X}(i)})' \ . \tag{2.42}$$

The indicator function $\tilde{I}_p(i)$ ensures that the summations in (2.41) and (2.42) are taken over only those particles that are tracking target i.

The permutation symmetry issue mentioned earlier comes to the forefront here. Notice that without a clustering on the partitions, it is not necessarily true that partition i of particle j is tracking the same target that partition i of particle $j+1$ is tracking. Therefore, taking a mean across a partition is meaningless. This is remedied by applying the clustering algorithm discussed in Section 2.2.6 before evaluation of (2.39) through (2.42).

2.2.8 Resampling

In the traditional method of resampling, after each measurement update, N_{part} particles are selected with replacement from $\mathbf{X}_p$ based upon the particle weights w_p. The result is a set of N_{part} particles that have uniform weight which approximate the multitarget density $p(\mathbf{X}|\mathbf{Z})$. Particles that do not correspond to measurements are eliminated – in particular, particles whose T_p value is not supported by measurements (too many or too few targets) are selected with low probability.

The main problem with resampling is that it reduces particle diversity. This leads to reduced performance for a fixed number of particles. Therefore, resampling is only be done when absolutely necessary.

The particular resampling that was used in this work is systematic resampling [11]. This resampling strategy is easily implemented, runs in order N_{part}, is unbiased, and minimizes the Monte Carlo variance. Many other resampling schemes and modifications are presented in the literature [21]. Of these methods, we have found that adaptively choosing at which time steps to resample [86] based on the number of effective particles leads to improved performance while reducing compute time.

All results presented herein use the method of [86] to determine which times to resample and use systematic resampling [11] to perform resampling. We have also found that Markov Chain Monte Carlo (MCMC) moves using a Metropolis-Hastings scheme [21] leads to slightly improved performance in our application. However, the performance improvement does not warrant the increased computation and so it is not used here.

2.2.9 Multiple Model Particle Filtering

Real targets are often poorly described by a single kinematic model. Target behavior may change dramatically – e.g., a target may stop moving or begin rapid acceleration. We refer to these methods of target behavior as target "modes". In the literature, the Interacting Multiple Model (IMM) algorithm [9] is used to address this. In short, the IMM algorithm uses a set of models to describe target behavior. Each model is called a mode and characterizes to some degree the way a target is behaving. For example, there may be a model for targets that are moving with constant velocity, one for targets that are accelerating and one for targets that are stopped. In addition, the IMM uses models of how the mode of a target evolves over time, e.g., how likely a moving target is to stop and vice versa.

The IMM algorithm estimates on-line the target mode and uses it for filtering. The designer selects a set of M models or modes $m = 1 \cdots M$ that represent all possible priors on motion of the target (e.g., stopped, accelerating, performing a coordinated turn). Associated with each model m is the mode probability (probability the target is following this mode at the current time). At initialization, mode probabilities are given based on prior knowledge. While the filter tracks the target, mode probabilities are continuously re-estimated online.

The target mode is typically assumed to evolve in a Markov fashion, specified *a priori* by transition probabilities π_{ij} between target mode i and j. Sensor measurements allow the filter to update the estimate of the mode probabilities at each time step. A sub-filter is associated with each of the M modes. The sub-filters estimate the state $\mathbf{x}$ conditioned on both the measurements $\mathbf{Z}$ and the mode i, i.e. the i^{th} sub-filter estimates $p_i(\mathbf{x}|\mathbf{Z})$.

When a particle filter is used as the target tracker, the IMM algorithm is especially

Table 2.8: Generic interacting multiple model particle filter propagation.

Time Update
• Select the mode : $m^{k+1} \sim \pi_{m^k, m^{k+1}}$
• Propose target state : $\mathbf{x}^{k+1} \sim q_{m^{k+1}}(\mathbf{x}^{k+1} \vert \mathbf{x}^k, \mathbf{z})$
Measurement Update
• Update weight : $w^{k+1} = w^k \frac{p(\mathbf{z} \vert \mathbf{x}^{k+1}) p(\mathbf{x}^{k+1} \vert \mathbf{x}^k)}{q(\mathbf{x}^{k+1} \vert \mathbf{x}^k, \mathbf{z})}$

simple. Each particle is expanded to contain a mode estimate for each target. The particle is propagated forward in time according to the dynamics implied by the modes of the targets. Transitions between modes happen for each target according to π. The weighting and resampling process work to reinforce modes that are in agreement with measurements at the expense of those that are not. Specifically, for each particle at time k (which contains an estimate of the mode m^k and state $\mathbf{x}^k$) a particle is proposed at time $k+1$ according to Table 2.8.

2.3 Simulation Results

2.3.1 Introduction

We illustrate the performance of our multitarget tracking scheme by considering the following model scenario.

A set of ground targets move in a $5000m \times 5000m$ surveillance area. Targets are modeled using the four-dimensional state vector $\mathbf{x} = [x, \dot{x}, y, \dot{y}]$. The target motion in the simulation is taken from a set of recorded data based on GPS measurements of vehicle positions collected as part of a battle training exercise at the Army's National Training Center (see Figure 2.1). This battle simulation provides a large number of real vehicles, including army HMMWVs, armored personnel carriers, tanks, and the like. The vehicles follow a prescribed trajectory over natural terrain. Based on an

empirical fit to the data, we found that a nearly constant velocity model (see (2.4)) was adequate to model the behavior of the vehicles for these simulation studies and is therefore used in all experimental results presented herein. In another study [77], we have found that a multiple model particle filter with modes corresponding to nearly constant velocity, rapid acceleration, and stationarity provides slightly more efficient filtering.

We use the idealized sensor described in Section 2.1.2. The sensor scans a fixed rectangular region of 50×50 pixels, where each pixel represents a $100m \times 100m$ area on the ground plane. The sensor returns Rayleigh-distributed measurements in each pixel, depending on the number of targets that occupy the pixel. In some simulations, we consider pre-thresholded measurements and therefore energy is returned corresponding to each measured pixel according to (2.14). In other simulations, we consider thresholded measurements in which case binary returns are received according to (2.15).

We present the results of 5 simulation studies here. First, in Section 2.3.2, we illustrate the benefit of the adaptive proposal scheme detailed in Section 2.2.5. We contrast the performance of the CP, IP, and AP methods in two scenarios, one where targets are always well separated and one more realistic scenario where targets frequently interact. Second, in Section 2.3.3, we qualitatively evaluate the performance difference when using pre-thresholded measurements versus thresholded measurements. Third, in Section 2.3.4, we evaluate the ability of the tracker to determine the number of targets when the number is initially unknown. Fourth, in Section 2.3.5, we investigate the computational burden of the algorithm and how it scales with number of targets. Fifth, in Section 2.3.6, we illustrate partition swapping when two targets cross. The scenario is contrasted with and without partition sorting as

described in Section 2.2.6.

2.3.2 Adaptive Proposal Results

In Figure 2.2, we compare the performance of the Independent Partitions (Table 2.3), Coupled Partitions (Table 2.4), and Adaptive Partitions (Table 2.5) proposal schemes presented here with that of the traditional scheme of sampling from the kinematic prior (Table 2.2) in terms of RMS tracking error. In this example we use 3 targets with motion taken from real recorded trajectories. The targets remain close in sensor space for about 50% of the time. Thresholded measurements with $P_d = 0.5$ are used and the SNR parameter λ is varied from 1 to 21.

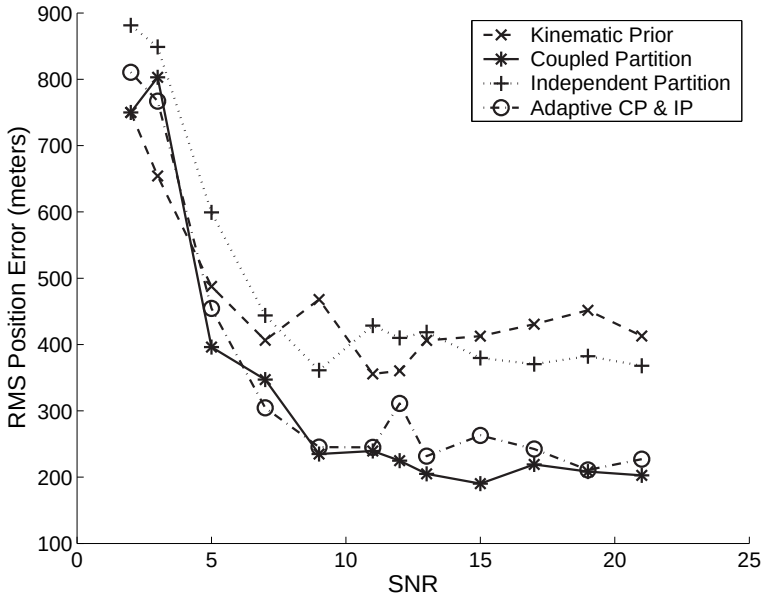


Figure 2.2: The performance of the coupled partition, independent partition, and adaptive partition schemes in comparison to simply using the kinematic prior. Performance is measured in terms of RMS position error in meters. In this simulation, we have extracted 3 targets from our large database of real recorded target trajectories. The targets were chosen so as to spend approximately one-half of the simulation in close proximity. The IP algorithm used alone is inappropriate during target crossings and so performs poorly here. The CP algorithm is always appropriate, but computationally demanding. The AP algorithm adaptively switches between IP and CP resulting in good performance at reduced computation.

Due to partition swapping, the IP method is inappropriate during target crossings

and hence the tracker using IP has poor performance. The CP method makes no assumption about the independence of the targets and therefore performs very well, although at significantly higher computational cost. Most importantly, the adaptive method, which uses IP on partitions that are deemed independent and CP otherwise, performs nearly as well as the CP method itself. AP achieves approximately a 50% reduction in computational burden (measured by floating point operations) as compared to the CP method alone (see Table 2.9).

In this simulation, we have extracted 3 targets from our large database of real recorded target trajectories. The targets were chosen so that they spent approximately one-half of the simulation in close proximity. The AP algorithm correctly chooses to use IP during the half of the simulation where targets well separated and CP during the other half, which results in the stated reduction in computation.

Table 2.9: Floating point operations (as measured by MatLab) for KP, CP, IP, and AP Methods.

Particle Proposal Method	Floating Point Operations
Coupled Partition	1.25e+8
Independent Partition	6.74e+6
Adaptive Partition	5.48e+7
Kinematic Prior	6.32e+6

As a means of directly comparing the IP and CP methods with the kinematic prior (KP), we construct a second model problem. We consider five well separated targets, and look at the performance in Figure 2.3. For the purposes of this model problem, we restrict target motion to be linear, measurement to state coupling to be linear, and the noise processes to be Gaussian. We use the motion model given by (2.4) both for the simulation of target motion and in the filter. In this case we can use the Kalman filter as a bound. It is of course not necessary to make these assumptions for the particle filter. In fact, the strength of the particle filter (and

nonlinear filtering in general) approach is that no linearity/Gaussianity assumptions are needed. However, we have restricted the problem in this manner here so that we can compute an asymptotic performance bound and show that the particle filter implementation indeed reaches the bound.

More general performance bounds, which apply to the discrete time nonlinear filtering problems such as that of Figure 2.2 are available in the literature [87][88]. However, bounds require knowledge of the true target kinematics, $p(\mathbf{X}^k, T^k|\mathbf{X}^{k-1}, T^{k-1})$, which we do not have. As stated earlier, the simulations involve real recorded target motion and hence we do not know the kinematic model precisely.

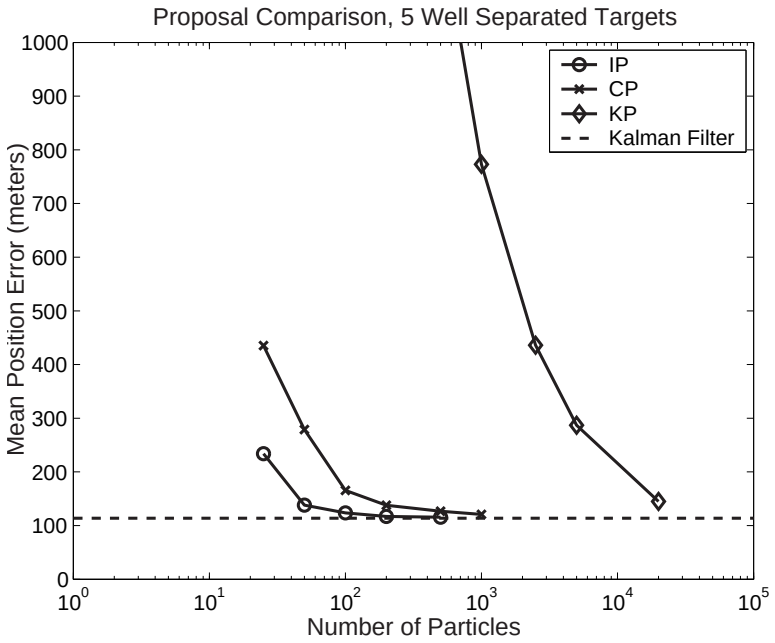


Figure 2.3: The performance of the proposal schemes considered here compared to the performance bound. For the purposes of this example, we consider five well separated targets with linear motion, linear state-to-measurement coupling, and Gaussian initial density. Therefore, the Kalman filter is optimal and provides a performance bound. We see that the coupled partition method achieves similar performance as the kinematic proposal with a factor of 100 fewer particles. Furthermore, the independent partition method achieves similar performance as the kinematic proposal with a factor of 1000 fewer particles.

The CP method is shown with a particular choice of R, $R = 10$. We see that the CP method reduces the number of particles required (as compared to the Kinematic

Prior) by a factor of approximately 100. It was seen earlier in Table 2.9 that the computational increase of CP is approximately a factor of 20. This tradeoff makes CP a more efficient strategy than simply increasing the number of particles. It can be seen that the IP technique reduces the number of particles needed by between two and three orders of magnitude as compared to the traditional technique (KP). Since the work per particle to perform IP is nearly identical to that of sampling from the kinematic prior, IP actually reduces computational burden by two to three orders of magnitude when targets are well separated.

These simulations are the result of particular choices of the plant noise and measurement noise parameters. The number of particles required to reach the Kalman filter bound is sensitive to these choices. Specifically, as the ratio of plant noise to measurement noise increases, the number of particles to reach the bound increases. However, the relative performance of the IP, CP and KP algorithms remains consistent.

2.3.3 The Value of Not Thresholding

We investigate here the gain from using pre-thresholded measurements in the multitarget tracking scenario. One of the strengths of our association-free method is the ability to directly include pre-thresholded measurements into the filter through the likelihood $p(\mathbf{z}|\mathbf{X}, T)$.

In this simulation, we study three real targets chosen from our database and benchmark the performance of the tracker versus SNR (λ) for thresholded measurements with $P_d = 0.1 \cdots 0.9$. At a constant SNR, as P_d is reduced, so is P_f according to the relation given in (2.15). The performance of the algorithm versus SNR and P_d is given in Figure 2.4.

We contrast the performance of the algorithm using thresholded measurements

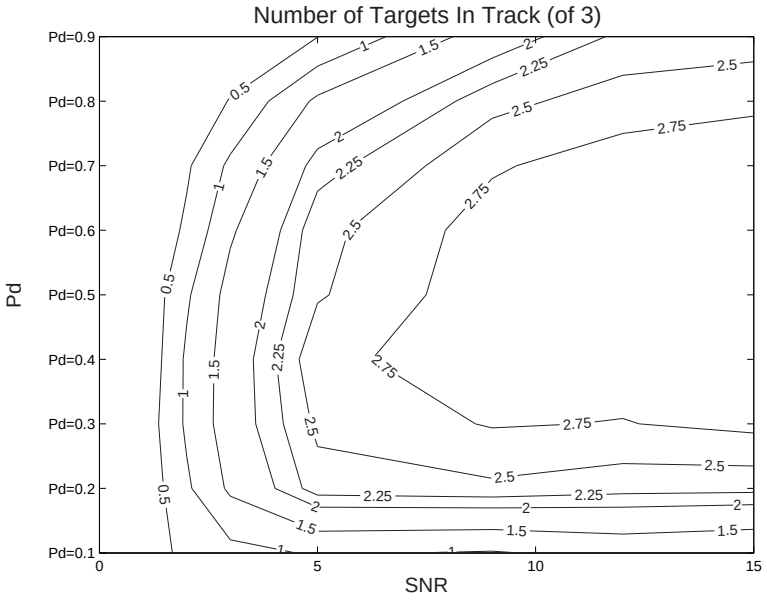


Figure 2.4: A contour plot showing the number of targets tracked versus P_d and SNR when using thresholded measurements. The simulation consists of three targets taken from the collection of real targets. In a real system, the operator has a choice of how to set the threshold, essentially choosing the P_d and P_f. Optimal performance occurs near $P_d = 0.4$.

with the performance when using pre-thresholded measurements at the same set of SNR values. Figure 2.5 is a plot showing the performance of the algorithm using thresholded measurements at $P_d = 0.4$ (the best performance from Figure 2.4) and the algorithm using pre-thresholded measurements in terms of the number of targets successfully tracked. We see that pre-thresholded measurements provide similar tracking performance at an SNR of 1 as the thresholded measurements provide at an SNR of 5, for a gain of about 7dB from not thresholding the measurements.

2.3.4 Unknown Number of Targets

We demonstrate the ability of the filter to determine the number of targets and their states when started with no knowledge of either. For this simulation, the existence probability at initialization (see Section 2.2.2) for each of the sensor cells is set to 0. The birth rate is assumed to be constant spatially at .02 and the death

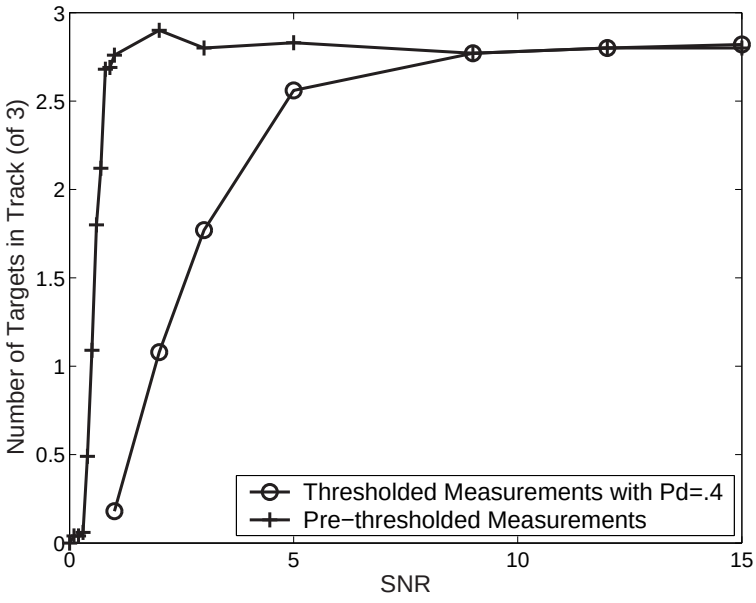


Figure 2.5: A plot comparing the performance of the tracker using thresholded measurements with the performance using pre-thresholded measurements in a three target experiment. The simulation consists of three targets taken from the collection of real targets. Using pre-thresholded measurements provides similar tracking performance at $SNR = 1$ as the thresholded measurements provide at $SNR = 5$.

rate is assumed to be .005.

We measure the performance of the algorithm in two ways. First, we compare the estimated number of targets to the true number of targets, where the estimated number of targets at time k is defined as

$$\hat{T}^k = \sum_{T=0}^{\infty} T \int_{\mathbf{X}} d\mathbf{X} p(\mathbf{X}, T|\mathbf{Z}) \approx \sum_{p=1}^{n} w_p T_p \tag{2.43}$$

Second, we use the ground truth to calculate the number of actual targets that are successfully tracked by the filter. For each of the hypothesized target t, we have an estimate of the target state as

$$\hat{x}_t^k = \int \mathbf{x}_t d\mathbf{x}_1 \cdots \mathbf{x}_t p(\mathbf{x}_1 \cdots \mathbf{x}_T|\mathbf{Z}) \approx \sum_{p=1}^{n} \hat{w}_p \mathbf{X}_{p,t} \tag{2.44}$$

where $\hat{w}_p$ is normalized to sum to one over all particles that contain partition t and the partitions are ordered according to the algorithm of Section 2.2.6. The target estimates are then matched up with the ground truth to give a measure of how many

true targets are being successfully tracked, which we denote $T_{tracked}$. Note this metric is penalized for both targets that should have been detected but weren't as well as targets that were successfully detected but then poorly tracked. The two metrics $\hat{T}^k$ and $T_{tracked}$ taken in combination allow for determination of the number of false targets initiated as well as the number of true targets not under track.

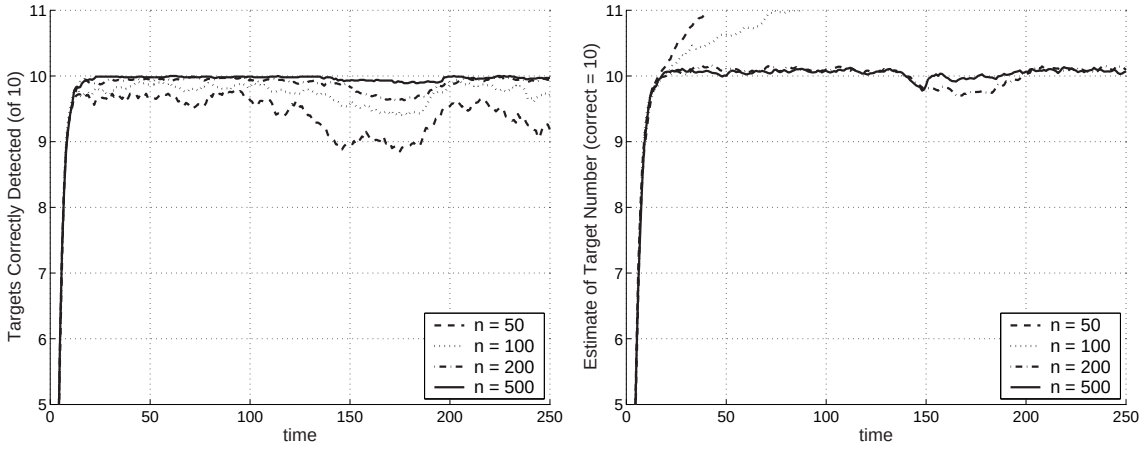


Figure 2.6: The performance of the particle filter implementation of JMPD for detecting and tracking ten real targets versus number of particles. For this experiment, the SNR is $10dB$, the removal rate is .005 and the arrival rate is .02. Ideal performance is $\hat{T} = 10$ and $T_{tracked} = 10$ at all times. The dip in performance between time 150 and 200 represents a period of time where two targets occupied the same detection cell and hence the estimates of target position were not as good.

We show in Figures 2.6 and 2.7 the performance of particle filter based implementation of JMPD at detecting and tracking 10 real targets. In this simulation, ten targets were drawn from the database of real target trajectories. All ten targets are present at filter startup, and the filter has no prior information about the number or states of the targets. Ideal performance is for the filter to estimate $\hat{T} = 10$ targets and have $T_{tracked} = 10$ targets in track at all times.

2.3.5 Computational Considerations

Using a MatLab/C hybrid coding implementation on an off-the-shelf 3GHz Linux box, we find that the AP method is able to track 10 real targets with scans of the

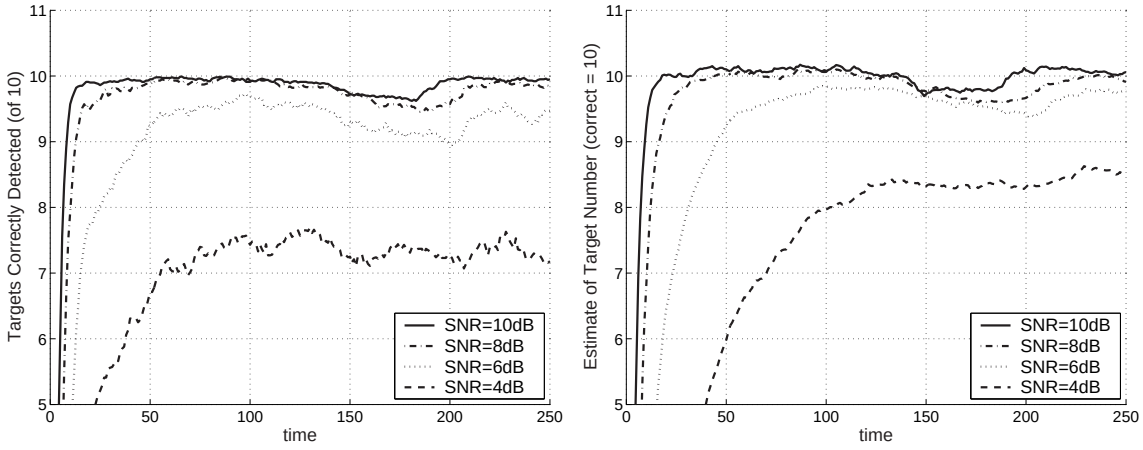


Figure 2.7: The performance of the particle filter implementation of JMPD for detecting and tracking ten real targets versus signal to noise ratio. For this experiment, the number of particles is 200, the removal rate is .005 and the arrival rate is .02. Ideal performance $\hat{T} = 10$ and $T_{tracked} = 10$ at all times. The dip in performance between time 150 and 200 represents a period of time where two targets occupied the same detection cell and hence the estimates of target position were not as good.

surveillance area coming in once per second faster than real time. Figure 2.8 shows the tracking performance when using particle filter implementation of JMPD on ten real targets. The plot is averaged over 50 trials, where in each trial a random set of 10 targets is chosen from our large database of real targets.

One factor that effects computation is the number of coupled targets. This effect can have a greater impact on computational complexity then the number of targets. When targets move close together, their coupling must be explicitly modeled and the CP algorithm becomes necessary. This algorithm is significantly more computationally demanding then the IP method.

In Figure 2.9, we show the computational runtime results of simulations where $1 \cdots 10$ targets are tracked. We include for reference the average number of coupled targets during the simulations. For each trial, we select t targets at random from our collection of real recorded data. Depending on which targets are selected, there may or may not be groups of targets that are close in sensor space during the simulation.

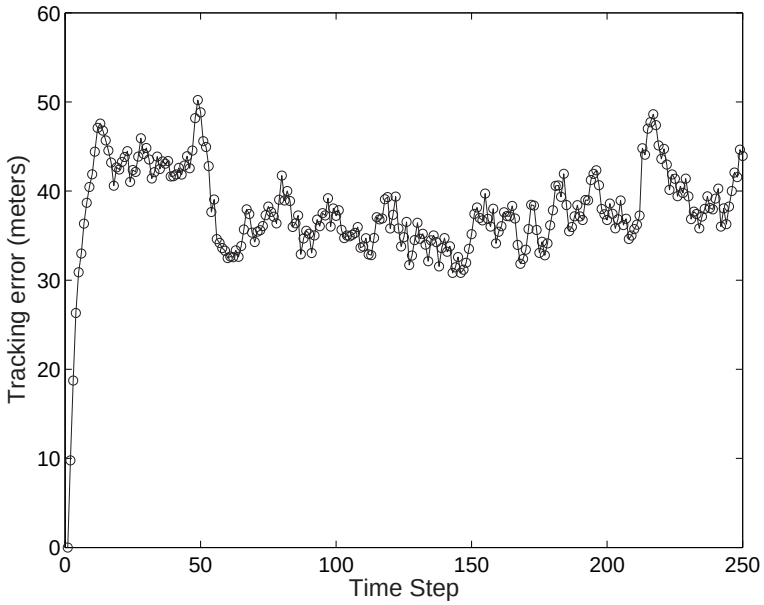


Figure 2.8: The performance of the particle filter implementation of JMPD when tracking ten real targets. This set of targets contains two convoys (targets following each other closely throughout the simulation), one of four targets and one of three targets. For each simulation, at each time step tracking error is measured as the mean track error for the ten targets. The plot shows the median tracking error across all 50 simulations. The filter is initialized with the true target locations and so initial tracking error is 0. Steady state tracking error is on the order of 40 meters. As mentioned earlier, the sensor measures 100m x 100m resolution cells on the ground. The particle filter implementation of JMPD uses 250 particles which allows near real time tracking.

This fact results in a different level computational complexity depending on the set of targets chosen. For this reason, the plot in Figure 2.9 is averaged over 50 trials, each consisting of a random draw of the targets from the real target motion database described earlier.

2.3.6 Partition Swapping

Partition swapping, as discussed in Section 2.2.6, is the phenomenon where different particles have different orderings of the various targets. This ordering difference occurs because of the permutation symmetry of the JMPD (and all multitarget tracking problems). Specifically, the multitarget states $\mathbf{X} = \left(\mathbf{x}_1, \mathbf{x}_2\right)$ and $\mathbf{X}' = \left(\mathbf{x}_2, \mathbf{x}_1\right)$ refer to the same multitarget state and are hence identical.

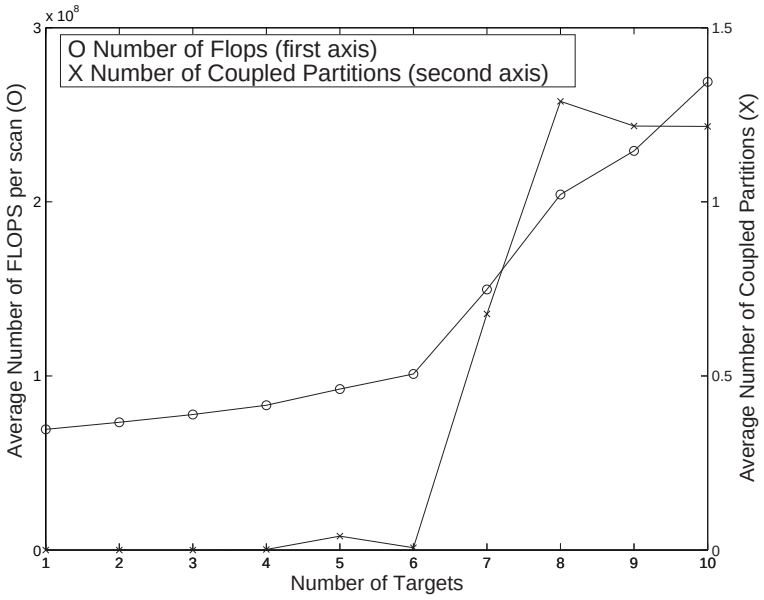


Figure 2.9: Floating point operations (as measured by MatLab) versus number of targets for a pure MatLab implementation of the multitarget particle filter. One factor that effects the computations required is the number of closely spaced targets, as the coupling most be modeled explicitly and the CP algorithm becomes necessary. We include for reference here the average number of coupled targets over all simulations.

Partition sorting is implementationally useful for two reasons. First, when partitions are identically ordered, computationally efficient algorithms such as the IP method may be used. Second, estimates of target state may be directly computed using particle locations and weights by simply averaging over partitions. If the partitions are not identically ordered, the filter still performs estimation of the JMPD precisely but requires the more computationally demanding CP algorithm. Furthermore, the sensor management algorithms (to be described in Chapter III) are invariant to partition ordering. Therefore, in this sense partition sorting is a cosmetic manipulation rather than a fundamental requirement of the algorithm.

We illustrate below the effect of partition swapping on a sample vignette. When targets are close together, measurement-to-target ambiguity may result in the partitions of individual particles being reordered. In Figure 2.10, we give a 9 time-step vignette which includes a target crossing. Initially, the targets are well separated and

identically ordered (e.g., $Time = 44$) and the IP method is used for particle proposal. When the targets cross ($Time = 60$), partition swapping occurs and the CP method must be used. Without partition sorting using the K-means algorithm of Section 2.2.6, this swapping persists even after the targets separated and the CP method must be used even at $Time = 84$. This results in an inefficient algorithm, as the CP method is more computationally demanding. Note however, that the tracking is still performed acceptably, it is only the estimates from the filter and the computational efficiency that are degraded when the partitions are unordered.

In Figure 2.11 we show the same vignette as in Figure 2.10, but this time we use the partition sorting algorithm outlined in Section 2.2.6 at each time step. While the CP method must still be used when the targets are occupying the same detection cell, when they move out ($Time = 72$) the IP method may be used again. The partition sorting allows for the more computationally efficient IP method to be used by reordering the particles appropriately.

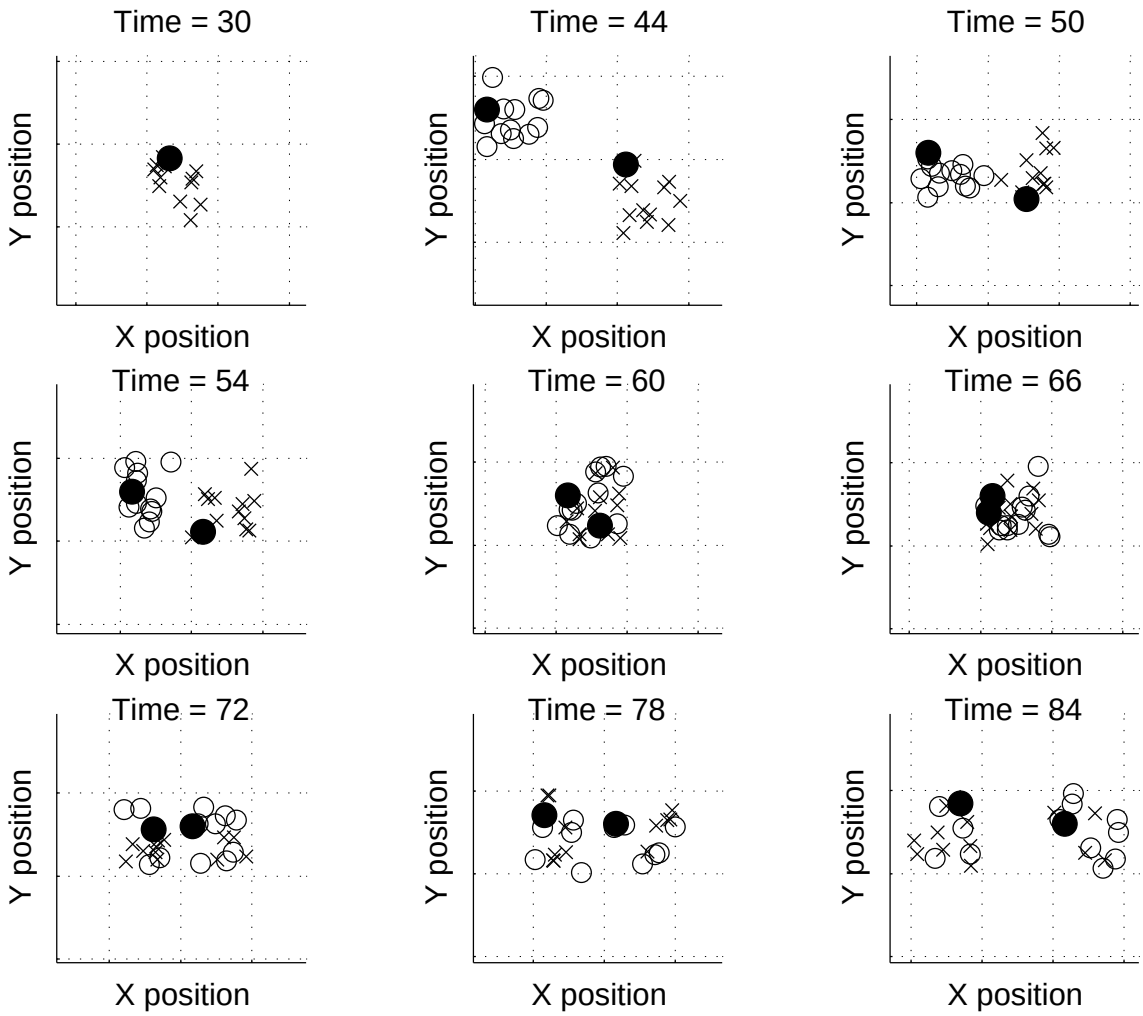


Figure 2.10: The partition swapping present in the particle filter implementation of JMPD when partition sorting is not done. True target locations are indicated by a solid circle. At $Time = 30$ only one target is visible in the plot window. At $Time = 44$, both targets can be seen and the two partitions for each particle, plotted with $\times$ and $\circ$, are well separated. From $Time = 60$ to $Time = 66$, the two targets occupy the same detection cell. At $Time = 84$, some partition swapping has occurred, indicated by the fact that there are mixtures of $\times$ and $\circ$ corresponding to each target location.

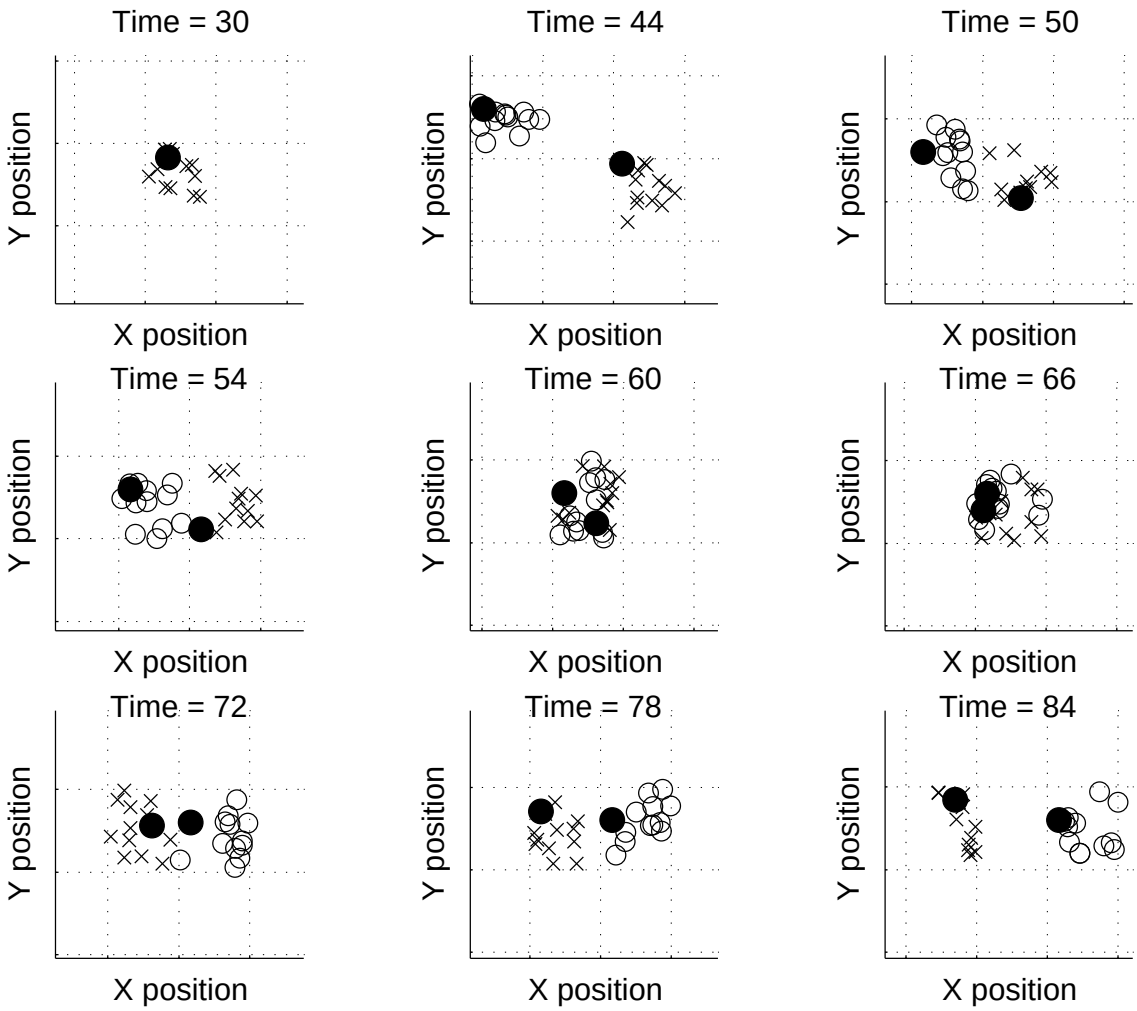


Figure 2.11: The ordering of the partitions when K-means partition sorting is sone at each time step. This figure illustrates the same multitarget tracking scenario as in Figure 2.10, except here we perform partition sorting at each time step. True target locations are indicated by a solid circle. At time 30 only one target is visible in the plot window. At $Time = 44$, both targets can be seen and the two partitions for each particle, plotted with $\times$ and $\circ$, are well separated. From $Time = 60$ to $Time = 66$, they occupy the same detection cell. The targets move apart starting at $Time = 72$. Notice that the partition swapping visible in Figure 2.10 at $Time = 72$ through $Time = 84$ is avoided here because of partition sorting.

CHAPTER III

Information Based Sensor Mangement

In this chapter, we present our information-based strategy for myopic sensor resource allocation. As mentioned earlier, a good measure of the quality of a sensing action is the reduction in entropy of the posterior distribution that is induced by the measurement. Since we wish to determine the best sensing action to take before actually executing it, we require a measure of the *expected* reduction in entropy that a sensing action will produce. This is done by first enumerating all possible sensing actions. A sensing action may consist of choosing a particular sensor mode (e.g. SAR mode or GMTI mode), a particular dwell point/pointing angle, or a combination of the two. Next, the *expected* information gain is calculated for each of the possible actions, and the action that yields the maximum expected information gain is taken. The measurement received is used to update the JMPD, which is in turn used to determine the next measurement to make.

The chapter proceeds as follows. In Section 3.1, we present the Rényi divergence as a means of measuring the change in information between two densities along with a theoretical motivation behind using this metric for sensor resource allocation. Second, in Section 3.2, we provide the details of using the Rényi divergence for sensor scheduling in the JMPD setting. We give first the Rényi divergence in the general

JMPD setting, followed by the simplification due to our particle filter representation of the JMPD, and then explicitly give the form of the expected Rényi divergence. We conclude Section 3.2 with a note on the selection of the parameter α in the Rényi divergence. Next, in Section 3.3, we discuss alternate but related formulations of the scheduling metric including the weighted Rényi divergence and the Rényi divergence between marginalized JMPDs. Finally, in Section 3.4, we provide several detailed simulations showing the performance of this method in comparison to simple periodic scanning and several other sensor management strategies.

3.1 The Rényi Divergence

In this work, the calculation of information gain between two densities p_1 and p_0 is done using the Rényi information divergence [61][89], (also known as the α-divergence):

$$D_\alpha(p_1||p_0) = \frac{1}{\alpha - 1} ln \int p_1^\alpha(x) p_0^{1-\alpha}(x) dx \ . \tag{3.1}$$

The adoption of the Rényi information divergence as a sensor scheduling criterion can be motivated by universal hypothesis testing results of large deviation theory [90][91]. Specifically, consider the problem of testing between the hypotheses

$$\begin{aligned} H_0 &: p(\mathbf{X}^k, T^k|\mathbf{Z}^k) = p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1}) \\ H_1 &: p(\mathbf{X}^k, T^k|\mathbf{Z}^k) \neq p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1}) \end{aligned} \tag{3.2}$$

based on an i.i.d. sample $\{\mathbf{X}_{(j)}^k\}_{j=1}^n$ from the posterior $p(\mathbf{X}^k, T^k|\mathbf{Z}^k)$, e.g., as generated by the particle filtering algorithm described earlier. H_1 is the hypothesis that the new measurement has changed the target state density, while H_0 is the hypothesis that the new measurement has not changed the target state density. The performance of any test of H_0 versus H_1 is specified by its receiver operating characteristic (ROC)

(α_n, β_n) where we have defined the false alarm probability $\alpha_n = p(\text{decide } H_1 \,|H_0)$ and the miss probability $\beta_n = p(\text{decide } H_0 \,|H_1)$. If the true posterior distribution under H_1 were known, the Neyman-Pearson likelihood ratio test (LRT), parameterized by a decision threshold γ, is optimal in the sense of achieving minimum β_n for any specified level α_n (determined by γ).

Under broad assumptions, for large n the error rates of the optimal test satisfy Theorem 3.4.3 in [90],

$$\frac{1}{n}\log \alpha_n \approx - \sup_{\alpha\in[0,1]} \{\alpha\gamma - (1-\alpha)D_\alpha(p_1\|p_0)\} \tag{3.3}$$

$$\frac{1}{n}\log \beta_n \approx - \sup_{\alpha\in[0,1]} \{-\alpha\gamma - (1-\alpha)D_\alpha(p_0\|p_1)\} \tag{3.4}$$

where p_0 and p_1 denote the posterior distribution $p(\mathbf{X}^k, T^k|\mathbf{Z}^k)$ under H_0 and H_1, respectively.

In particular, if one selects $\gamma = 0$ then the two error rate exponents (right sides (3.3) and (3.4)) are identically $-(1-\alpha^*)D_{\alpha^*}(p_1\|p_0)$ for some $\alpha^* \in [0,1]$. Thus the Rényi α-divergence specifies the error exponents of the Neyman-Pearson optimal test.

For the composite hypotheses (3.2) the generalized likelihood ratio test (GLRT) of H_0 versus H_1 is asymptotically (large n) optimal and can be implemented by thresholding an empirical estimate of the error rate exponent (3.3) to achieve a specified level of false alarm α_n (Theorem 7.1.3 in [90]). This lends strong theoretical justification for using the Rényi divergence sensor selection criterion proposed here.

Returning to (3.1), we note that the α parameter may be used to adjust how heavily one emphasizes the tails of the two distributions p_1 and p_0. In the limiting case of $\alpha \to 1$ the Rényi divergence becomes the commonly utilized Kullback-Leibler

(KL) discrimination (3.5).

$$\lim_{\alpha \to 1} D_\alpha(p_1||p_0) = \int p_0(x) ln \frac{p_0(x)}{p_1(x)} dx \ . \tag{3.5}$$

If $\alpha = 0.5$, the Rényi information divergence becomes the Hellinger affinity $2ln \int \sqrt{p_1(x)p_0(x)}dx$, which is related to the Hellinger-Battacharya distance squared [92] via

$$D_{Hellinger}(p_1||p_0) = 2\Big(1 - exp\Big(.5D_{\frac{1}{2}}(p_1||p_0)\Big)\Big) \ . \tag{3.6}$$

Ultimately, a specific value of α must be chosen. In section 3.2.4, we show using empirical evidence as well as theoretical results for close densities that $\alpha = 0.5$ provides the maximum discriminatory ability and as such is the right choice for our application.

3.2 Application of the Rényi Divergence in the JMPD Setting

In this section, we present the details of the mathematics behind using the Rényi divergence as a method of scheduling sensors [93][94]. We proceed by first specializing the Rényi divergence to the case where the densities under consideration are the prior and posterior JMPD. We then show the simplifications afforded by the particle filter representation of JMPD. Finally, we explicitly write out the form of the expected Rényi divergence in the JMPD-PF setting.

3.2.1 Rényi Divergence Between the Prior and Posterior JMPD

The function D_α in (3.1) is a measure of the divergence between the densities p_0 and p_1. In our application, we are interested in computing the divergence between the predicted density $p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})$ and the updated density after a measurement is made, $p(\mathbf{X}^k, T^k|\mathbf{Z}^k)$. Therefore, we write

$$D_\alpha\left(p(\cdot|\mathbf{Z}^k)||p(\cdot|\mathbf{Z}^{k-1})\right) = \frac{1}{\alpha - 1} ln \int_{\mathbf{X}} p(\mathbf{X}^k, T^k|\mathbf{Z}^k)^\alpha p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})^{1-\alpha} d\mathbf{X}^k \tag{3.7}$$

Where, as earlier, the multitarget state $\mathbf{X}$ is a vector of variable dimension. The symbol $\int_{\mathbf{X}} f(\mathbf{X})d\mathbf{X}$ is used as short hand notation to denote the integral over the domain. This can be precisely written as

$$\int_{\mathbf{X}} d\mathbf{X} f(\mathbf{X}, T) \doteq \sum_{T=0}^{\infty} \int d\mathbf{x}_1 ... \mathbf{x}_T f(\mathbf{x}_1 \cdots \mathbf{x}_T, T) \ . \tag{3.8}$$

Using Bayes' rule applied to the JMPD (2.5) and simple algebra, we can successively simply D_α as

$$D_\alpha \left(p(\cdot|\mathbf{Z}^k) || p(\cdot|\mathbf{Z}^{k-1}) \right) = \frac{1}{\alpha - 1} ln \int_{\mathbf{X}} \left(\frac{p(\mathbf{z}^k|\mathbf{X}^k, T^k, m) p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})}{p(\mathbf{z}^k|\mathbf{Z}^{k-1}, m)} \right)^{\alpha} p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})^{1-\alpha} d\mathbf{X}^k \tag{3.9}$$

and

$$D_\alpha \left(p(\cdot|\mathbf{Z}^k) || p(\cdot|\mathbf{Z}^{k-1}) \right) = \frac{1}{\alpha - 1} ln \frac{1}{p(\mathbf{z}^k|\mathbf{Z}^{k-1}, m)^{\alpha}} \int_{\mathbf{X}} p(\mathbf{z}^k|\mathbf{X}^k, T^k, m)^{\alpha} p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1}) d\mathbf{X}^k \ . \tag{3.10}$$

To make the formulation explicitly clear, we use $p(\mathbf{z}^k|\mathbf{Z}^{k-1}, m)$ to denote the distribution on sensor outputs at time k given the set of previous measurements $\mathbf{Z}^{k-1}$ and the fact that the action m was taken.

3.2.2 Rényi Divergence when the JMPD is Represented by the Multitarget Particle Filter

The time-updated (prediction) JMPD $p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})$ is approximated by a particle filter consisting of a set of samples $\mathbf{X}_p$ and associated weights w_p. This approximation reduces the divergence calculation from an integral to a sum over particles. Specifically, our particle filter approximation of the density (2.21) represents the JMPD $p(\mathbf{X}, T|\mathbf{Z})$ by a set of N_{part} samples with weight w_p, $p = 1...N_{part}$, reducing (3.10) to

$$D_\alpha \left(p(\cdot|\mathbf{Z}^k) || p(\cdot|\mathbf{Z}^{k-1}) \right) = \frac{1}{\alpha - 1} ln \frac{1}{p(\mathbf{z}^k|\mathbf{Z}^{k-1}, m)^{\alpha}} \sum_{p=1}^{N_{part}} w_p p(\mathbf{z}|\mathbf{X}_p, m)^{\alpha} \ , \tag{3.11}$$

where $p(\mathbf{z}^k|\mathbf{Z}^{k-1}, m)$ is calculated as

$$p(\mathbf{z}^k|\mathbf{Z}^{k-1}, m) = \sum_{p=1}^{N_{part}} w_p p(\mathbf{z}|\mathbf{X}_p, m) \ . \tag{3.12}$$

The reduction from an integral over $\mathbf{X}$ to a sum over particles in (3.11) comes about because the particle filter has non-zero probability mass in only a small number of possible multitarget states, precisely the states $\{\mathbf{X}_p, p = 1...N_{part}\}$.

This gives us a compact expression for computing the Rényi Divergence between a prior and posterior JMPD in the case where our multitarget particle filtering method is used to represent the densities. If the measurement $\mathbf{z}$ had already been made, this could be used to evaluate the information gain that the measurement has provided.

3.2.3 The Expected Rényi Divergence for a Sensing Action

Our real aim is to choose the sensing action to take *before actually receiving* the measurement $\mathbf{z}$. Specifically, we would like to choose to perform the measurement that makes the divergence between the current density and the density after a new measurement as large as possible. This indicates that the sensing action has maximally increased the information content of the measurement updated density, $p(\mathbf{X}^k, T^k|\mathbf{Z}^k)$, with respect to the density before a measurement was made, $p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})$. However, we cannot choose the action that maximizes the divergence as we do not know the outcome of the action before taking it.

We propose, then, as a method of sensor management to calculate the expected value of (3.11) for each of the M possible sensing actions and to choose to take the action that maximizes the expectation. In this notation m ($m = 1...M$) will refer to any possible sensing action under consideration, including but not limited to sensor mode selection and sensor beam positioning. In this manner, we say that we are making the measurement that maximizes the expected gain in information as

measured by the Rényi Divergence.

The expected value of (3.11) may be written as an integral over all possible outcomes $\mathbf{z}$ when performing sensing action m as

$$\langle D_\alpha \rangle_m = \int d\mathbf{z}^k p(\mathbf{z}|\mathbf{Z}^{k-1}, m) D_\alpha \left(p(\cdot|\mathbf{Z}^k) || p(\cdot|\mathbf{Z}^{k-1}) \right) \quad . \tag{3.13}$$

In the special case where measurements are thresholded (binary) and are therefore either detections or no-detections (i.e. $z = 0$ or $z = 1$), this integral reduces to

$$\langle D_\alpha \rangle_m = \frac{1}{\alpha - 1} \sum_{z=0}^{1} p(z|\mathbf{Z}^{k-1}, m) ln \frac{1}{p(z|\mathbf{Z}^{k-1}, m)^\alpha} \sum_{p=1}^{N_{part}} w_p p(z|\mathbf{X}_p, m)^\alpha \quad . \tag{3.14}$$

Computationally, the value of (3.14) can be calculated for M possible sensing actions in $O(MN_{part})$. Notice further that the sensor management algorithm is permutation invariant as it only depends on the likelihood of the measurements given the particles.

We have specialized here to the case where the measurements are thresholded (binary), but make the following comments about the extension to more complicated scenarios. This is relevant for sensors that can make continuous valued measurements such as that described in Section 2.3.3. It is straightforward to extend the binary case to a situation where the measurement $\mathbf{z}$ may take on one of a finite number of values. This is relevant in a situation where, for example, raw sensor returns are passed through an automatic target recognition algorithm and translated into target identifications that come from a discrete set of possibilities. In the case where $\mathbf{z}$ is continuous valued, the integral of (3.13) must be solved approximately. In the simulation section we address one such approach applicable to scalar (but continuous valued) $\mathbf{z}$, where the range of $\mathbf{z}$ is quantized into a small number of regions and the integral is evaluated as a discrete sum. As we will see there, very little performance loss is incurred with this approximation.

Table 3.1: The information based sensor management algorithm.

1. Generate particles representing $p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})$ by proposing particles representing $p(\mathbf{X}^{k-1}, T^{k-1}|\mathbf{Z}^{k-1})$ forward via the kinematic prior.
2. Compute the expected gain in information for each possible sensing action m (e.g. evaluate 3.13 using the particles generated in step (1) for all m).
3. Use particles representing $p(\mathbf{X}^{k-1}, T^{k-1}|\mathbf{Z}^{k-1})$ and the most recently received measurement $\mathbf{z}^k$ to propose a new set of particles representing $p(\mathbf{X}^k, T^k|\mathbf{Z}^k)$ (see Chapter II).
4. set k $\leftarrow$ k+1, and go to step (1)

In summary, the information based sensor management algorithm proceeds as follows. At each occasion where a sensing action is to be made, we predict the posterior JMPD from time $k-1$ forward to time k to form the prior JMPD at time k. We then evaluate the expected gain in information between the prior JMPD at time k and the posterior JMPD at time k for all possible sensing actions m by evaluating (3.14) for each m. We then task the sensor to perform the sensing action that gives the maximum expected gain in information (reduction in entropy). The sensing action is then performed, resulting in a particular measurement $\mathbf{z}$. This measurement is then used to form the posterior JMPD via Bayes' rule (2.5). The complete particle filtering and sensor management algorithm is summarized in Table 3.1.

3.2.4 On the Value of α in the Rényi Divergence

The Rényi divergence has been used in the past in many diverse applications, including content-based image retrieval, image georegistration, and target detection [89][92]. These studies provide guidance as to the optimal choice of α.

In the georegistration problem [89] it was empirically determined that the value of α leading to highest resolution clusters around either $\alpha = 1$ or $\alpha = 0.5$ correspond-

ing to the KL divergence and the Hellinger affinity respectively. The determining factor appears to be the degree of differentiation between the two densities under consideration. If the densities are very similar, i.e. difficult to discriminate, then the indexing performance of the Hellinger affinity distance ($\alpha = 0.5$) was observed to be better that the KL divergence ($\alpha = 1$). These empirical results give reason to believe that either $\alpha = 0.5$ or $\alpha = 1$ are good choices. We investigate the performance of our scheme under both choices in Sections 3.4.1 and 3.4.2.

An asymptotic analysis [89] shows that $\alpha = .5$ results in the maximum discriminatory ability between two densities that are very similar. The value $\alpha = .5$ provides a weighting which stresses the tails, or the minor differences, between two distributions. In the case where the two densities of interest are very similar (as in our application where one is a prediction density and one is a measurement updated density), the salient differences are in the regions of low probability, and therefore we anticipate that this choice of α will yield the best results.

3.3 Generalizations to Rényi Divergence

Use of the Rényi Divergence as a sensor scheduling metric has the effect that actions are taken so as to maximize the expected gain in information. This method has the advantage that no adhoc assumptions as to the relative utility of various types of information (e.g. information about target position versus information about target identification) need to be made. Actions that are expected to gain the most information (whether it be about position or identification) are chosen.

In some circumstances, there may be prior information about the relative utility of different types of information, e.g., tracking, identification, or detection. In this section, we show how this prior information can be naturally incorporated into the

expected divergence calculations discussed so far. We give two generalizations:

1. A *weighted divergence*, applicable when certain multitarget states $\mathbf{X}$ are more important to learn information about (e.g. when it is more important to gather information about certain target types than others).
2. A *divergence between marginalized JMPDs*, applicable when certain types of information are more important than others (e.g. position information is more important than identification information).

3.3.1 Weighted Rényi Divergence

If certain multitarget states $\mathbf{X}$ are more important to learn information about, we can generalize the Rényi Divergence (3.7) to

$$D_\alpha^*\left(p(\cdot|\mathbf{Z}^k)||p(\cdot|\mathbf{Z}^{k-1})\right) = \frac{1}{\alpha-1} ln \int_{\mathbf{X}} p(\mathbf{X}^k, T^k|\mathbf{Z}^k)^\alpha p(\mathbf{X}^k, T^k|\mathbf{Z}^{k-1})^{1-\alpha} f(\mathbf{X}^k, T^k) d\mathbf{X}^k \quad . \tag{3.15}$$

where $f(\mathbf{X}^k, T^k)$ is a (possibly time varying) weight function that emphasizes information about certain states $\mathbf{X}$ more than others (note that (3.7) corresponds to the choice $f(\mathbf{X}^k, T^k) = 1, \forall \mathbf{X}, T, k$). A sufficient condition that ensures the integral evaluates to a positive number and therefore D_α^* remains a divergence is that $f(\mathbf{X}, T)$ correspond to a probability density function.

Following the same calculations as in Section 3.2, the generalized divergence results in a simple modification of the expected divergence calculation (3.14)

$$\langle D_\alpha^* \rangle_m = \frac{1}{\alpha-1} \times \sum_{z^k=0}^{1} p(z^k|\mathbf{Z}^{k-1}, m) ln \frac{1}{p(z^k|\mathbf{Z}^{k-1}, m)^\alpha} \sum_{p=1}^{N_{part}} w_p p(z|\mathbf{X}_p, m)^\alpha f(\mathbf{X}_p, m) \tag{3.16}$$

where $f(\mathbf{X}_p, m)$ is the function on the multitarget particle emphasizing information

about some multitarget states more than others. The divergence in (3.11) corresponds to $f(\mathbf{X}_p, m) = 1, \forall \mathbf{X}, T, k, m$.

An example application of this strategy is the case where learning information about certain types of targets is more important than for other types. In this case, the function $f(\mathbf{X}_p, m)$ would be large when $\mathbf{X}$ includes the high-value target type and m corresponds to the appropriate measurement, which means that information about this target is more valuable. This would tend to give more resources to targets of this type at the expense of other less valuable types.

3.3.2 Rényi Divergence Between Marginalized JMPDs

If is known *a priori* that certain types of information are more important than others, one can use a Rényi divergence between marginalized JMPDs to allocate the sensor.

Consider the case where one only wants to consider kinematic information when scheduling the sensor. The state of the system $\mathbf{X}$ contains both kinematic $(\mathbf{x}, \dot{x}, y, \dot{y})$ and target type (t) information

$$\mathbf{X} = [x_1, \dot{x}_1, y_1, \dot{y}_1, t_1 \cdots x_T, \dot{x}_T, y_T, \dot{y}_T, t_T] \ . \tag{3.17}$$

In this case, one can marginalize across target type and generate a new state variable $\mathbf{Y}$ that only contains kinematic information

$$\mathbf{Y} = [x_1, \dot{x}_1, y_1, \dot{y}_1, \cdots x_T, \dot{x}_T, y_T, \dot{y}_T] \ . \tag{3.18}$$

The marginalization is performed as

$$p(x_1, \dot{x}_1, y_1, \dot{y}_1, \cdots x_T, \dot{x}_T, y_T, \dot{y}_T | \mathbf{Z}) = \tag{3.19}$$

$$\int_{t_1} \cdots \int_{t_T} dt_1 \cdots dt_T p(x_1, \dot{x}_1, y_1, \dot{y}_1, t_1, \cdots x_T, \dot{x}_T, y_T, \dot{y}_T, t_T | \mathbf{Z}) \ . \tag{3.20}$$

Then the density $p(\mathbf{Y}|\mathbf{Z})$ can be used to calculate expected information gain on the reduced state space as

$$D_\alpha\left(p(\cdot|\mathbf{Z}^k)||p(\cdot|\mathbf{Z}^{k-1})\right) = \frac{1}{\alpha-1} ln \int_{\mathbf{Y}} p(\mathbf{Y}^k, T^k|\mathbf{Z}^k)^\alpha p(\mathbf{Y}^k, T^k|\mathbf{Z}^{k-1})^{1-\alpha} d\mathbf{Y}^k \quad . \tag{3.21}$$

In this particular case, marginalization results in an especially simple implementation, where a new set of pseudo particles $\mathbf{Y}_p$ are generated from the existing particles $\mathbf{X}_p$ by simply ignoring the target type components.

One may envision treating kinematic and target type information separately (although this marginalization ignores the coupling in uncertainty between the two) by marginalizing out target type to get $\mathbf{Y}$ and marginalizing out kinematics to get $\mathbf{V}$ and using

$$D_\alpha\left(p(\cdot|\mathbf{Z}^k)||p(\cdot|\mathbf{Z}^{k-1})\right) = \frac{1}{\alpha-1} ln \int_{\mathbf{Y}} p(\mathbf{Y}^k, T^k|\mathbf{Z}^k)^\alpha p(\mathbf{Y}^k, T^k|\mathbf{Z}^{k-1})^{1-\alpha} + \frac{1}{\alpha-1} ln \int_{\mathbf{V}} p(\mathbf{V}^k, T^k|\mathbf{Z}^k)^\alpha p(\mathbf{V}^k, T^k|\mathbf{Z}^{k-1})^{1-\alpha} \quad . \tag{3.22}$$

Of course, the individual terms in this equation can be given a relative weight and this method can be combined with the method of Section 3.3.1 to generalize even further.

3.4 Simulation Results

In this section, we provide a set of simulation results to show the feasibility and benefit of sensor management in the multitarget tracking scenario.

In Sections 3.4.1 and 3.4.2 we present two detailed simulations where we are interested in tracking the kinematic state of a collection of moving targets. The performance of the information based method is compared to a simple periodic scheme and

to two other sensor management algorithms via Monte Carlo experiments. The measures of performance here are the (average) number of targets successfully tracked, and the (median) RMS tracking error. In both simulations, we assume the sensor is limited by time, bandwidth and other physical constraints which only allow it to measure a subset of the surveillance area at any epoch, and the goal of the algorithm is to effectively choose which portions of the surveillance region to measure at each time step.

In Section 3.4.3 we give results involving the performance of the sensor manager with a multimode sensor. In this simulation, the sensor may again choose where to point the sensor, but additionally much choose the mode. We simulate three modes available on real platforms, a fixed target indication (FTI) mode, moving target indication (MTI) mode, and a identification (ID)[1] mode. The goal here is to track and identify a collection of targets that may be moving or stopped using the appropriate sensor mode and pointing direction. We again compare the performance of the information-based method with a simple periodic scheme and two other sensor management algorithms via Monte Carlo testing. The measures of performance in this simulation are again the (average) number of targets successfully tracked, and the (median) RMS tracking error as well as the (average) number of targets correctly identified.

Third, in Section 3.4.4 we look at simulations involving the weighted Rényi divergence and the Rényi divergence between marginalized JMPDs. In the first case, we focus on a situation where a particular target type is of high value and hence learning information about it is of high importance. In the second case, we focus on the tracking performance when only information about position is used to drive

[1]The FTI sensor simulation serves as an approximation to a synthetic aperture radar (SAR), the MTI serves as an approximation to a ground moving target indicator radar (GMTI), and the ID sensor simulation serves an approximation to an automatic target recognition processing chain, containing perhaps a high resolution radar (HRR)

the sensor. This is contrasted to an alternative sensor management algorithm that is derived to explicitly minimize the tracking error.

Fourth, in Section 3.4.5, we investigate the performance of the tracking and sensor management algorithm under model mismatch. There are two models of importance: the kinematic model, which specifies probabilistically how targets move; and the sensor model, which specifies probabilistically how the sensor measurements couple to target states.

We finish this chapter with a note on computational complexity of the algorithm and provide timing numbers versus number of targets in Section 3.4.6.

3.4.1 Tracking Three Simulated Targets Using the Information Based Approach

We gauge the performance of the sensor management scheme by considering the following model problem. There are three targets moving on a 12×12 sensor grid. The movement of each target is modeled using the four-dimensional state vector $[x, \dot{x}, y, \dot{y}]'$. Target motion is simulated using a constant-velocity (CV) model with large plant noise (i.e. according to the model of Section 2.1.1). Motions for each target are independent. The trajectories have been shifted and time delayed so there are two times during the simulation where targets cross paths (i.e. come within sensor resolution).

The target kinematics assumed by the filter (2.4) are nearly CV as in the simulation. At each time step, a set of L (not necessarily distinct) cells are measured. The sensor is at a fixed location above the targets and all cells are always visible to the sensor. When measuring a cell, the imager returns either a 0 (no detection) or a 1 (detection) which is governed by a probability of detection (P_d) and a per-cell false alarm rate (P_f). The signal to noise ratio (SNR) links these values together. In this illustration, we take $P_d = 0.5$, and $P_f = P_d^{(1+SNR)}$, which is a stan-

dard model for thresholded detection of Rayleigh returns (see Section 2.1.2). When there are T targets in the same cell, the detection probability increases according to $P_d(T)=P_d^{\frac{1+SNR}{1+T*SNR}}$. This model is known by the filter and used to evaluate (2.5). The filter is initialized with 10% of the particles in the correct state (both number of targets and kinematic state). The rest of the particles are uniformly distributed in both the number of targets and kinematic state.

We contrast the performance of the tracker when the sensor uses a non-managed (periodic) scheme with the performance when the sensor uses the information based sensing management scheme presented in Section 3.2. The periodic scheme measures each cell in sequence. At time 1, cells $1...L$ are measured. At time 2, cells $L+1...2L$ are measured. This sequence continues until all cells have been measured, at which time the scheme resets. The managed scheme uses the expected information divergence to calculate the best L cells to measure at each time. This often results in the same cell being measured several times at one time step. Multiple measurements made in the same cell are independent (i.e. each measurement in a target containing cell returns a detection with probability P_d irrespective of whether earlier measurements resulted in a detection).

Figure 3.1 presents a single-time snapshot, which graphically illustrates the difference in behavior between the two schemes.

Qualitatively, in the managed scenario measurements are focused in or near cells that the targets are in. Quantitatively, the covariance ellipses calculated by the filter show that performance is significantly better in the managed scenario. The ellipses are centered at the filter estimate of target location and describe the uncertainty about the target location. Equations (2.41) and (2.42) are used to compute the target estimate and covariance, respectively. The plot shows all points (x, y) where

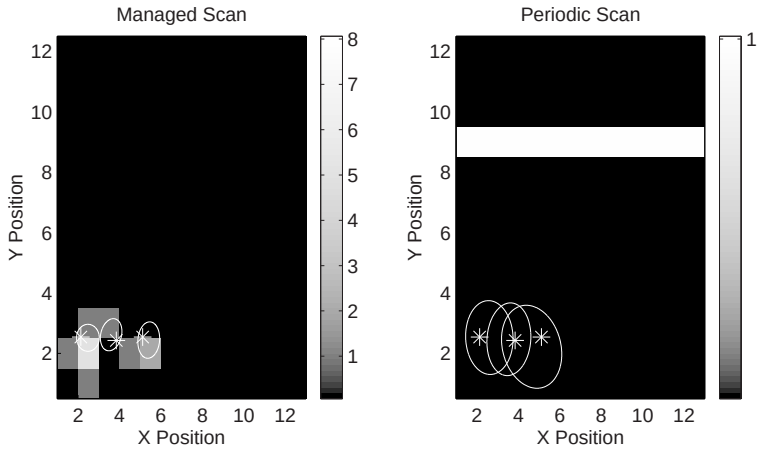


Figure 3.1: An illustration contrasting managed and non-managed tracking performance. (L) Using sensor management, and (R) A periodic scheme. Targets are marked with an asterisk, the (x,y) covariance spread of the filter estimate is given by the ellipse, and grey scale is used to indicate the number of times each cell has been measured at this time step (the total number of looks is identical in each scenario). In the periodic scenario, one twelfth of the region is scanned at each time step starting at the bottom and proceeding to the top before repeating (cells scanned at this epoch are indicated by the white stripe). With sensor management, measurements are used only in areas that contain targets.

$[x\ y]\Lambda^{-1}[x\ y]' < 1.$

A more detailed examination is provided in the Monte Carlo simulation results of Figure 3.2. We refer to each cell that is measured as a "look", and are interested in empirically determining how many looks the non-managed algorithm requires to achieve the same performance as the managed algorithm at a fixed number of looks. The sensor management algorithm was run with 24 looks (i.e. was able to scan 24 cells at each time step) and is compared to the non-managed scheme with 24 to 312 looks. Here we take $\alpha = 0.99999$ (approximately the KL divergence) in (3.5). It is found that the non-managed scenario needs approximately 312 looks to equal the performance of the managed algorithm in terms of RMS error. Multitarget RMS position error is computed by computing the average RMS error across all targets. The sensor manager is approximately 13 times as efficient as compared to allocating the sensors without management. This efficiency implies that in an operational scenario target tracking could be done with an order of magnitude fewer sensor

dwells. Alternatively put, more targets could be tracked with the same number of total resources when this sensor management strategy is employed.

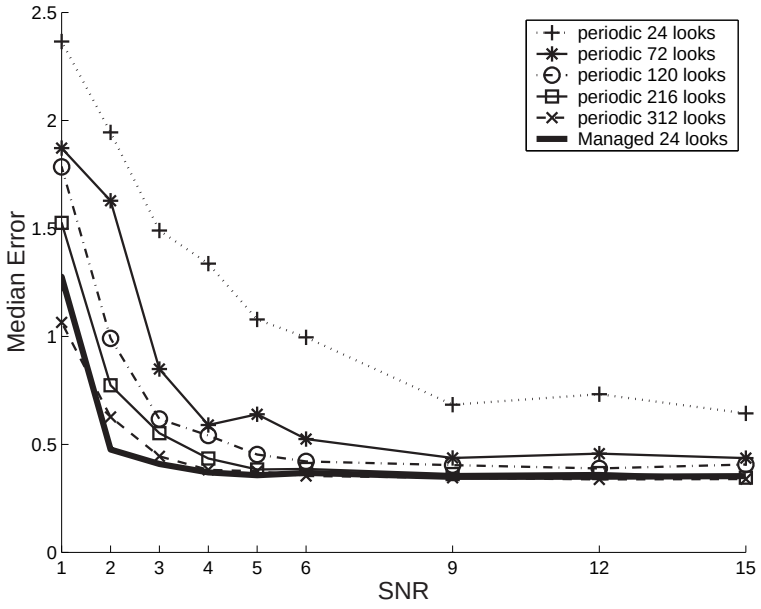


Figure 3.2: A Monte Carlo comparison of divergence based sensor management to the periodic scheme for a model problem. Managed performance with 24 looks is similar to non-managed with 312 looks.

To determine the sensitivity of the sensor management algorithm to the choice of α, we test the performance with $\alpha = .1$, $\alpha = .5$, and $\alpha \approx 1$. Figure 3.3 shows that in this case, where the actual target motion is very well modeled by the filter dynamics, the performance of the sensor management algorithm is insensitive to the choice of α. We generally find this to be the case when the filter model is closely matched to the actual target kinematics.

3.4.2 Simultaneous Detection and Tracking of Ten Real Targets Using the Information Based Approach

We test the sensor management algorithm here using a modified version of the above simulation, which demonstrates the technique in a scenario of increased realism. Here we have ten real targets moving in a $5000m \times 5000m$ surveillance area. Each target is modeled using the four-dimensional state vector $[x, \dot{x}, y, \dot{y}]'$. Target

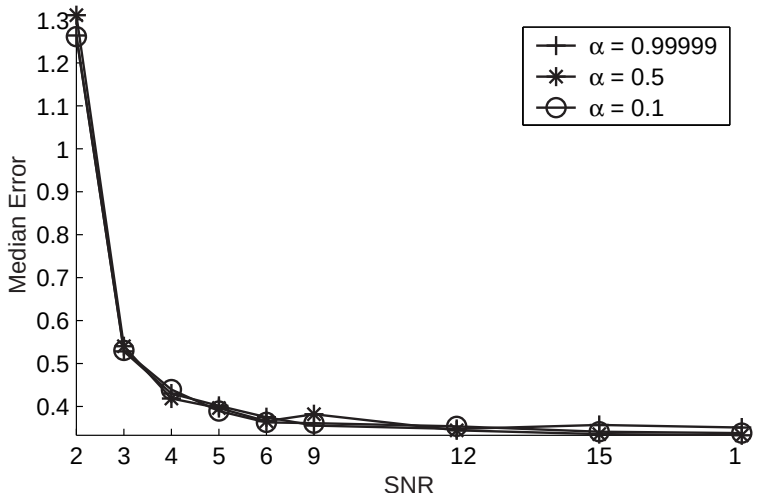


Figure 3.3: The performance of the sensor management algorithm with different values of the Rényi Divergence parameter α. We find that in the case where the filter dynamics match the actual target dynamics, the algorithm is insensitive to the choice of α.

trajectories for the simulation come directly from the set of recorded data based on GPS measurements of vehicle positions over time collected as part of a battle training exercise at the Army's National Training Center (see Figure 2.1). Target positions are recorded at 1 second intervals, and routinely come within sensor cell resolution (i.e. cross). Therefore, there is often measurement to track ambiguity, which is handled automatically by JMPD since there is no measurement-to-track assignment necessary.

The filter again assumes nearly constant velocity motion with large plant noise as the model of target kinematics. However, in this case the model is severely at odds with the actual target behavior which contains sudden accelerations and move-stop-move behavior. This model mismatch adds another level of difficulty to this scenario that was not present previously.

At each time step, an imager is able to measure cells in the surveillance area by making measurements on a grid with 100m$\times$100m detection cell resolution. The sensor simulates a moving target indicator (MTI) system in that it may lay a beam

down on the ground that is one resolution cell wide and many resolution cells deep. Each time a beam is formed, a vector of measurements (a vector zeros and ones corresponding to non-detections and detections) is returned, one measurement for each of the ten resolution cells. In this simulation, we refer to each beam that is laid down as a single "look".

As in the previous simulation, the sensor is at a fixed location above the targets and all cells are always visible to the sensor. When making a measurement, the imager returns either a 0 (no detection) or a 1 (detection) governed by P_d, P_f, and SNR. When there are T targets in the same cell, the detection probability increases according to $P_d(T)=P_d^{\frac{1+SNR}{1+T*SNR}}$.

Simulations Regarding Target Detection

Figures 3.4 and 3.5 provide simulations of performance for detecting and tracking the ten real targets. Figure 3.4 is analogous to Figure 2.6 from Chapter II, in that it investigates the performance of the algorithm versus number of particles. Similarly, Figure 3.5 is analogous to Figure 2.7 from Chapter II, in that it investigates the performance of the algorithm versus signal to noise ratio.

The measures of performance are $\hat{T}$ and $T_{tracked}$ as given in Section 2.3.4, and the model parameters are again the initial existence probability for each of the sensor cells, the birth rate, and the death rate (see Section 2.2.2). This simulation uses the same amount of sensor resources as in the examples from Section 2.3.4 (enough to cover the entire region exactly once at each time step).

A Comparison to Other Management Strategies

We provide in this section a comparison of the divergence based sensor management scheme, a periodic scheme, and two other methods of sensor management. This

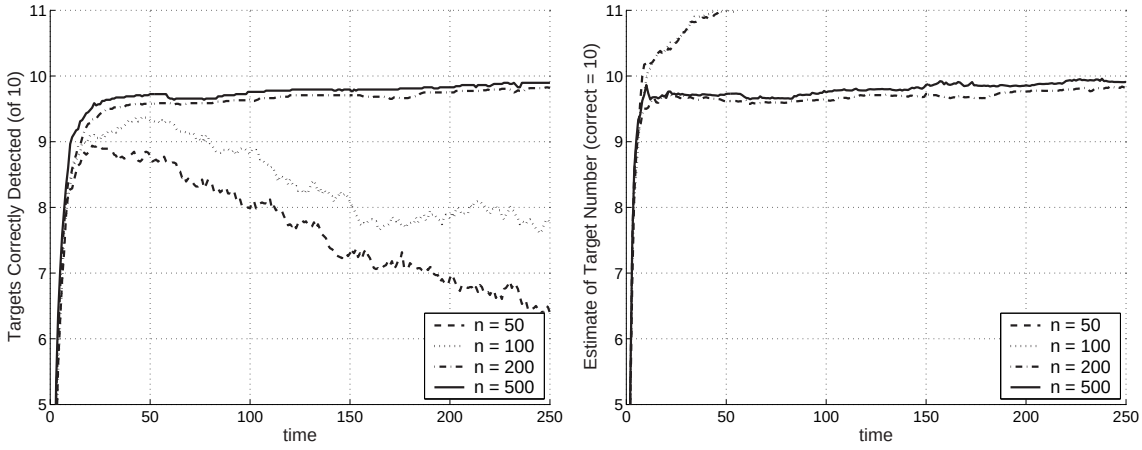


Figure 3.4: Sensor management for detecting and tracking ten real targets versus the number of particles. As in the earlier non-managed experiment, the SNR is $10dB$, the removal rate is .005 and the arrival rate is .02. Ideal performance is $\hat{T} = 10$ and $T_{tracked} = 10$ at all times.

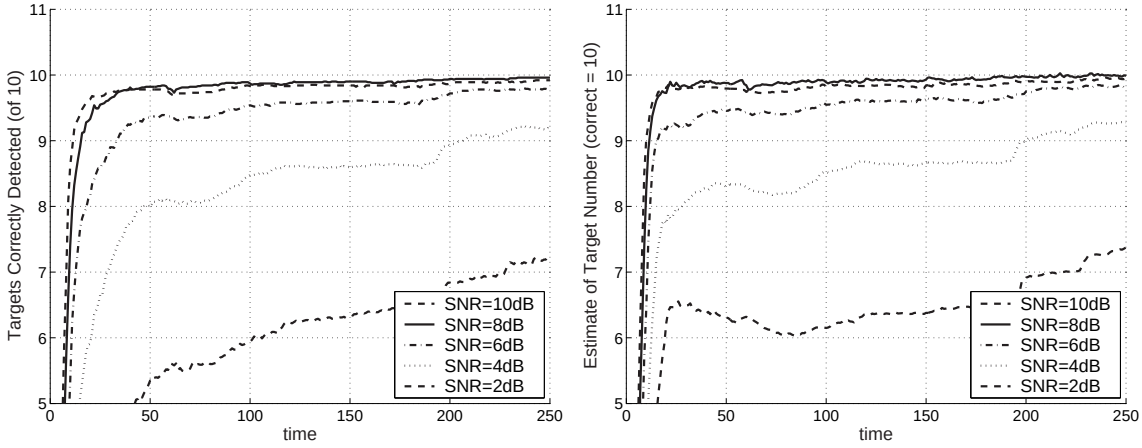


Figure 3.5: Sensor management for detecting and tracking ten real targets versus signal to noise ratio. As in the earlier non-managed experiment, the number of particles is 200, the removal rate is .005 and the arrival rate is .02. Ideal performance $\hat{T} = 10$ and $T_{tracked} = 10$ at all times.

comparison is carried out in a tracking-only scenario. The two other methods are local search schemes that use the heuristic that the target is likely to remain near its estimated position at the next time increment.

Sensor management algorithm "A" manages the sensor by pointing it at or near the estimated location of the targets. Specifically, algorithm "A" performs a gating

procedure to restrict the portion of the surveillance area that the sensor will consider measuring. The particle filter approximation of the time updated JMPD (2.4) is used to predict the location of each of the targets at the current time. The set of cells that the sensor manager considers is then restricted to those cells containing targets plus the surrounding cells, for a total of 9 cells in consideration per target. The dwells are then allocated randomly among the gated cells.

Sensor management algorithm "B" tasks the sensor based on the estimated number of targets in each sensor cell. Specifically, the particle approximation of the time updated JMPD is projected into sensor space to determine the filter's estimate of the number of targets in each sensor cell. The cell to measure is then selected probabilistically, favoring cells that are estimated to contain more targets. In the single target case, this method reduces to measuring the cell that is most likely to contain the target.

Notice that methods "A" and "B" introduce new assumptions not present in the information-based algorithm. In particular, both give value only to looking near where targets are predicted to be. There seems to be no natural extension of this heuristic to include detection of new targets. Furthermore, algorithm "A" assumes equal weight to all areas predicted to contain targets and algorithm "B" gives precedence to areas predicted to contain multiple targets.

We compare the performance of the various managed strategies and the periodic scheme in Figure 3.6 by looking at RMS error versus number of sensor dwells ("looks"). All tests use $P_d = 0.5$, $SNR = 2$, and $P_f = P_d^{(1+SNR)}$ and are run with the number of targets and target states initialized with ground truth. As before, multitarget RMS error is computed by taking the average RMS error across all ten targets.

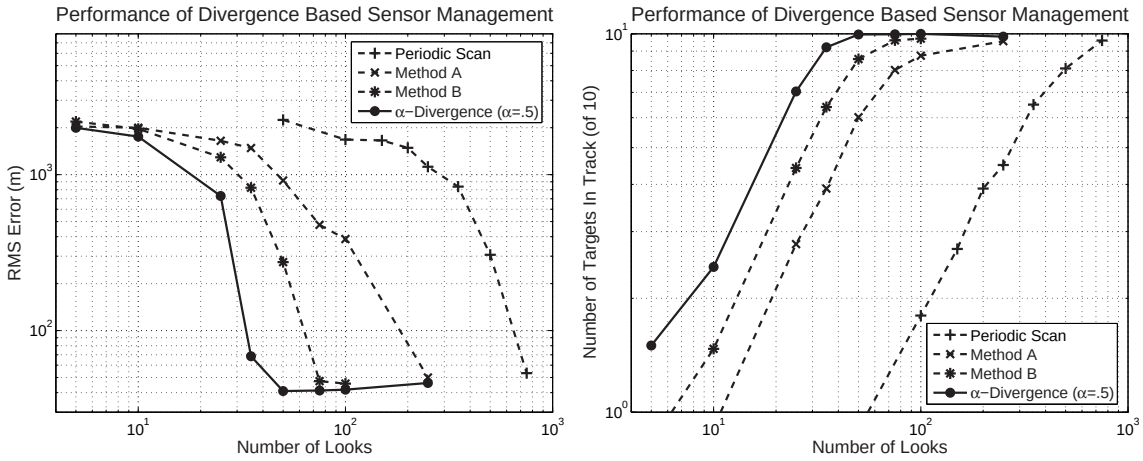


Figure 3.6: A comparison of the information-based method to periodic scan and two other methods. The other schemes are also based on recursive estimation of the JMPD, but use the likelihood of target presence as the scheduling criteria, rather than expected information gain. The performance is measured in terms of the (median) RMS error versus number of looks and the (average) number of targets in track. The α-divergence strategy out performs the other strategies, and at 35 looks performs similarly to non-managed with 750 looks.

Figure 3.6 shows that the non-managed scenario at 750 looks performs approximately the same as the managed algorithm at 35 looks in terms of RMSE error. Thus, we conclude that the sensor manager is approximately 20 times as efficient as allocating the sensors without management. Furthermore, the divergence driven sensor management scheme outperforms the local search heuristics A and B.

On the Value of α

We provide in this section a comparison of the performance of the sensor management algorithm under different values of α in (3.1). This problem is more challenging then the simulation of Section 3.4.1 for several reasons (e.g. number of targets, number of target crossing events, and model mismatch). Of particular interest is the fact that the filter motion model and actual target kinematics do not match very well. The asymptotic analysis performed previously (see Section 3.2.4) leads us to believe that $\alpha = 0.5$ is the right choice in this scenario.

In Figure 3.7, we show the results of 50 Monte Carlo trials using our sensor management technique with $\alpha = 0.1$, $\alpha = 0.5$, and $\alpha = 0.99999$. All tests use $P_d = 0.5$, $SNR = 2$, and $P_f = P_d^{(1+SNR)}$ and are run with the number of targets and states initialized with ground truth. The statistics are summarized in Table 3.2. We find that indeed the sensor management algorithm with $\alpha = 0.5$ performs best here as it does not lose track on any of the 10 targets during any of the 50 simulation runs. We define the track to be lost when the filter error remains above 100 meters (1 cell) after some point in time. Both the $\alpha \approx 1$ and $\alpha = 0.1$ case lose track of targets on several occasions.

Table 3.2: Tracking performance for different values of the Rényi Divergence parameter α. The values are the result of fifty independent simulations of the sensor management algorithm with three simulated targets.

α	Mean Position Error(m)	Position Error Variance (m)
0.1	49.57	614.01
0.5	47.28	140.25
0.99999	57.44	1955.54

3.4.3 Scheduling a Multimode Sensor using the Information Based Approach

As mentioned earlier, one of the principal benefits of the information based sensor management approach is that the complex tradeoffs between different sensing actions are automatically taken into account. We have already seen this in the case of a radar with an agile array that must trade the decision to measure locations predicted to contain single targets and a particular uncertainty level versus measuring locations containing multiple targets with a corresponding uncertainty. The tradeoffs become even more complex in the case of a sensor that is able to decide between several modes of operation. In this section, we investigate via simulation studies a situation involving a sensor that has three modes available for use

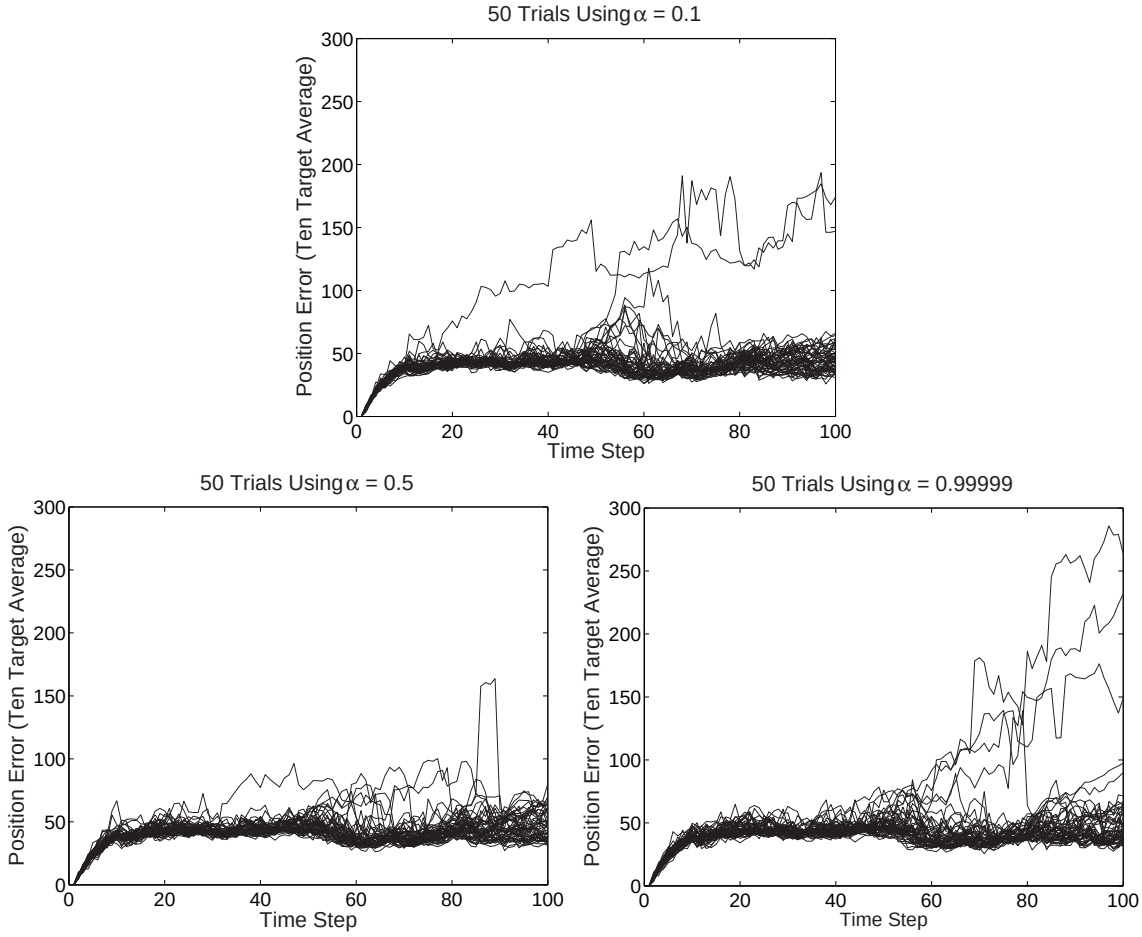


Figure 3.7: A comparison of sensor management performance under different values of the Rényi Divergence parameter, α. On simulations involving ten real targets, we find that the choice $alpha = 0.5$ leads to the best tracking performance.

- A moving target indicator (MTI), which is a mode that able to detect the position of targets only when they are moving,
- A fixed target indicator (FTI), which is a mode that is able to detect the position of targets only when they are stopped, and
- An identification (ID) sensor, which is able to determine the type (e.g. jeep or tank) of a target

In order to task the sensor, we need to compute the myopic gain in information for each of the possible sensing actions. This includes each possible measurement

when using the MTI sensor, each possible measurement when using the FTI sensor, and each possible measurement when using the ID sensor.

We use an MTI sensor as described in Sections 3.4.1 and 3.4.2 (i.e. using thresholded measurements). We add the additional level of realism here that targets moving slower than a minimum detectable velocity (MDV) of 1m/s are not detectable by the MTI sensor and act like empty cells. This enters into the sensor model, in that $\chi_i(\mathbf{x}_t)$ now only counts targets that are moving faster the MDV as being coupled to a sensor cell (see 2.1.2). The FTI sensor is modeled similarly, except that only targets that are stopped (i.e. $\dot{x} = \dot{y} = 0$) are visible to the sensor. The expected myopic gain in information is computed using (3.14) for all pointing directions for each of these two modes.

The ID sensor models a complete automatic target recognition (ATR) system that involves high range resolution radar and a signal processing algorithm. There are 3 possible target types for this simulation. We model the performance by a confusion matrix, which describes the probability that algorithm will return a particular classification when it is pointed at a particular target type. The model is given in Table 3.3 and says that when a single target occupies a detection cell, the probability of correctly identifying the target is 0.6, with the misclassifications spread evenly about the other two classes. Also, when multiple targets occupy the same cell or no targets are in the cell, the ATR algorithm returns a random classification. This model is reasonable for common ATR systems, which rely on the geometry of scattering centers for targets to provide classification calls. When multiple targets are contributing to an energy return, this geometry is corrupted and ATR performs very poorly.

	Actual Cell Status			
Classification Probability	Type 1	Type 2	Type 3	Empty or Multiply Occupied
Type 1	0.60	0.20	0.20	0.33
Type 2	0.20	0.60	0.20	0.33
Type 3	0.20	0.20	0.60	0.33

Table 3.3: The model for the identification sensor. Each measurement of a single target is independent and provides the correct identification 60% of the time. Measurements of empty cells or cells containing multiple targets return a random classification call.

The expected myopic gain in information from using the ID sensor follows directly from (3.13), where the number of possible outcomes is now 3:

$$< D_\alpha >_m = \frac{1}{\alpha - 1} \sum_{z=1}^{3} p(z) ln \frac{1}{p(z)^\alpha} \sum_{p=1}^{N_{part}} w_p p(z|\mathbf{X}_p)^\alpha \tag{3.23}$$

The goal is to use the sensor to simultaneously determine the position and target types of a group of maneuvering targets. Again, the target motion is taken from real targets that are performing combat maneuvers. At each time step, the sensor must choose from among M different sensing actions (choosing both mode and pointing angle). Initially, the positions of the targets are known (with some covariance) and the identification is unknown.

We present in Figure 3.8 a comparison between the performance of the algorithm using the information based method, periodic scan, and the two methods described in Section 3.4.2. The periodic scan and alternative sensor management strategies are defined as they were in Section 3.4.2 with the addition that now the alternate methods rotate through the 3 possible sensing modalities.

3.4.4 Application of the Modified Divergence Metric

In this section, we give two simulation results to illustrate application of the modified divergence metrics introduced in Sections 3.3.1 and 3.3.2.

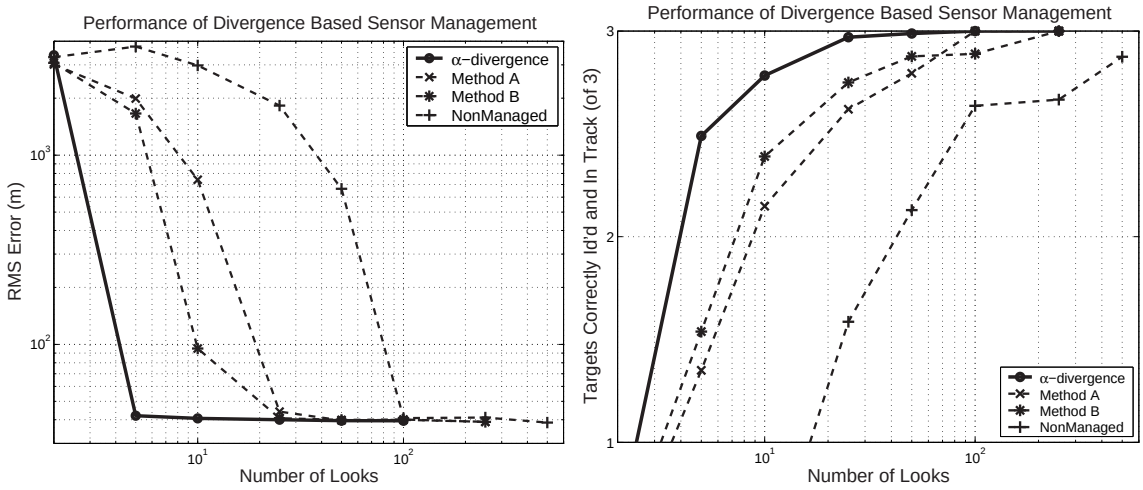


Figure 3.8: A comparison of the information-based method to periodic scan and two other methods for simultaneous track and ID. The performance is measured in terms of the (median) RMS error and the (average) number of targets correctly identified and in track. Again, the α-divergence strategy out performs the other strategies.

The Weighted Rényi Divergence

In this subsection we consider two simulations of a two-target tracking problem. Each target has a unique identification which is known to the filter at initialization. We contrast the performance under two different choices of divergence weighting in Figure 3.4.4. The first simulation uses $f(\mathbf{X}^k, T^k) = 1$ (i.e. the usual Rényi Divergence based scheduling metric). In the second simulation, the divergence is biased towards preferring information about target 2 by choosing $f(\mathbf{X}^k, T^k) = .55$ if $\mathbf{X}$ contains a target of type 2 and $f(\mathbf{X}^k, T^k) = .45$ if $\mathbf{X}$ contains a target of type 1. In this manner, learning information about target type 2 is of greater importance than learning information about target type 1.

We see that in the case of uniform information weighting, target 1 is tracked more successfully than target 2 (86% versus 82%). When the information gain criteria is weighted toward favoring target 2, target 2 attracts a larger proportion of measurements. Target 2 is now tracked 98% of the time at the expense of tracking target 1,

which is now tracked only 75% of the time.

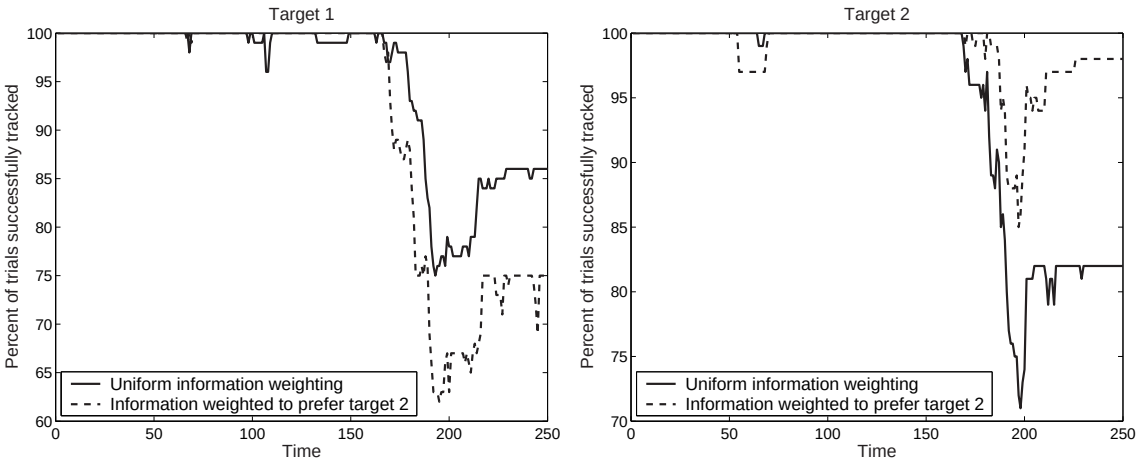


Figure 3.9: An illustration of performance when using a weighted divergence which prefers information about a certain type of target. There are two figures, each corresponding to a particular target. The two targets are of different type. Each of the figures contains two curves. The first curve gives the performance when information is weighted uniformly. The second curve shows the performance when the information gain criteria has been weighted to prefer information about target 2. The plots show that when the metric is weighted to prefer target 2, tracking performance of this target is significantly improved (at the expense of the target 1).

Rényi Divergence Between Marginalized JMPDs

As discussed in Section 3.3.2, using a marginalized version of the JMPD in the divergence calculation will lead the algorithm to schedule using only information about particular parts of the state space. In this section, we consider one such marginalization where target type and velocity information is marginalized out of the JMPD, leaving only position information. Therefore, the sensor will be scheduled based only on gaining position information about the targets. We contrast the performance with the divergence on the full JMPD in terms of performance at locating and identifying a target.

In addition, we introduce a new sensor scheduling algorithm which is derived explicitly to minimize the tracking error [95]. The performance of this new algorithm will provide a baseline with which to compare the information based algorithms. Let c

denote a sensor cell. Furthermore, let $p(c|\mathbf{Z})$ denote the probability a target is in cell c given the measurements $\mathbf{Z}$. Then the sensor allocation strategy to minimize tracking error is to choose the sensing action m that maximizes the posterior probability the target is in cell c for the most likely cell c, i.e.

$$m_{opt} = \arg\max_m E_z\left(\max_c p(c|\mathbf{Z}, \mathbf{z}, m) \right)$$
$$\arg\max_m \int_{\mathbf{z}} p(\mathbf{z}|\mathbf{Z}, m)\left(\max_c p(c|\mathbf{Z}, \mathbf{z}, m) \right) d\mathbf{z} \tag{3.24}$$

The particle filter implementation of JMPD reduces this maximization to an easily computed quantity, analogous to that of the Rényi Divergence.

Figure 3.4.4 shows that the scheduler that is derived specifically to minimize tracking error is the best performing algorithm, when only considering tracking performance. The information gain between marginalized JMPDs, is slightly worse in terms of tracking performance. The information gain between the full state JMPD performs poorest when only considering tracking performance. This is attributable to the fact that the information gain criterion puts a premium on correctly identifying the target. Therefore, some dwells are used to identify the target rather than track the target. Using identification quality as the metric, the information gain criterion clearly outperforms the other two methods.

3.4.5 Performance under Model Mismatch

In this section, we present empirical results regarding the performance of the proposed algorithms under model mismatch. As described in Sections 2.1.1 and 2.1.2, the time and measurement evolution of the JMPD requires models of target kinematics and the sensor.

In practice, these models may not be accurately known to the filter. In particular, in the simulation experiments of 3.4.2, target motion was taken from real target

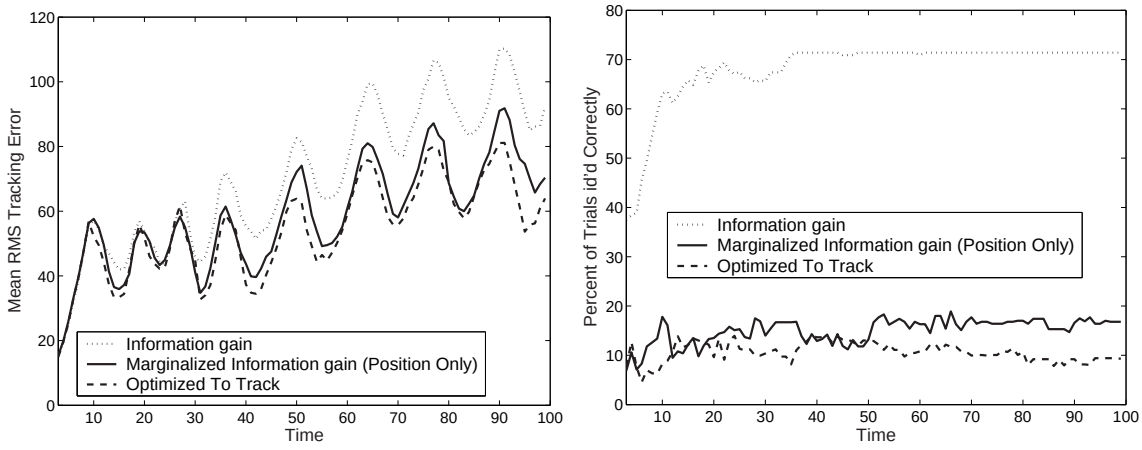


Figure 3.10: Comparison between the Rényi divergence scheduler, a Rényi divergence between marginalized JMPDs, and a scheduler designed to minimize tracking error. The divergence criteria is able to do good identification with only a small degradation in tracking performance. As expected, the scheduler designed to minimize track error performs best in terms of tracking error, followed by the divergence between marginalized JMPDs and the divergence. However, in terms of identification performance, the divergence between full state JMPDs is superior.

trajectories and so the kinematic model was imperfectly known (which is anticipated to result in degraded performance over a situation where the model was perfectly known). In Figure 3.4.5, we quantify just how a poor estimate of the kinematic model effects performance of the algorithms. The algorithm is remarkably robust to kinematic model mismatch, with a graceful degradation as the estimate of target kinematics becomes more flawed.

Furthermore, the sensor model may not be known exactly to the filter. Using the pixelated sensor described in Section 2.1.2, we consider the case where the SNR estimated by the filter is mismatched from the true SNR of the targets under surveillance. This results in a mismatch between the true detection and false alarm rate of the targets and the detection and false alarm rates estimated by the filter. Figure 3.4.5 shows simulations where the true SNR of the targets is varied from $SNR = 3$ to $SNR = 15$ and the filter estimate of the SNR is varied from $SNR = 2$ to $SNR = 24$. Again, the filter is remarkably robust to sensor model mismatch, with

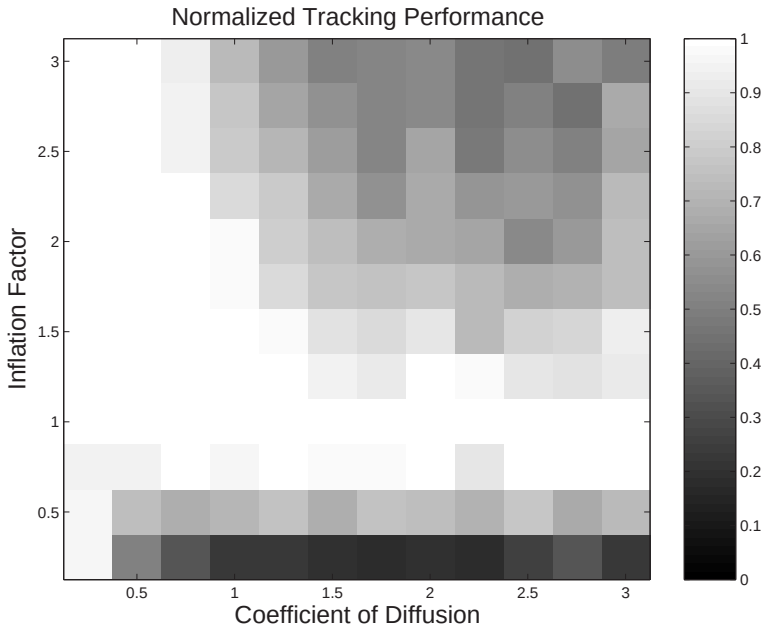


Figure 3.11: Tracking performance degradation when the kinematic model is mismatched. The vertical axis of the graph shows the true coefficient of diffusion of the target under surveillance. The horizontal axis shows the mismatch between the filter estimate of kinematics and the true kinematics (matched = 1). The color scheme shows the relative degradation in performance present under mismatch (< 1 implies poorer performance than the matched filer).

a graceful degradation as the estimate of SNR becomes more errant.

3.4.6 Computational Complexity of the Algorithm

We present in this section empirical results regarding the computational complexity of the method. In particular, this simulation involves a 15x15 km surveillance region with a number of ground moving targets. The sensors are able to measure 100mx100m cells on the ground, meaning that at each time step there are 22,500 cells where the expected Rényi Divergence must be computed in order to determine the best sensing action.

In Figure 3.13, we investigate the runtime performance of the algorithm versus number of targets under surveillance. For equitable comparison, as the number of targets increases, the number of sensor resources increases (i.e. the number of resources

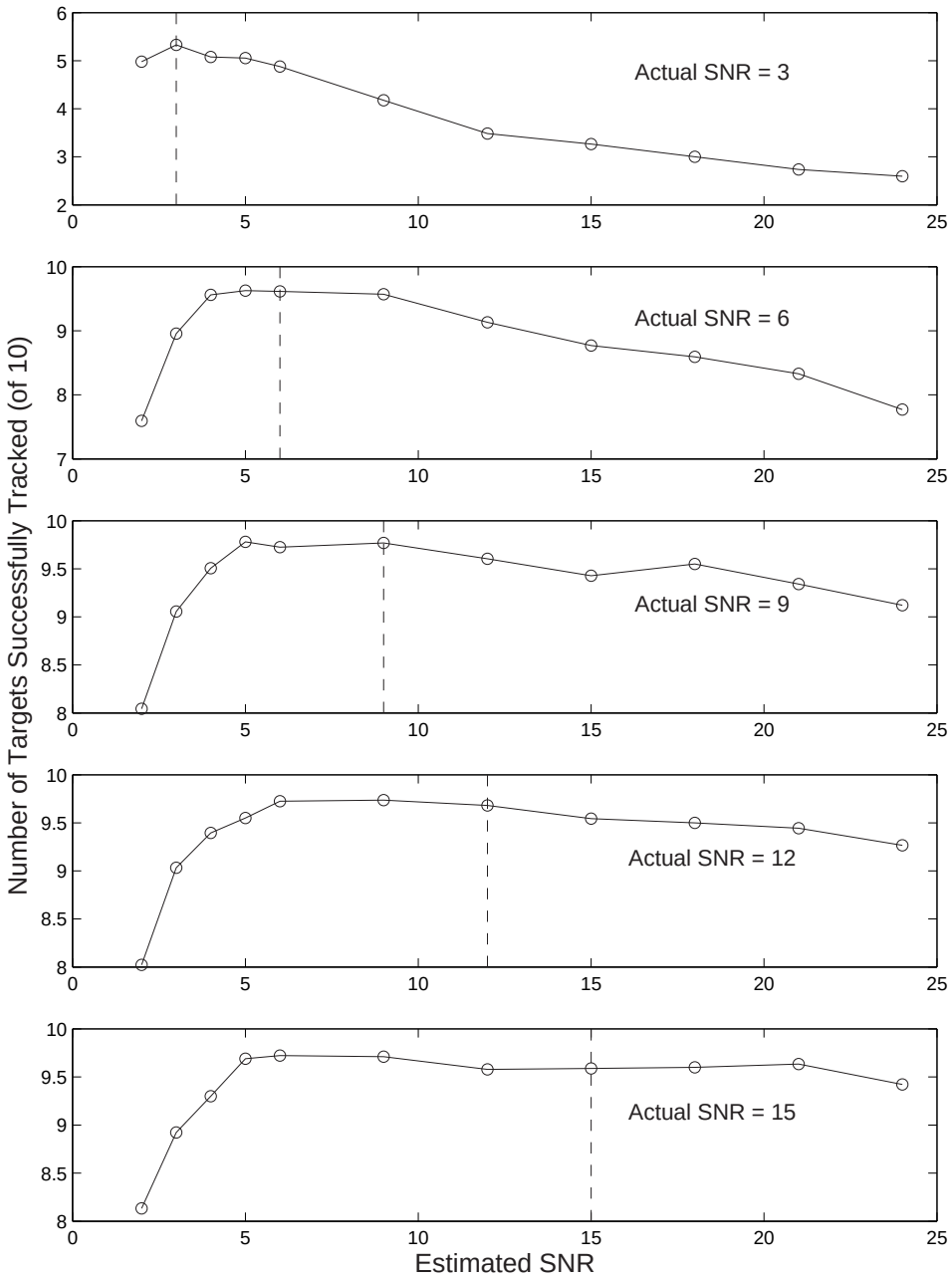


Figure 3.12: Tracking performance degradation when the sensor model is mismatched. Each graph corresponds to a true target SNR. The curves show tracking performance as the estimated SNR (i.e. that used by the filter) is varied. Performance degrades gradually particularly for large SNR targets.

per target is kept constant throughout the algorithm). With modest optimization, a hybrid MatLab/C implementation of the algorithm is able to track on the order of 40 targets in real time and perform tracking and sensor management on 10 targets

in real time.

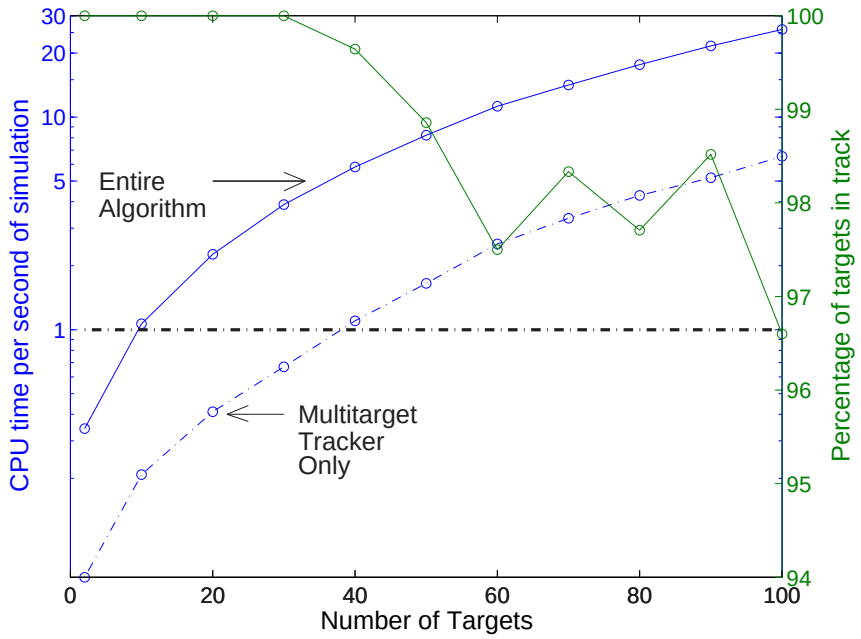


Figure 3.13: Execution time on an off the shelf 3GHz Linux box for the myopic sensor management algorithm in the case of thresholded measurements. The algorithm is able to track on the order of 40 targets in real time and perform tracking and sensor management on 10 targets in real time. All runs use 250 particles

We also investigate the effect of non-discrete measurements on performance of the algorithm. As alluded to earlier, one method of addressing the expectation over z when z is a continuous valued random variable is to quantize z into a set of N regions defined by $z_1 \cdots z_N$ and evaluate the sensor management integral using the trapezoid rule, e.g.,

$$\begin{aligned} < D_\alpha >_m &= \frac{1}{\alpha - 1} \int_{z=0}^{\infty} dz p(z|\mathbf{Z}^{k-1}, m) ln \frac{1}{p(z|\mathbf{Z}^{k-1}, m)^\alpha} \sum_{p=1}^{N_{part}} w_p p(z|\mathbf{X}_p, m)^\alpha \\ &\approx \frac{1}{\alpha - 1} \sum_{i=1}^{N} \left\{ p(z_i < i < z_{i+1}|\mathbf{Z}^{k-1}, m) \times \right. \\ &\qquad \left. ln \frac{1}{p(\frac{z_i + z_{i+1}}{2}|\mathbf{Z}^{k-1}, m)} \sum_{p=1}^{N_{part}} w_p p\left(\frac{z_i + z_{i+1}}{2} |\mathbf{X}_p, m \right)^\alpha \right\} . \end{aligned} \tag{3.25}$$

A natural way to choose the region boundaries z_i is to generate a training set of the z which will be encountered and define the regions using the LGB algorithm

to clustering the samples into N groups. Figure 3.14 shows simulation results for tracking three real targets when z is quantized to N levels ($N = 2, 4, \cdots 32$). As the figure illustrates, quantizing to two levels has very little impact on the overall filter performance.

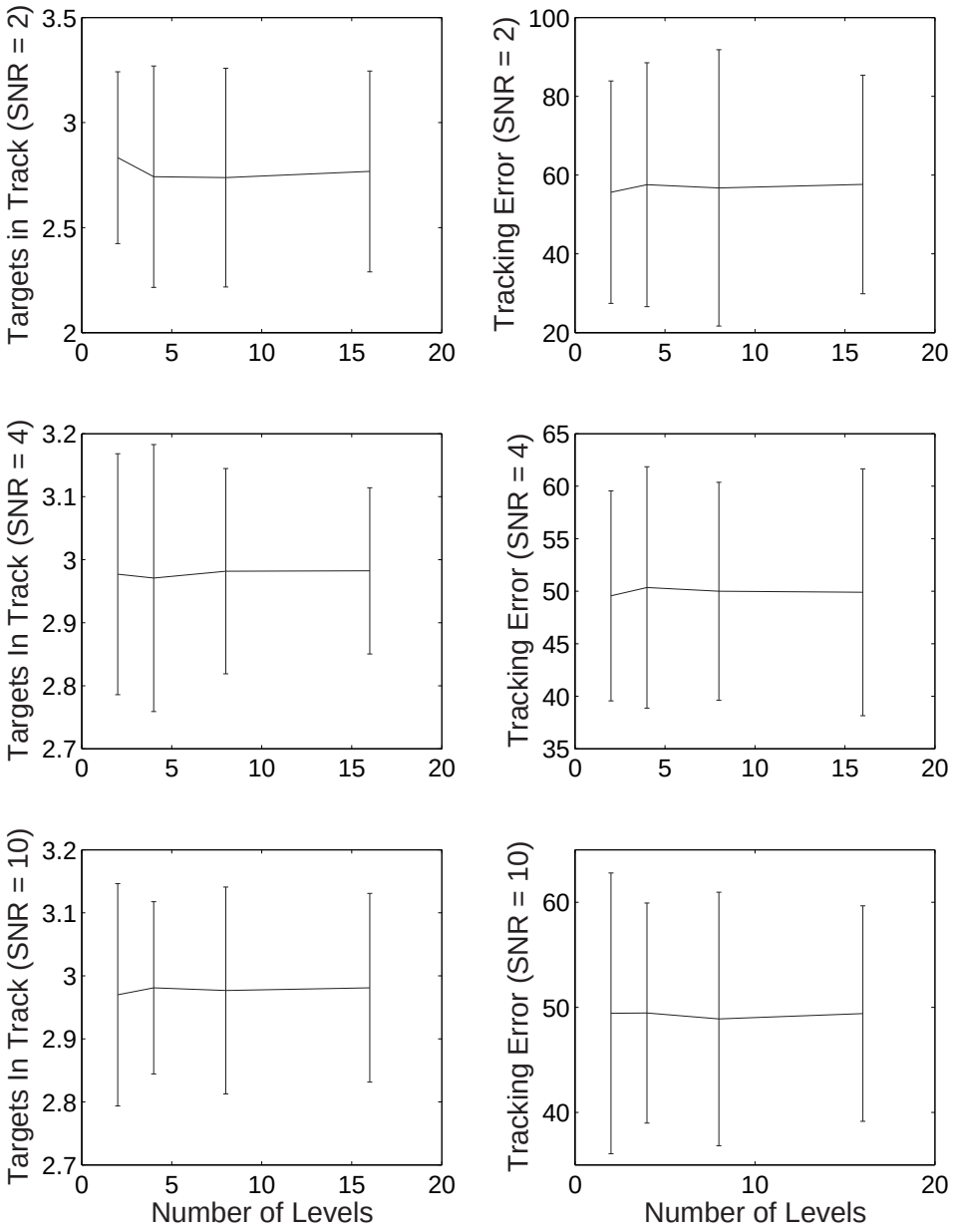


Figure 3.14: Performance of the algorithm using continuous valued measurements when the sensor manager approximates the expectation with a finite sum. Each graph shows the estimated number of targets (true target number is 3) and the median tracking error versus number of quantization levels. The algorithm performs quite well even when the measurements are quantized to 2 levels.

CHAPTER IV

Non-myopic Information Based Sensor Management

In this chapter, we extend the information based sensor resource allocation strategy of Chapter III to scenarios in which long term scheduling is important. We maintain the philosophy that actions should be chosen so as to maximize the expected amount of information extracted from the scene. The extension developed here is to fold in the long term effect of current decisions, and hence schedule the sensor to maximize expected information gain over a (potentially infinitely long) horizon rather than merely for a single step. This extension dramatically increases the computational burden of the problem. We address this by developing several approximate solution techniques.

This chapter proceeds as follows. First, in Section 4.1, we provide two motivating examples where long-term scheduling will provide a benefit over myopic scheduling in the multitarget detection, tracking, and identification setting studied here. Second, in Section 4.2, we present several methods of addressing the non-myopic scheduling problem. Specifically, in Sections 4.2.2 and 4.2.3 we investigate and dismiss as too computationally burdensome various methods of enumerating action sequences and computing via rollout the expected long term value. These methods are discussed and evaluated to give a baseline of performance to which other approximate methods

will be compared. An alternate method, which is based on directly approximating the long-term value of current actions through a penalty function, is given in Section 4.2.4. This method is motivated as a direct approximation to the value-to-go term in Bellman's optimality equation and has the virtue that it does not require explicit enumeration of action sequences. While it does not apply to all problems, it is applicable to a broad set of problems including those considered in the motivation section. Finally, in Section 4.2.5, we investigate an entirely different approach to the problem which is based on learning the best policy through a set of training examples. This broadly applicable technique requires a training set and a large amount of (potentially off-line) computation but is useful for providing bounds on approximation strategies as well as addressing problems not in the class of those covered by the value-to-go approximation. We conclude the chapter in Section 4.3, by presenting simulation results on several model problems that compare and contrast the various approaches and illustrate the performance gains possible with long term scheduling.

4.1 Motivating Examples for Non-myopic Scheduling

In this section, we describe two scenarios where non-myopic multistage scheduling will outperform myopic single stage scheduling as a means of motivating the solution approaches discussed later. Both examples are situations where the dynamics of the process is time-varying and evolving in a predictable manner. It is then possible to predict that the information gathering ability of a particular action will be reduced at a later time. Therefore, a policy which considers the long term impact of sensing actions will lead to the best policy.

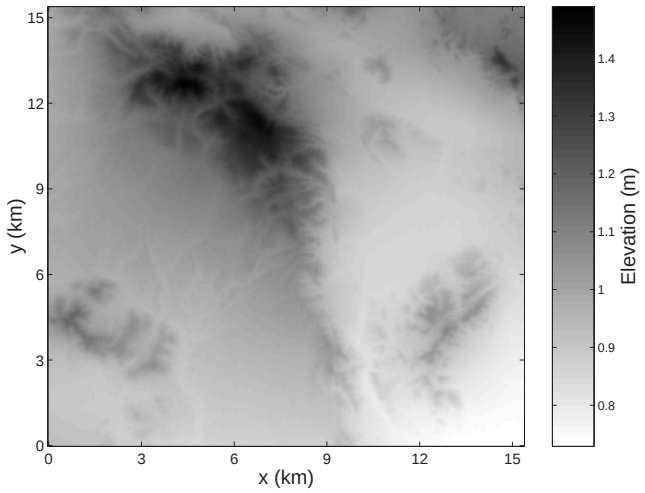
Motivating Example 1 : Time Varying Visibility Maps

This model problem considers a moving airborne sensor that is able to image a portion of a ground surveillance area. As in previous model problems, the goal is to determine the number of targets in the surveillance area and the state of each target. The setup is similar to that of the model problems considered in Chapters II and III, but we add in the additional constraint that the surveillance area has elevated terrain and the sensor is moving.

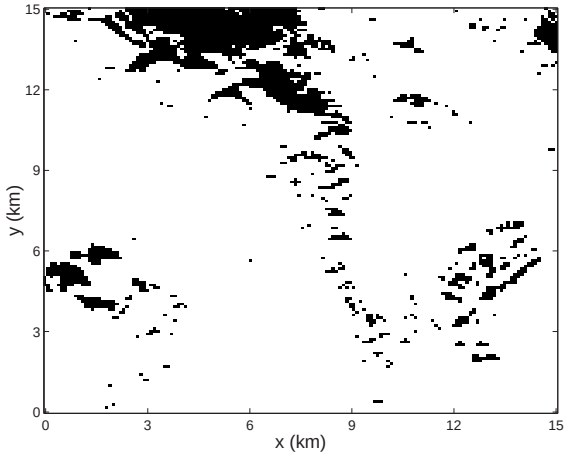
This additional constraint means that at each time step, the sensor position relative to the surveillance area causes certain portions of the ground to be unobservable. Given the sensor position and the terrain elevation, we can compute a visibility mask which determines how well a particular spot on the ground can be seen by the sensor. As an example, in Figure 4.1 we give the visibility masks computed from a sensor positioned below and to the left of a surveillance area characterized by a particular elevation map.

Visibility constraints enter into the sensor management formulation through the sensor model $p(\mathbf{z}|\mathbf{X}, T)$. In particular, if a cell is partially obscured, the probability of a non-detection in that cell is high regardless of the cell occupancy. Therefore, interrogating a partially obscured cell has small expected information gain (if the cell is completely obscured, interrogating the cell will have 0 expected information gain). Hence, any policy (including a myopic policy) will prefer not to task the sensor to interrogate obscured areas.

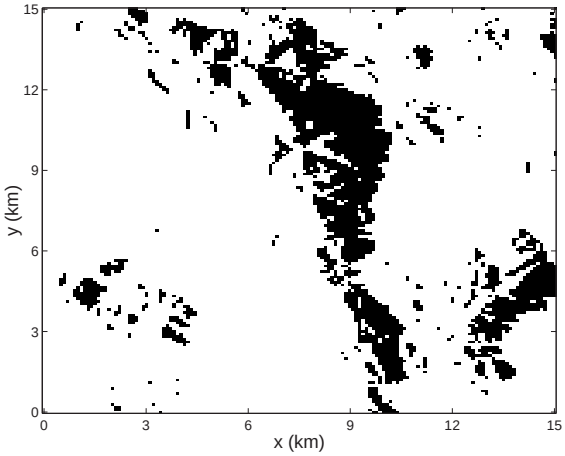
Non-myopic sensor management will improve performance over myopic sensor management in this situation as it is able to take actions to counteract future visibility constraints. A specific example is when a target is predicted to become obscured to the sensor for a brief amount of time. In this case, extra sensor dwells immediately



(a) Elevation map of surveillance region



(b) Visibility mask corresponding to a sensor positioned below the surveillance region



(c) Visibility mask corresponding to a sensor positioned to the left of the surveillance region

Figure 4.1: Visibility masks for a sensor positioned below and to the left of the surveillance region, along with the elevation map of the region. In this example, we show binary visibility masks (non-visible areas are black and visible areas are white). However, in general, visibility may be between 0 and 1 indicating areas of reduced visibility, e.g., an area partially obscured by foliage.

before the target enters into the obscured area (at the expense of not interrogating other targets) will sharpen the estimate of target location. This sharpened estimate will allow better prediction of where and when the target will emerge. This is illustrated graphically with a six time-step vignette in Figure 4.2.

Notice that a myopic sensor management scheme will not be able to take advan-

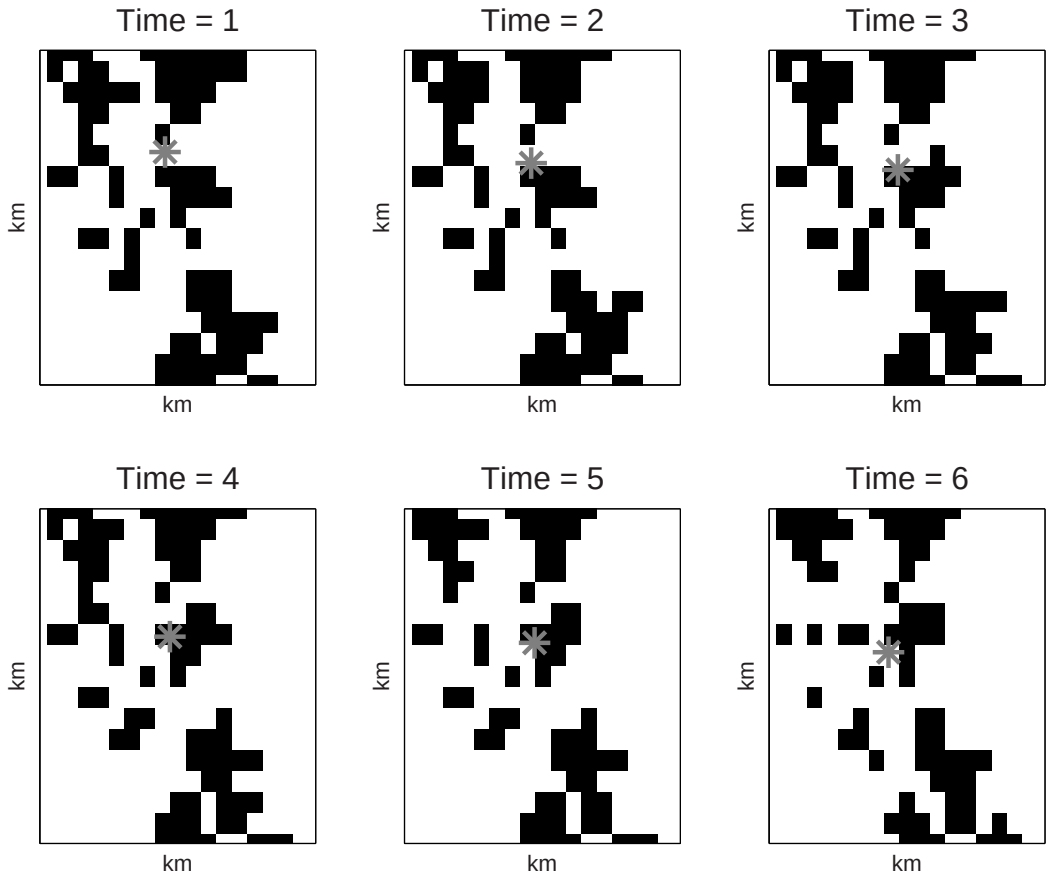


Figure 4.2: A motivating example for non-myopic optimization. During the six time step vignette, the target moves through an obscured area. The target is depicted by an asterisk. Obscured areas are in black and visible areas are in white. Extra dwells just before becoming obscured (time = 1) should aid in relocalization after the target emerges (time = 6).

tage of a predictable change in visibility. By definition, the myopic scheduler only considers a 1-step prediction, and an action is taken based on maximizing expected gain in information after the measurement is made. On the other hand, a full long-term (non-myopic) strategy will choose the sequence of actions that leads to the most overall gain in information when the present action provides optimal information to all future sensor actions. In this case, that may mean taking actions corresponding to small immediate gain in information for the purposes of maximizing long term gain in information.

Motivating Example 2 : Multiple Cell Occupancy

A second motivating example is the situation where multiple targets come within sensor resolution of each other. These situations are common in military scenarios where a coarse sensor is used to interrogate convoys (i.e., vehicles following each other closely). Simulations with convoy movement have been already considered with the myopic sensor manager (e.g., in Sections 3.4.1, 3.4.2, and 3.4.3).

Again, the sensor model $p(\mathbf{z}|\mathbf{X}, T)$ describes the performance of the sensor when such events happen. For the MTI and FTI sensor simulations considered earlier, the sensor was modeled as having an elevated energy return in any multiply occupied cell. In particular, the energy returned from each target adds coherently leading to an increased detection probability at a fixed false alarm rate. On the other hand, for the ID sensor, the sensor performs poorly when multiple targets occupy the same detection cell. Since the ID sensor is modeled as some type SAR-based ATR system, the individual scattering centers of the particular target are important for ID. When multiple targets are closely spaced, the radar returns add, causing the ATR algorithm to perform very poorly.

Due to this reduced utility of the sensor when multiple targets enter a single sensor cell, it is important to plan ahead when it is predicted that targets will come close together in the future. Specifically, if targets are moving predictably toward occupying a single cell, the targets should have higher priority for interrogation as ability to gain information about these targets is soon to diminish. A myopic strategy does not account for this as it relies only on single-step information gain. A non-myopic strategy, on the other hand has the ability to plan many steps ahead and take actions now that will maximize the long term benefit.

4.2 Approximate Non-myopic Scheduling

The examples illustrated above show that long term sensor scheduling is beneficial in situations where the future information gathering ability changes in a predictable manner. However, non-myopic scheduling presents a significant computational challenge, preventing an exact solution in all but the simplest of situations.

If computational resources were no limit, the optimal solution would be to enumerate every possible sequence of future actions that could be made starting at the current time, e.g., every sequence $\mathbf{a}$ of the form $\mathbf{a} = (a_1^k, a_2^{k+1}, \cdots a_T^{k+T})$, and choose the sequence that leads to the best expected long term information gain. Obviously, this cannot be done for any real problem in practice as the number of sequences is exponential in the number of possible actions M and the number of time steps one schedules ahead T, i.e., the computations required are $O(M^T)$. We investigate this brute force method further in Section 4.2.2 as a means of providing a baseline for the best achievable algorithm performance in a small model problem.

A more efficient strategy is to prune the action sequences that are investigated so as to only consider those that are potentially fruitful. There are many such strategies in the literature, including [68][96]. In Section 4.2.3 we investigate one such method, where the measure of the utility of investigating a path is again based on its expected information content. While providing a computational speedup, this algorithm is still exponential in the possible actions and horizon length and therefore not an acceptable solution.

Since both of these methods are too computationally complex for application in realistic problems, we turn our attention to a method that does not explicitly enumerate action sequences. The method is predicated on approximating the long

term ramifications of taking a particular action at the current time. A penalty is applied to actions that are not good for the long term, and vice versa. This makes actions that are rewarding due to future considerations more desirable to choose at the current time step, thus approximating the non-myopic decision. The algorithm, which we will refer to as the value-to-go approximation, is $O(N_{part} * M * T)$ and is discussed in Section 4.2.4.

A final method of choosing actions in a non-myopic fashion that we consider here is reinforcement learning. The idea here is to learn the value of actions from a large set of training examples. This can be done off-line via a large set of training examples before a mission begins. Alternatively, methods of this type can be accomplished on-line via policy improvement. The application of learning methods to our problem, along with the advantages and limitations of such an approach, is discussed in Section 4.2.5.

4.2.1 Notation and Preliminaries

In this section, we introduce the notation necessary to address the long term scheduling problem. We cast the problem into a Markov Decision Process (MDP) framework. This will allow us to explicitly demonstrate the qualitative statements made earlier about the exponential growth in computational requirements for solving the non-myopic scheduling problem exactly. It will also motivate the approximate methods discussed later.

Long-term scheduling may be finite horizon or infinite horizon. The former is appropriate for problems which have some time limit or terminal point where a decision must be made. In this case, one typically desires to get the maximum reward over the finite horizon. On the other hand, infinite horizon scheduling is appropriate for scenarios that do not have a natural end time. In this case, one schedules to

maximize a weighted sum of rewards where the weight is chosen to exponentially suppress future rewards relative to current rewards. The infinite horizon setup is most appropriate for the model problems considered in this work.

We denote the state of the system at time t by S_t. The state of a system includes everything required to characterize the problem at a particular time instant. In our setting, this includes the JMPD, hospitability and visibility maps, and sensor models among other things.

The goal is to find the best *policy* Π^*, which over the long term is the most rewarding mapping from states to actions. The value function associated with any policy Π, denoted $V^{\Pi}(s)$, is the expected total discounted reward when being in state $S_0 = s$ and following the actions prescribed by policy Π, i.e.,

$$V^{\Pi}(s) = E\Big\{ \sum_{t=0}^{\infty} \gamma^t r_t(S_t, \Pi(S_t)) | S_0 = s \Big\} \ , \tag{4.1}$$

where $r_t(S_t, \Pi(S_t))$ is an immediate reward, and γ is a discount factor included to value future rewards less than immediate rewards. An optimal policy is a policy that satisfies

$$\Pi^*(s) = arg \max_{\Pi \in P} V^{\Pi}(s), \forall s \in S \ . \tag{4.2}$$

where P is the set of all possible policies (mappings from states to actions).

The optimal policy is the unique solution to Bellman's equation,

$$V^*(s) = \max_a E\Big\{ r_t(S_t, a) + \gamma V^*(S_{t+1}) | S_t = s, A_t = a \Big\} \ , \tag{4.3}$$

which says in state s, the action $\hat{a}$ is to be chosen as

$$\hat{a} = arg \max_a E\Big\{ r(s, a) + \gamma V^*(s') \Big\} \ . \tag{4.4}$$

The MDP formulation requires a definition of state of the system s as well as the reward structure $r(s, a)$.

As mentioned earlier, the JMPD, kinematic and sensor models, and ancillary information all contribute to the system state. We will see that for the methods investigated here (with the exception of the reinforcement learning approach discussed later) there is no need to specify a fixed finite dimension state vector.

In all cases, we choose to use as immediate reward the one-step (myopic) gain associated with action a. This choice is a direct extension of the myopic sensor management approach, i.e.,

$$r(s,a) \doteq D_\alpha \left(p(\cdot|\mathbf{Z}^{k+1}) || p(\cdot|\mathbf{Z}^k) \right) \quad . \tag{4.5}$$

In principle, one can find the optimal policy via dynamic programming methods when the state space and action space is discrete and of low cardinality. However, when the number of states or actions becomes large the solution becomes computationally intractable. In our setting, the situation is even more challenging as the state space is continuous and of large dimension. Therefore, approximate techniques for determining the policy Π^* are required. In the following subsections, we discuss several methods of approximating Π^*.

4.2.2 Monte Carlo Rollout for Non-myopic Sensor Management

We first investigate a straightforward Monte Carlo (MC) technique that considers all action sequences of the form $(a_1^k, a_2^{k+1}, \cdots a_T^{k+T})$ and computes the expected information gain for the sequence by repeatedly simulating its application and computing the average gain acquired. While straightforward to describe, this method has a computational burden of $O(P * N_{part} * M^T)$, where P is the number of Monte Carlo simulations of each action sequence, M is the number of actions at each time and T is the number of time steps the algorithm looks ahead. This method does have the nice property that its complexity is independent of the size of the state

space. However, it is still exponential in the number of actions and time horizon. Due to this computational complexity, it is not a viable solution to the long term scheduling problem. However, it is useful as it will allow us to judge the performance of approximate techniques to be developed later (on very small model problems).

The term "rollout" has been used to describe this technique by Tesauro [97] in the context of determining good strategies for playing the game backgammon. It is used there as a synonym for repeatedly playing out a given position in order to calculate the expected reward starting from that position. A similar strategy has been used in the computer science literature under the name "sparse sampling" [98]. For simplicity, we first describe the two-step non-myopic solution in this section and comment on the extension to multiple time steps later.

The two-step rollout procedure is shown graphically in Figure 4.3. We first predict the target density at the measurement time $(k+1)$ by performing model update as in the myopic scheme. The prediction density, $p(\mathbf{X}^{k+1}, T^{k+1}|\mathbf{Z}^k)$ is used to determine all possible actions at time $k+1$, $a_1^{k+1} \cdots a_N^{k+1}$.

For each action at time $k+1$, we perform the following two steps repeatedly to generate a MC average of the information gain, which is used to approximate the expected value. First, the action is simulated resulting in a measurement $\hat{z}^{k+1}$. The density of $\hat{z}^{k+1}$ is formed from $p(\mathbf{X}^{k+1}, T^{k+1}|\mathbf{Z}^k)$, as in (3.12). The simulated measurement is then used to update the density and form $p(\mathbf{X}^{k+1}, T^{k+1}|\mathbf{Z}^k, \hat{z}^{k+1})$. The realized gain in information from this measurement is calculated between the densities $p(\mathbf{X}^{k+1}, T^{k+1}|\mathbf{Z}^k)$ and $p(\mathbf{X}^{k+1}, T^{k+1}|\mathbf{Z}^k, \hat{z}^{k+1})$ using (3.1).

This predicted posterior is then model updated to form the prediction density at time $k+2$, $p(\mathbf{X}^{k+2}, T^{k+2}|\mathbf{Z}^k, \hat{z}^{k+1})$. At this point, the expected one-step (myopic) gains for each possible action at time $k+2$ is generated using (3.14). The value of

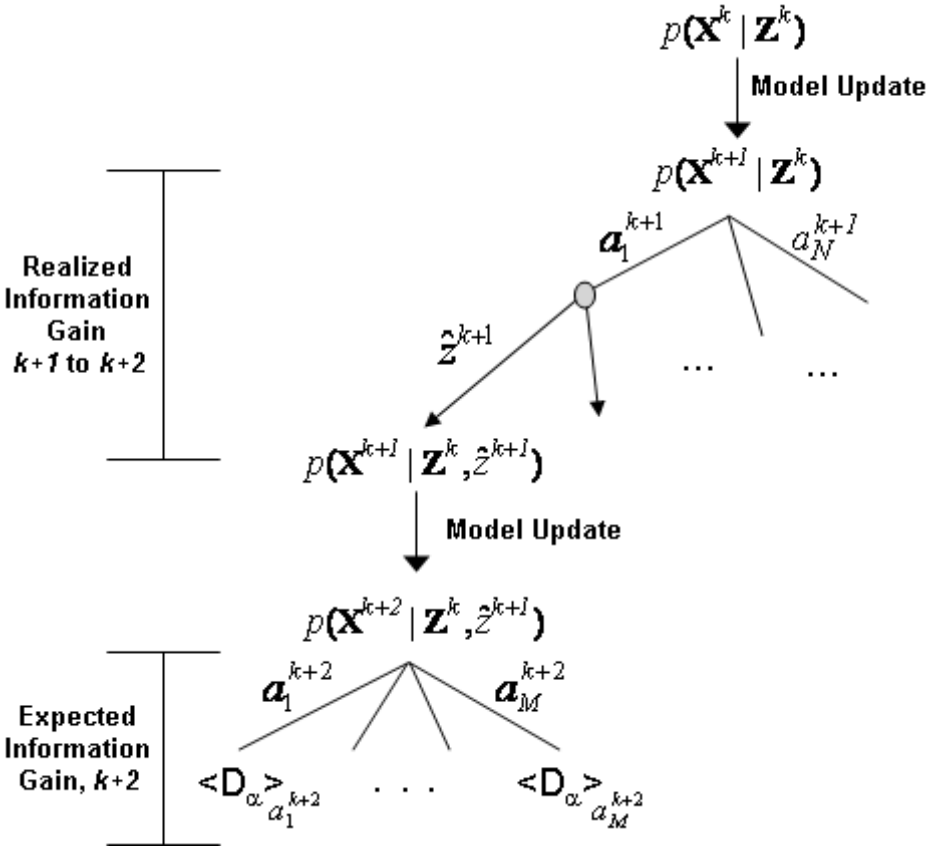


Figure 4.3: A two-step rollout approach to non-myopic scheduling. The value of an action at time $k+1$ is taken to be the realized gain from the action plus the expected gain at the next step. This procedure is run many times to generate a MC average of the two-step gain for each action a_i^{k+1}.

action a_i^{k+1} is then the actual realized gain from time step $k+1$ to time step $k+2$ plus the mean of the expected gain at time $k+2$. We call this 2-step procedure "searching the path associated with the action a_i^{k+1}".

The extension to looking more than two time steps into the future is straightforward but computationally prohibitive – as mentioned earlier the algorithm is exponential in the number of actions and the horizon. For example, a three-step rollout would perform an additional simulation step using $p(\mathbf{X}^{k+2}, T^{k+2}|\mathbf{Z}^k, \hat{z}^{k+1})$ to simulate a measurement $\hat{z}^{k+2}$ at time $k+2$. This would generate a predicted posterior at time $k+2$, $p(\mathbf{X}^{k+2}, T^{k+2}|\mathbf{Z}^k, \hat{z}^{k+1}, \hat{z}^{k+2})$. A model update would form $p(\mathbf{X}^{k+3}, T^{k+3}|\mathbf{Z}^k, \hat{z}^{k+1}, \hat{z}^{k+2})$, and the expected myopic gain at time $k+3$ would be

calculated. This procedure would be repeated for each action at time $k+1$ many times to generate a MC average of the expected gain for making that measurement.

A related approach has recently been given by Chong [99]. The main difference is that only "partial" rollout is used. All actions at the first level of the tree are considered. From there, rather than considering all possible next actions, a base policy is executed. The performance in large part depends on good choice of the base policy. Furthermore, Chhetri [68] performs a tree search but uses only the maximum likelihood estimate of $\hat{z}$ rather than averaging over realizations of z.

4.2.3 Adaptive Trajectory Selection for Improved Monte Carlo Rollout

In this section, we describe a method of performing the MC rollout discussed above where we restrict ourselves to searching down the tree only a small number of times. Given this computational budget, we wish to adaptively determine the best trajectories to investigate. We find that this method reduces computational expense required to achieve a certain level of performance, but does not achieve the computational simplification required for tractability.

At time $k+1$, there are M possible actions. Each action corresponds to the first step in a trajectory down the tree. Associated with each action is an expected (long-term) gain in information for executing that action, and we wish to determine this as precisely as possible. In Section 4.2.2, we determined this gain by simply searching down each path many times and using the empirical average of information gain as a surrogate for the expected information gain.

Here we wish to select the paths to search so as to best estimate the expected information gain with a fixed number of samples. We propose to select the best trajectory to simulate by computing the gain in information about the trajectory that making an additional simulation will garner. This will provide an automatic method

to prune trajectories – i.e., decide which paths are not worth further investigation and which paths deserve greater attention.

We define $p_{a_i}(g|G_{a_i})$ to be a density on the expected long-term gain in information g if we were to actually take action a_i, conditioned on the long-term information gains simulated so far from searching down trajectories starting with action a_i, G_{a_i}. Of course, at beginning of each decision epoch, we will have not searched any trajectories yet and so $G_{a_i} = \emptyset$. Our goal is to determine $p_{a_i}(g|G_{a_i})$ for all actions a_i as accurately as possible using a fixed search budget, so that when we actually task the sensor we are tasking it to make the action that maximizes the expected long-term gain in information. At the onset, we have N possible actions and no idea which action is the best to take. We propose to construct the initial density on the expected long-term information gain for actually taking action a_i by looking down the trajectory associated with action a_i a small number of times (P) to generate samples from the density $p_{a_i}(g|G_{a_i})$. These samples from $p_{a_i}(g|G_{a_i})$ will be used to approximate $p_{a_i}(g|G_{a_i})$ in a particle filter like manner, e.g., $p_{a_i}(g|G_{a_i}) = \frac{1}{P}\sum_{p=1}^{P}\delta(g - g_p)$.

We then wish to simulate an additional K trajectories to improve our estimate of the expected long term information gain when taking action a_i, $p_{a_i}(g|G_{a_i})$. We use an information directed method for selecting which trajectory to investigate for each of the K investigations. The method proceeds as follows. For each action a_i, we compute the expected gain in information with respect to $p_{a_i}(g|G_{a_i})$ that making one additional simulation of that action will garner. Then we investigate that path that generates the largest expected gain in information. We repeat this procedure for all K investigations that we are to make.

Formally, we can compute the expected gain in information for investigating action a_i as follows. Before investigating a new path, we have a density $p_{a_i}(g|G_{a_i})$. Assume

that we have decided to investigate a particular action and this investigation has generated a new realization of the expected long-term gain $\hat{g}$. The updated density (by Bayes' rule) becomes

$$p_{a_i}(g|G_{a_i}, \hat{g}) = \frac{p_{a_i}(\hat{g}|g)p_{a_i}(g|G_{a_i})}{p_{a_i}(\hat{g}|G_{a_i})} \quad . \tag{4.6}$$

Using the Alpha-Divergence metric (3.1), and a method identical to that of Section 3.2, we can determine that the expected gain in information between $p_{a_i}(g|G_{a_i})$ and $p_{a_i}(g|G_{a_i}, \hat{g})$ for searching the trajectory starting with action a_i is proportional to the entropy of the distribution associated with that action,

$$\int_g p_{a_i}(g|G_{a_i}) ln\left(p_{a_i}(g|G_{a_i})\right) dg \quad , \tag{4.7}$$

which is the intuitive result that the best trajectory to search is the trajectory associated with the highest uncertainty (entropy). The method then proceeds as follows. Given a fixed computational (time) budget, we first investigate each possible action a small number of times (e.g., 10). We then use the remaining of the allotted time to select which trajectory to investigate based on that path that has the highest entropy. In expectation, this strategy maximally increases information about the paths. After the time budget has been exhausted we choose to make the measurement with highest expected long-term information gain.

A related approach has been considered by [68]. The approach considers reducing the trajectories searched via branch and bound techniques rather than the information directed strategy discussed here.

4.2.4 Direct Approximation of the Value-to-go Function

In this section, we look at the non-myopic scheduling problem from a different viewpoint [100]. Instead of enumerating action sequences and searching trajectories,

we instead work to devise a penalty function that incorporates the long term effect of taking a current action. Specifically, this method directly approximates the second term on the right hand side of equations (4.3) and (4.4), the long term value (also known as the value-to-go). While not applicable to all scenarios, this method applies to a broad class of situations commonly encountered in the multitarget tracking problem. When applicable, it has the potential to be a computationally efficient method of biasing actions towards those that are beneficial in the long term.

The strategy is predicated on the following observations. First, if by waiting to perform an action until a later time step the ability to gain (myopic) information decreases, the action should have higher priority to perform now. Conversely, if the ability to gain (myopic) information is greater in the future, the action should be delayed.

The approximation we advocate is to replace the value-to-go term with an information based quantity which gives the difference between the expected myopic information gaining capability at the current time and the expected myopic information gaining capability at a future time. Intuitively, this captures the "opportunity cost" or "regret" for not taking an action at the current time.

For a concrete example, consider the case of time varying visibility. If an area is predicted to be less visible in a future time step, the desire to interrogate it at this time step should be enhanced. Conversely, if an area is predicted to be more visible in a future time step, the desire to interrogate it at this time step should be supressed. Of course, more than just visibility must be accounted for. The expected future occupancy and expected future uncertainty is relevant as well. As it is based on the expected gain in information available, the proposed method nicely accommodates all of these factors simultaneously.

This technique applies to a variety of other scenarios. For example, consider the convoy-movement scenario. By using kinematic prediction, one may be able to determine that two targets are about to come close together (e.g., enter the same sensor detection cell). This signals reduced ability to gain information about those targets in the future and therefore the targets should be interrogated at the current time step.

To specify the technique precisely, recall that the optimal method for choosing the action to make at the current time is given by (4.3). We wish to approximate the value-to-go term, $E[V^*(s')]$, by a function $N(s,a)$ which captures the long term reward of action a in state s and is easily computable. That is, we make the approximation that

$$\begin{aligned} \hat{a} &= arg\max_a \left\{ E\Big[r(s,a)\Big] + \gamma E\Big[V^*(s')\Big] \right\} \\ \hat{a} &\approx arg\max_a \left\{ E\Big[r(s,a)\Big] + N(s,a) \right\} \ . \end{aligned} \tag{4.8}$$

We propose to use as $N(s,a)$ the "gain in information for waiting". Specifically, let $\bar{g}_a^k$ denote the expected myopic gain when taking action a at time k. Furthermore, let $p_a^k(\cdot)$ denote the distribution of myopic gains when taking action a at time k. Then we approximate the long-term value of taking action a by the gain (loss) in information received by waiting until a future time step to take the action,

$$N(s,a) \approx \sum_{t=1}^{T} \gamma^t sgn\big(\bar{g}_a^k - \bar{g}_a^{k+t}\big) D_\alpha\big(p_a^k(\cdot)||p_a^{k+t}(\cdot)\big) \ , \tag{4.9}$$

where T is selected as the number of time steps in the future that are to be considered (i.e., the horizon length).

Each term in the summand has two components. First, $sgn\big(\bar{g}_a^k - \bar{g}_a^{k+t}\big)$ is a sign term which signifies if the expected information gain in the future is more or less than the expected information gain in the present. A negative value implies that the future

presents a better opportunity and that the action ought to be discouraged at present. A positive value implies that the future presents a poorer opportunity and that the action ought to be encouraged at present. The second term, $D_\alpha\big(p_a^k(\cdot)||p_a^{k+t}(\cdot)\big)$ measures the Rényi divergence between the density on myopic gains at the current time step and the density on myopic gains t time steps in the future. What results is a magnitude of the difference between the two densities. A small number implies the two are very similar and therefore the non-myopic term will have little impact on the decision making. Conversely, a large number implies the two are very different and the non-myopic term should contribute strongly to the decision making. The product of these terms gives the magnitude and direction in which the information gain is changing with time.

A practical problem involved with implementing this method is determining densities on information gains $p_a^k(\cdot)$. We have found in the case where thresholded measurements are made, these densities can quickly and efficiently be approximated as Gaussian, which results in a nice closed form expression for $D_\alpha\big(p_a^k(\cdot)||p_a^{k+t}(\cdot)\big)$. Specifically, if $p \sim N(\mu, \sigma^2)$ and $q \sim N(m, s^2)$, the Rényi divergence between p and q can be shown to be given by

$$D_\alpha\big(p||q\big) = \frac{1}{\alpha - 1} ln \frac{e^{\frac{(m-\mu)^2}{2(s^2/(1-\alpha)+\sigma^2/\alpha)}}}{s^{-\alpha}\sigma^{\alpha-1}\sqrt{\sigma^2(1-\alpha)+\alpha s^2}} \ . \tag{4.10}$$

Therefore, the Rényi Divergence between the distribution on current and future information gaining ability is given in closed form under the Gaussian approximation.

To completely specify the approximatimation technique advocated here, we introduce a weighting w which gives relative precedence to the non-myopic and myopic terms in the approximation to Bellman's equation, i.e., the complete approximation

to (4.3) is given by

$$\hat{a} = arg\max_{a}\left(E\Big[r(s,a)\Big] + w\sum_{t=1}^{T}\gamma^{t}sgn\big(\bar{g}_{a}^{k} - \bar{g}_{a}^{k+t}\big)D_{\alpha}\Big(p_{a}^{k}(\cdot)||p_{a}^{k+t}(\cdot)\Big)\right) \quad . \tag{4.11}$$

As $w \to 0$ the approximate technique schedules myopically (ignores the future), and as $w \to \infty$ the technique schedules considering only the future (ignores the immediate benefit). An appropriate choice for w will balance the needs of the present with the needs of the future.

4.2.5 Reinforcement Learning for Non-myopic Scheduling

In this section, we investigate another approach to the non-myopic scheduling problem. Specifically, we look at reinforcement learning (RL) methods which use training episodes to learn an effective policy. This method has some advantages over the previous approaches. In particular, policies developed earlier are static, i.e., they cannot learn the dynamics of the problem automatically from experience or training data. In situations where there is model mismatch (e.g., poor estimates of sensor SNR) or if the sensor or target kinematics are changing with time (e.g., sensor drift) a RL approach may be more appropriate. The drawbacks to the RL approach are twofold: one must be able to generate training samples (either via off-line simulation or by learning while another policy is executed) and there is a large computational complexity associated with computing good policies. Training sample generation is problematic in that the real scene cannot typically be simulated before the mission. On-line learning is possible but requires policy exploration (i.e., deliberately taking non-optimal actions).

Q-learning [101], a special case of RL, is a popular and well studied method of learning the value function given in Section 4.2.1. For these reasons, we use it in our setting. First, define the *Q-function* $Q(s,a)$ as the (long-term) value for taking

action a when in state s. Q is directly related to the value function via

$$V(s) = \max_a Q(s, a) \tag{4.12}$$

Q-learning is a simple strategy where a set of training data is used to estimate the Q-function on-line. After sufficient refinement of the Q-function, it is then used to schedule the sensor by choosing

$$\hat{a} = arg \max_a Q(s, a). \tag{4.13}$$

Conceptually, one envisions a table which enumerates all possible discrete states s and discrete actions a. The number associated with each table entry, $Q(s, a)$, is the long-term value of taking action a when in state s. Q learning is a strategy where training examples consisting of state, action, next-state, immediate reward 4-tuples $\{s, a, s', r\}$ are used to learn the number in each table position. In the discrete action/state case, these training examples are used to learn the true mean value of taking the action via gradient descent as

$$Q(s, a) \Leftarrow (1 - \beta)Q(s, a) + \beta \left(r + \gamma V(s')\right) \quad . \tag{4.14}$$

Where β is a learning rate parameter and the value of the next state is computed using the Q-function approximation,

$$V(s') = \max_a Q(s', a) \quad . \tag{4.15}$$

This procedure is repeated again and again with training data. In practice, a modified version of this learning procedure is employed. A large set of training examples all corresponding to the same state are used to generate the $Q(s, a)$ that best fits all of the training examples. This is referred to as "batch" training and is typically done via a conjugate gradient type approximation. In the discrete state

case, under certain conditions on the learning rate and exploration of the training policy [102], it can be shown that the Q-learning algorithm converges to the optimal value function and hence the optimal scheduling policy.

However, in many target tracking problems (ours included) it is not possible to construct a small number of discrete states that completely capture the system state. Therefore, we use function approximation to represent the Q-function rather than a lookup table. The standard and simplest class of Q-function approximators are linear functions, i.e.,

$$Q(s,a) = \theta_a^T \phi_s \ . \tag{4.16}$$

Where ϕ_s is a feature vector associated with the state s and the coefficients of θ are to be estimated, i.e., the training data is used to learn the coefficients θ_a. Gradient descent is again used with the training data generate an to updated estimate of θ,

$$\theta_a \Leftarrow \theta_a + \beta \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \nabla_{\theta_a} Q(s,a) \ . \tag{4.17}$$

where for the linear function approximation, $\nabla_{\theta_a} Q(s,a) = \phi_s$.

We use Q-learning with linear function approximation to learn a policy which behaves non-myopically and is capable of dynamically adjusting to the scenario. The training process involves generation of {*estimated state, action, estimated next state, immediate reward*} 4-tuples over a large number of training episodes. This training data is then used to determine the coefficients of a linear function which represents the Q-function for each action. We use a conjugate gradient method for efficient training and the coefficients are chosen so as to minimize the mean squared error between the predicted reward and the received reward.

This is done in batch fashion, with the trained Q-function from the first set of episodes used to estimate the total (immediate + long term) reward from the second

set of episodes, and so on. In the training process, the value of an action is computed using the actual gain in information as measured by the Rényi Divergence.

Generating the feature vector ϕ is a challenging design problem for any learning algorithm. In our setting, the state of the system is characterized by the JMPD (which captures all of the uncertainty in the filter), the models of target kinematics and sensor effectiveness, as well as all ancillary information (e.g., terrain elevation maps). To generate the feature vector ϕ, we extract information-based features that provide a condensation of all of these quantities. Specifically, we use the expected myopic gains in information at time step k, for cells $c = 1 \cdots C$, $< D_\alpha >_c^k$, and the expected myopic gains in information at the next time step, $< D_\alpha >_c^{k+1}$, as features which characterize the state, i.e

$$\phi = \left[< D_\alpha >_1^k, \cdots, < D_\alpha >_C^k, < D_\alpha >_1^{k+1}, \cdots, < D_\alpha >_C^{k+1} \right] . \tag{4.18}$$

In the situation of time varying visibility, these features capture the immediate value of various actions and allow the system to learn the long term value by looking at the change in immediate value of the actions over time.

4.3 Simulation Results

In this section, we investigate several model problems to show the relative efficacy of the myopic scheme and the various approximate non-myopic schemes described above.

4.3.1 Target Localization with Time Varying Visibility

We investigate the following model problem, which is inspired by the first scenario described in Section 4.1. This problem is intentionally made to be simple with a small state space and a small action space so that the brute force method can be applied

and compared to the approximate methods. There are two targets which are each described by a one-dimensional position. Target 1 is initially positioned at $x = 2.1$ and Target 2 is initially positioned at $x = 14.9$.

For each dwell, the sensor may measure any one of 16 cells, each of which is 1 unit wide. The cell locations are fixed and centered at $.5, 1.5, \cdots, 15.5$ units. The sensor is allowed to make three (not necessarily distinct) dwells per time step. The sensor receives binary returns from the cell interrogated, which are independent from dwell to dwell. In cells that are occupied, a detection is received with probability P_d (set here at 0.9). In cells that are unoccupied a detection is received with probability P_f (set here at .01). This corresponds to a signal to noise ratio of 16dB, assuming Rayleigh distributed threshold detected returns.

At the onset, positions of the targets are known only probabilistically to the filter. The filter is initialized with the probability of target 1 location uniformly distributed across sensor cells $\{2 \cdots 6\}$ and the probability of target 2 location uniformly distributed across sensor cells $\{11 \cdots 15\}$.

The visibility of the various sensor cells is constructed to change in the following manner. At time step 1, all cells are visible to the sensor. At time steps $2, 3, 4$, cells $\{11 \cdots 15\}$ are invisible to the sensor. At time step 5 all cells are visible to the sensor again. This model problem closely emulates the situation where a target is initially visible to the sensor, becomes obscured, and then reemerges from the obscuration. This scenario can benefit from non-myopic scheduling as looking in the cells that are about to become obscured will minimize total track error at the end of the simulation.

At time 1 the myopic strategy, having no information about the future visibility, will choose cells uniformly from the set $\{2...6\} \cup \{11...15\}$. As a result, target 1

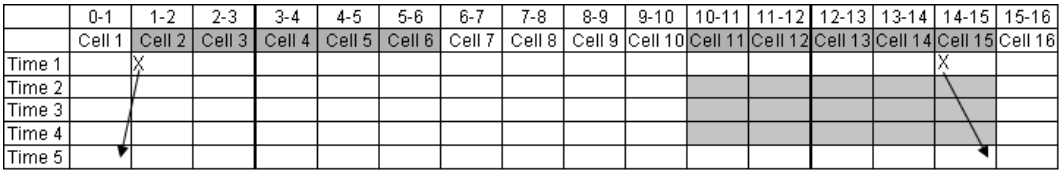


Figure 4.4: An illustration of the model problem with time varying visibility. At the onset, the filter has its position estimates of target 1 and target 2 uniformly distributed across cells $\{2...6\}$ and $\{11...15\}$, respectively. At time 1 all cells are visible to the sensor. At time 2, 3, and 4 cells $\{11...15\}$ are obscured. This situation emulates the situation where one target is initially visible to the sensor, becomes obscured and then reemerges.

and target 2 will on the average be given equal attention. The non-myopic strategy should preferentially choose cells from $\{11...15\}$ as they are to become invisible.

Results Using Full Monte Carlo Search and Information Directed Path Interrogation

We present a comparison between uniform searching of all paths as described in Section 4.2.2 with the information-directed search algorithm of Section 4.2.3 in Figure 4.5. Performance is compared in terms of median error after time 4 versus number of paths searched (which measures algorithm complexity). As expected, as either algorithm searches more paths, better decisions are made resulting in lower tracking error. However, uniform search requires more path interrogations to yield a desired error since it wastes investigations on paths of little value. The information directed method saves on the order of a factor of $2-4$ in computation time for a given error budget.

Results Using Approximation of Value-to-go

We illustrate here the performance of the information-based approximation to Bellman's equation given in Section 4.2.4. Figure 4.6 shows the tracking performance in the model problem as a function of the weighting of the value-to-go function, w. Included for reference is the myopic performance and the full Monte Carlo non-myopic performance.

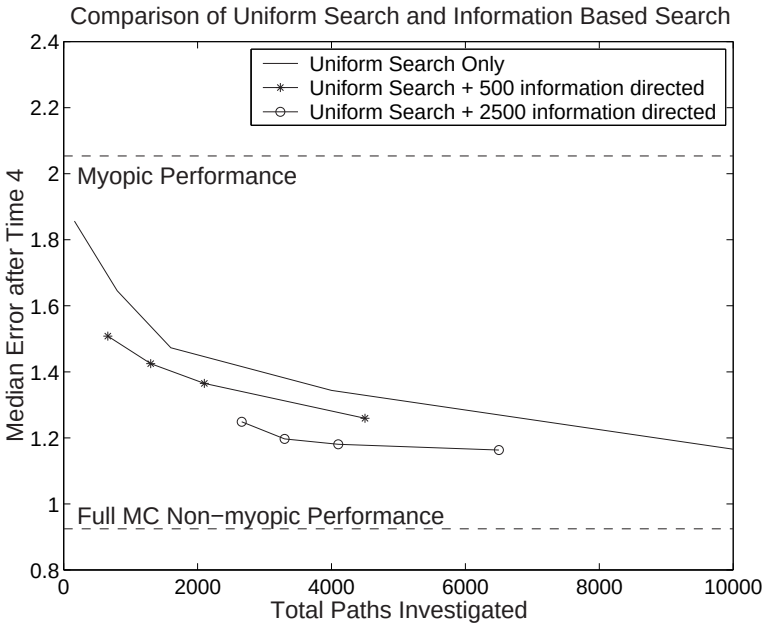


Figure 4.5: A comparison between uniform Monte Carlo and information-directed search. This graph contains three curves. The top curve contains the results of searching each path equally (uniform search) where the number of searches of each path. The bottom two curves are each seeded with uniform search and followed by information-directed searches. A comparison is made in terms of the total number of paths searched between the algorithms, which is a measure of algorithm complexity. For a given number of paths searched, information-directed search yields better performance. Shown for comparison are the results of myopic scheduling and an exhaustive non-myopic scheduling.

As mentioned earlier, at $w = 0$ the algorithm acts myopically and so the performance is identical to that of the myopic scheduler of Section 3.2. At large w, the algorithm takes actions based only on long term considerations (i.e., ignores the one-step value of an action, and is "overly" non-myopic). The resulting errors are slightly worse than being myopic. In between these two extremes, the proper trade-off between short term and long term considerations is made and the performance nearly reaches that of the exact non-myopic scheduler.

Comparison of myopic and approximate non-myopic strategies

In Table 4.1, we summarize the performance algorithms discussed in here in terms of computational complexity and tracking performance. We see that the value-to-go approximation provides performance similar to the full Monte Carlo search at a

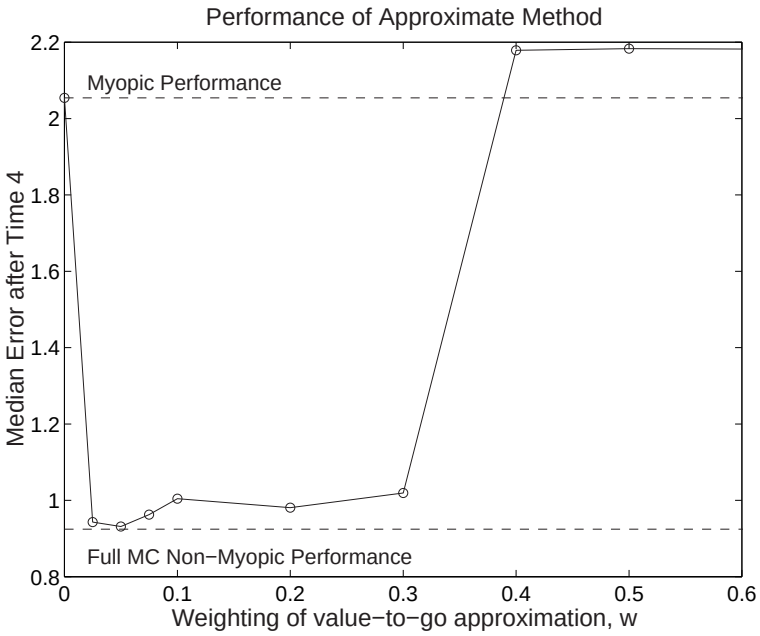


Figure 4.6: Performance of the approximate non-myopic scheduler as a function of the weighting of the value-to-go approximation, w. w weights the influence of the one-step value of an action with the long-term value. When chose properly, the two considerations are balanced and the performance equals that of the exact non-myopic scheduler.

computational cost similar to the myopic algorithm.

Table 4.1: Performance of the non-myopic scheduling algorithms in the case of time varying visibility.

Method	Description (Section)	CPU Time (sec)	Median Error (cells)
Myopic	3.2	0.189	2.054
Monte Carlo[a]	4.2.2	10.24	1.473
Monte Carlo	4.2.2	53.030	0.949
Monte Carlo	4.2.2	157.32	0.925
Information-Directed	4.2.3	8.458	1.249
Approximate ($w = 0.05$)	4.2.4	0.258	0.932

[a]MC Non-myopic shown for 250, 2500, and 5000 searches/path

Reinforcement Learning Methods

Figure 4.7 illustrates the performance of the Q-learning approach in comparison to the value-to-go approximation and a myopic strategy. The SNR in the model problem is slightly modified from that of Section 4.3.1 resulting in lower error for

each of the methods.

Q-learning is performed off-line via a large training set as discussed in Section 4.2.5. First, examples of states, actions, next states and rewards are generated over many training episodes. Next, the Q-function is trained using (batch) gradient descent to learn the best coefficients for θ. Finally, the trained Q-function is used to schedule the sensor.

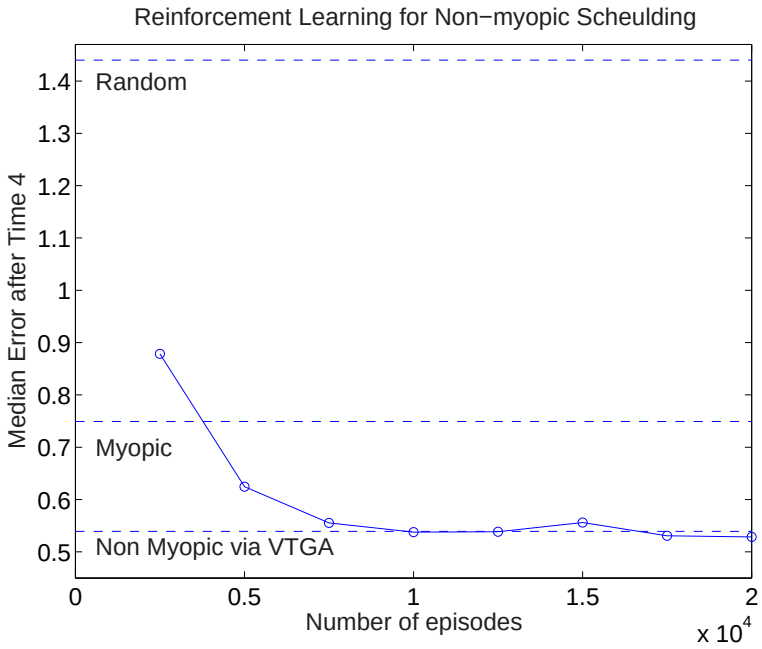


Figure 4.7: Performance of Q-learning versus random, myopic, and the value-to-go approximation in the model problem. The performance of the Q-learning algorithm is shown versus number of training episodes run. The performance converges to that of the value-to-go approximation with enough training episodes.

4.3.2 Target Localization with Time Varying Visibility and Real Targets

In this section, we consider a more realistic model problem which involves time varying visibility. Two targets move in a 5km x 5km surveillance region. The goal of the algorithm is to estimate the position and velocity of each target, i.e., estimate $[x, \dot{x}, y, \dot{y}]$ for targets 1 and 2. The target trajectories used in the simulation are taken from real, recorded target trajectories captured during the army training exercise

mentioned earlier (see Section 3.4 for a description of the exercise).

The sensor is able to image 100m x 100m cells on the ground, giving 2500 possible sensor actions. At each time step, the sensor can measure exactly one cell to determine the presence or absence of a target in the cell. With detection probability P_d, the sensor correctly detects a target in a cell. With false alarm probability P_f, the sensor incorrectly receives a detection from a cell that contains no targets. This is the usual thresholded Rayleigh detection model used throughout this work.

The sensor is moving, so the line of sight between the sensor and a detection cell is time varying. Coupled with terrain elevation, this results in the situation where the ability of the sensor to see a cell changes with time. In particular, target 2 is in an area of rapidly changing visibility. In this situation, the algorithm will benefit from non-myopic decisions, specifically those that encourage the sensor to measure cells that are about to become obscured.

At initialization, the filter has a (relatively) large uncertainty as to the target positions and is to learn the states while the targets are moving during a 50s episode. In each simulation, a random vehicle was chosen from the large database of recorded trajectories and placed randomly in this 500mx300m region. The filter only has the prior information that the target positions are uniformly distributed over a 500m x 300m region. Therefore, of the 2500 possible actions a much smaller number have any chance of yielding useful information during the course of the episode (the 15 cells from the initial prior plus surrounding cells, since the targets are moving). For that reason, the actions considered by the algorithm are pruned to a 6x6 block of cells (a 600m x 600m region on the ground) surrounding each target.

A myopic strategy makes tasking decisions based only on expected immediate reward. In particular, at the first time step all cells visible to the sensor are given

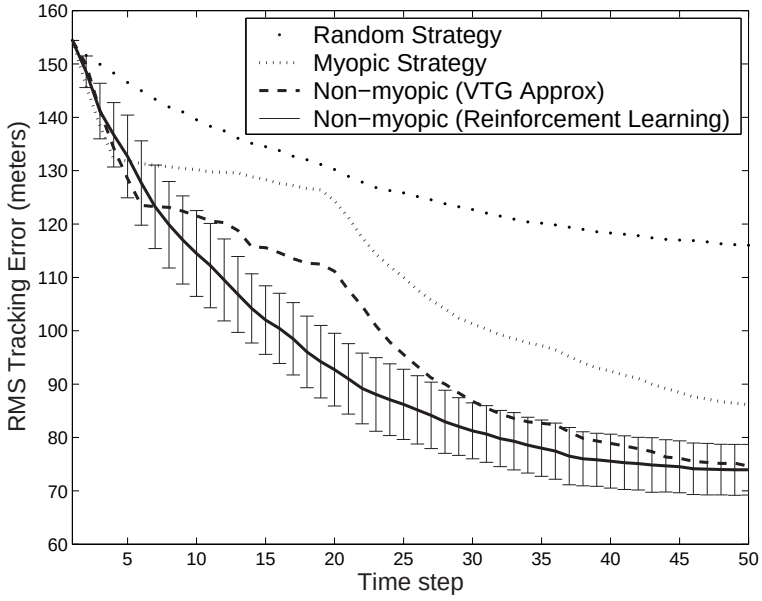


Figure 4.8: The performance of the approximate non-myopic scheduling policies on the realistic model problem described here. Performance is measured in terms of RMS tracking error (low is good) versus time. The approximation to Bellmans equation uses $w = 0.5$. The Q-learning strategy uses $500,000$ $\{state, action, next\ state, reward\}$ examples to learn a policy. As the policy learned (and hence performance) varies with the particular set of training examples, we show the mean and standard deviation of performance over 10 independent realizations of the training set. Shown for the purposes of comparison is the error for a purely random sensor allocation strategy, and the information based myopic strategy.

equal value. A non-myopic strategy, on the other hand, favors interrogating cells that will soon become obscured to the sensor.

In Figure 4.8, we present results of target localization in this scenario using the two approximate non-myopic scheduling algorithms advocated here : the information based reinforcement learning strategy and the value-to-go approximation. The Q-learning algorithm was trained on $10,000$ episodes (each consisting of 50 time steps) of the scenario. These 50 time step episodes record state, action, next state, and reward as mentioned earlier. In each of the training and testing episodes, the vehicle and start position was chosen randomly. Actions were chosen randomly during the simulation for maximal exploration of the state space. The value-to-go approximation

uses $w = 0.5$ as the weighting between immediate and future gains.

We see that the RL method performs best of all techniques considered here. However, the value-to-go approximation also significantly outperforms the myopic policy. Considering computational costs, this makes the value-to-go approximation a very powerful approach.

4.3.3 Target Classification with Multiple Occupancy

We turn our attention to a second model problem, which is related to the second scenario described in Section 4.1. There are four targets which are described by a one-dimensional position. The kinematic states (position and velocity) of the targets is known well. The goal is to point a target identification sensor so as to determine the correct type of each target. The classification sensor is modeled similarly as in Section 3.4.3. Specifically, cells that are multiply occupied produce random results when using the ID sensor. Furthermore, here we allow the misclassification rate of the sensor to be a variable p.

At time step 1, the targets are in separate sensor detection cells. The targets are moving so that at time step 2, two targets enter the same sensor detection cell. This situation is illustrated graphically in Figure 4.9.

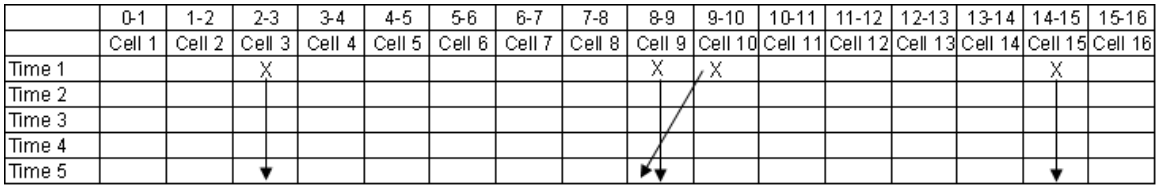


Figure 4.9: An illustration of the model problem with a multiple occupancy cell. At time 1, the filter knows the kinematic states of the targets well, but the identification is unknown. At time 2, two targets enter the same detection cell and stay there for the rest of the simulation. This situation corresponds to convoy movement.

At time 1 the myopic strategy, having no prediction about future cell occupancy, chooses to measure cells 3, 9, 10, and 15 with equal probability. As a result, at time

step 2 the targets in cells 9 and 10 may not have yet been interrogated and hence classification performance will be no better than chance (as the targets are now in a multiply occupied cell).

On the other hand, a non-myopic policy will preferentially interrogate cells that are about to become multiply occupied. Specifically, the approximate non-myopic algorithm which replaces the value-to-go function with an opportunity cost will realize that the ability to gain information at time step 2 is depressed for the targets in cells 9 and 10. This will force the sensor to preferentially measure these cells at time step 1.

We compare the performance of myopic scheduling with that of non-myopic scheduling and "overly" non-myopic scheduling by varying the weighting of the non-myopic term w in (4.11). The results are shown in Figure 4.3.3, where the performance is measured as a function of misclassification rate p and the number of looks available at each time step, L. We see that the non-myopic technique with properly chosen w (here $w = 0.5$) outperforms both the myopic ($w = 0$) and overly non-myopic ($w = 2$) strategies.

4.3.4 Tracking "Smart" Targets

In this subsection, we examine a problem involving "Smart" targets [103]. Smart targets are targets that are able to detect when they are under surveillance and react in a manner that makes future surveillance more difficult. We use the reinforcement learning approach to adaptively schedule a multi-modality sensor so as to most quickly and effectively detect the presence of smart targets and track them as they travel through a surveillance region. We investigate algorithms capable of choosing whether to use the active or passive mode of an agile sensor. The active mode is easily detected by the target, which makes the target prefer to move into

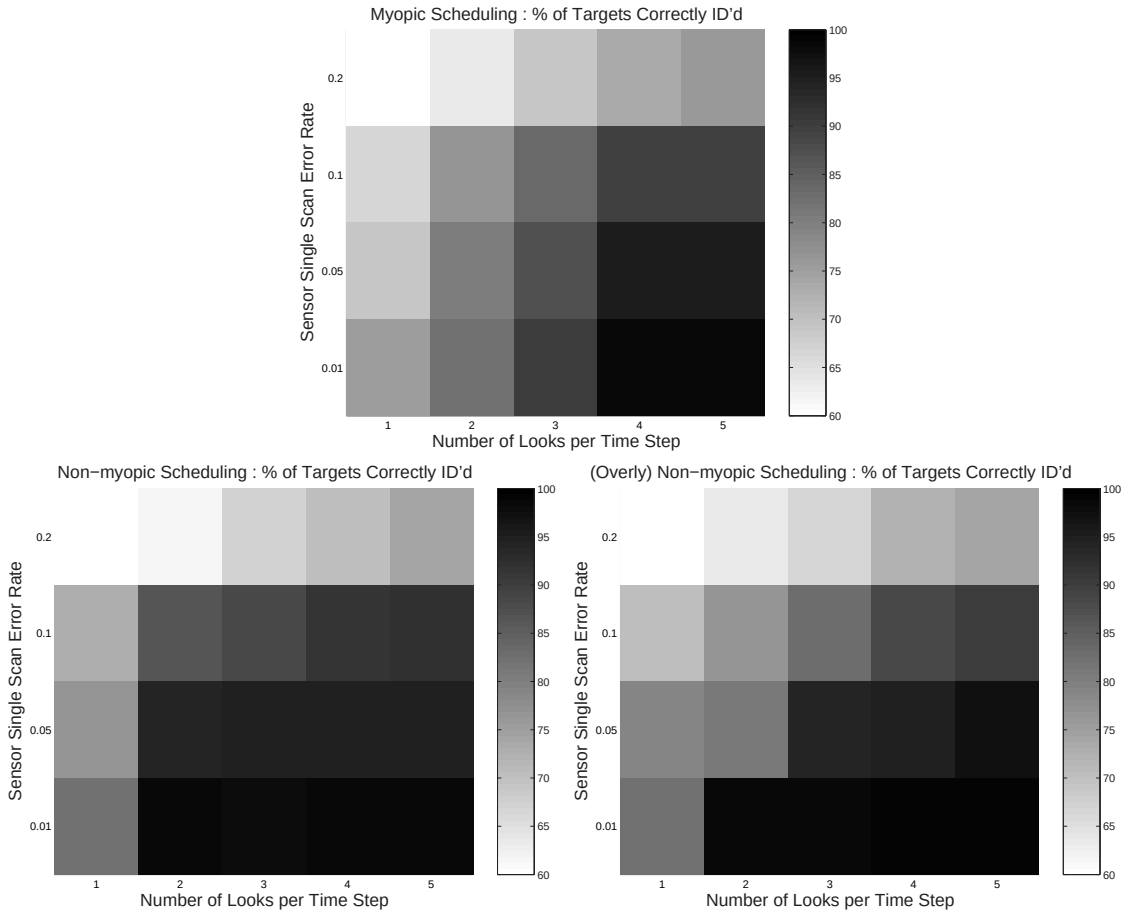


Figure 4.10: Performance of the information based approximate non-Myopic scheduler in the case of multiple occupancy. Performance is plotted as a function of the number of sensor dwells allowed at each time step and the single scan misclassification rate. The approximate non-myopic strategy outperforms the myopic strategy, particularly when the number of looks is small (e.g., 2)

hide mode. The passive mode is undetectable to the target. However, the active mode has substantially better detection and tracking capabilities then the passive mode. Using this setup, we characterize the advantage of a non-myopic policy with respect to myopic and random polices for multitarget detection and tracking.

The model problem is constructed as follows. There are two targets in the surveillance region with unknown initial position. At each time step, the sensor is able to measure a single cell to determine the presence or absence of targets. The sensor can use active mode or passive mode. Sensor modes are again characterized by a

detection probability P_d and a false alarm probability P_f dictated by a Rayleigh assumption on target returns.

When the target is in visible mode, the active mode works with high detection probability and low false alarm probability, $P_d = .9$ and $P_f = 1e-4$ (corresponding to SNR = 20dB). The passive sensor mode works with detection probability $P_d = .5$ and false alarm probability $P_f = 1e-4$ (SNR = 10dB). When in hide mode, both sensor modes are severely degraded and correspond to a target with SNR = 0dB.

Targets can sense when the active mode is used and move into hide mode to prevent further interrogation. Additionally, targets that have moved into hide mode may move back into visible mode when the passive sensor mode is used. The parameters of interest can be summarized by the following transition probabilities when for each of the two sensor modes:

$$\begin{bmatrix} Pr(\textit{visible to visible}) & Pr(\textit{visible to hide}) \\ Pr(\textit{hide to visible}) & Pr(\textit{hide to hide}) \end{bmatrix}$$

A myopic strategy of sensor management makes tasking decisions based only on the expected immediate reward. Here the myopic strategy will advocate using the active mode at all times as it has the largest expected gain in immediate information. Depending on the transition probabilities, this will often immediately force the targets into hide mode, making them difficult to observe in future time steps. A non-myopic strategy, on the other hand, will take into account the effect of current actions on future information gaining ability and be more prudent in using the active mode.

We present two simulations with different values for the transition probabilities.

In the first simulation, we use

$$\text{Transition Matrix Active Sensor Mode} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\text{Transition Matrix Passive Sensor Mode} = \begin{bmatrix} 1 & 0 \\ .2 & .8 \end{bmatrix}$$

which indicates that the target always moves into hide when the active mode is used, and moves from hide to visible with probability .2 when the active mode is used.

In the second simulation we ease the tendency of the target to be in hide mode, specifically we use

$$\text{Transition Matrix Active Sensor Mode} = \begin{bmatrix} .1 & .9 \\ 0 & 1 \end{bmatrix}$$

$$\text{Transition Matrix Passive Sensor Mode} = \begin{bmatrix} 1 & 0 \\ .33 & .67 \end{bmatrix}$$

which indicates that the target has some chance (10%) of staying in visible mode even if the active mode is used, and is more likely to move back into visible mode when the passive mode is used.

We trained a Q-function as discussed in Section 4.2.5 by running 100,000 training episodes for each of the two scenarios. Episodes were generated with random sensor allocations using the models of target behavior. The initial position of the targets and realization of the diffusive motion of the targets was chosen randomly for each training episode. The Q-function was trained using a linear function approximation on the set of state, action, next state, immediate reward 4-tuples generated during training in batch fashion. The algorithm was tested on 1,000 example episodes

where the initial position and realization of the diffusive motion of the targets was chosen randomly for each testing episode. The Q-function learned during the training episode was used to schedule the sensor by selecting mode and pointing direction.

In Figures 4.11 and 4.12, we present results of target localization using the Q-learning strategy detailed in Section 4.2.5 for the two simulations. We compare this performance to (a) a random strategy, (b) a myopic strategy, (c) a random strategy that only uses the passive mode, and (d) a myopic strategy that only uses the passive mode. The Q-learning strategy performs as well or better than the best of the four competing strategies in both cases.

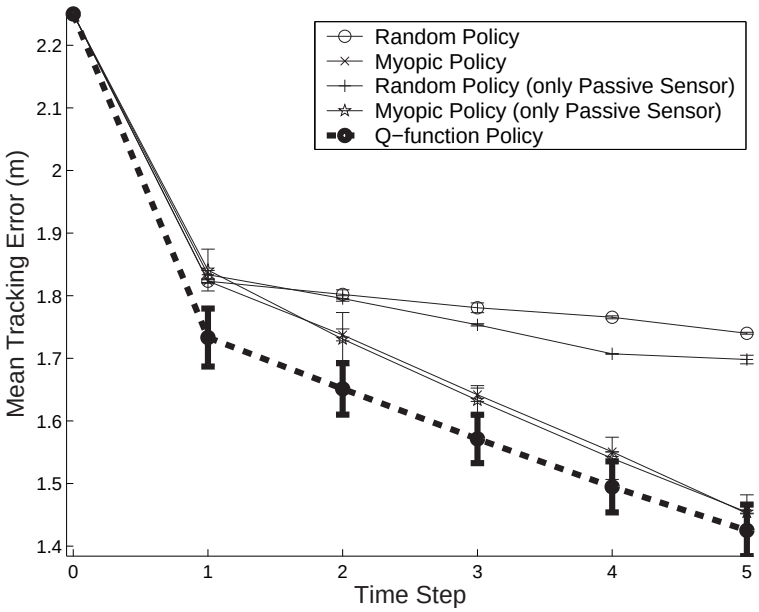


Figure 4.11: Target tracking performance for the first smart target simulation. Included are a random strategy, a myopic strategy, a random strategy that uses only the passive mode, a myopic strategy that uses only the passive mode, and the Q-learning strategy.

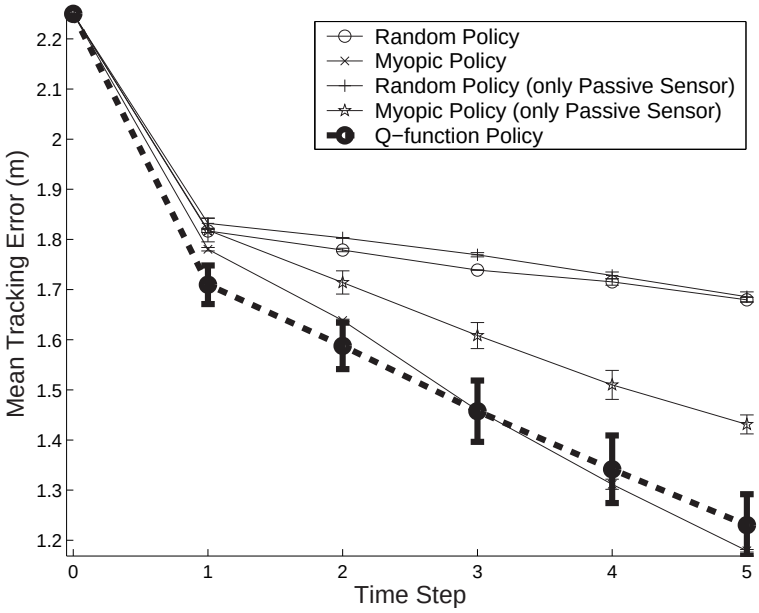


Figure 4.12: Target tracking performance for the second smart target simulation. Included are a random strategy, a myopic strategy, a random strategy that uses only the passive sensor, a myopic strategy that uses only the passive mode, and the Q-learning strategy.

CHAPTER V

Conclusion

This thesis has addressed the problem of scheduling the resources of agile sensors. We investigate an information-based approach, wherein sensor tasking decisions are made based on the principle that actions should be chosen so as to maximize the information expected to be extracted from the scene. The main benefit of this approach is that it provides a single unifying metric that is able to automatically capture the complex tradeoffs involved when choosing between the large number of possible sensor taskings.

The information-based approach to sensor management involves the development of three interrelated elements.

First, we form the joint multitarget probability density (JMPD), which is the fundamental entity capturing knowledge about the number of targets as well as the states of the individual targets (e.g. position, velocity and identification). Unlike more traditional methods, the JMPD does not assume any independence, but instead explicitly models the coupling in uncertainty between target states, between targets, and between target state and the number of targets. Furthermore, we do not assume the JMPD is of some parametric form (e.g. Gaussian, sum of Gaussian, etc.) but rather allow for complete generality in distribution. Because of this gener-

ality, the JMPD is a very high dimensionality object which must be estimated using sophisticated numerical techniques. Our representation of the JMPD is done via a novel multitarget particle filter using adaptive sampling schemes, which provides computational tractability.

Second, we use the estimate of the JMPD to make (myopic) sensor resource allocation decisions. The fundamental paradigm for sensor scheduling is to choose sensing actions which maximize the expected amount of information gained. This unifying metric allows us to automatically trade between sensor allocations that provide different types of information (e.g. actions that provide information about position versus actions that provide information about identification) without any adhoc assumptions as to the relative utility of each.

Finally, we extend the information-based sensor resource allocation paradigm to long-term (non-myopic) sensor scheduling. This extension is computationally challenging due to an exponential growth in action sequences with horizon time. We investigate two approximate methods to address this complexity. First, we look at directly approximating Bellman's equation by replacing the value-to-go function with an easily computed function of the ability to gain information in the future. Second, we apply the reinforcement learning technique called Q-learning as a means of learning a non-myopic policy from a set of example episodes.

There are several interesting directions in which to take this work. The first is that of decentralization, i.e., performing the tracking and sensor management in a distributed network of sensors. This involves determining how, when, and what to communicate between bandwidth and processing power constrained nodes in a network. Communications must be secure, reliable, and minimal. Computations must be light and memory must be used sparingly. This evolution is a necessity for

the transition of this work to a operational platform.

A second area of future work is the exploration of other applications of non-myopic scheduling. Some scenarios not considered in this work include sensors with time latencies, sensors with global constraints (e.g., power), and sensors with mobility. One needs to investigate whether the approximations developed here are viable or need to be extended to accommodate these situations. Furthermore, it needs to be determined if the learning methods can be made computationally tractable for use in a real system.

Other areas of future work include the generalization of the method to include continuous action spaces, applications involving three-dimensional tracking over a topographical domain, simultaneous scheduling of multiple sensors, and target detection where multiple targets may occupy the same resolution cell.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Y. Bar-Shalom, *Multitarget Multisensor Tracking: Advanced Applications.* Artech House, 1990.

[2] M.-S. Lee and Y.-H. Kim, "An efficient multitarget tracking algorithm for car applications," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 2, pp. 397–399, 2003.

[3] H. H. Gonzalez-Banos, C.-Y. Lee, and J.-C. Latombe, "Real-time combinatorial tracking of a target moving unpredictably among obstacles," *Proceedings of the IEEE Conference on Robotics and Automation*, May 2002.

[4] M. Montemerlo, S. Thrun, and W. Whittaker, "Conditional particle filter for simultaneous mobile robot localization and people tracking," *Proccedings of the IEEE Conference on Robotics and Automation*, vol. 1, pp. 695–701, 2002.

[5] M. Isard and A. Blake, "Visual tracking by stochastic propagation of conditional density," *Proceedings of the 4th European Conference on Computer Vision*, pp. 343–356, 1996.

[6] J. Vermaak and A. Blake, "Nonlinear filtering for speaker tracking in noisy and reverberant environments," *Proceedings of International Conference on Acoustics Speech and Signal Processing*, vol. V, pp. 3021–3024, 2001.

[7] D. Tweed and A. Calway, "Tracking multiple animals in wildlife footage," *Proceedings of the Conference on Pattern Recognition*, vol. 2, pp. 24–27, 2002.

[8] K. Chood and D. J. Fleet, "People tracking using hybrid Monte Carlo filtering," *Proceedings of The International Conference on Computer Vision*, vol. 2, pp. 321–328, 2001.

[9] S. Blackman, *Multiple-Target Tracking with Radar Applications.* Artech House, 1986.

[10] P. S. Maybeck, *Stochastic Models, Estimation, and Control, Volume 1.* Academic Press, 1982.

[11] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

[12] A. H. Jazwinski, *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

[13] S. J. Julier and J. K. Uhlman, "A new extension of the Kalman filter to nonlinear systems," *Proceedings of Aerosense: The Eleventh International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, vol. 3068, pp. 182–193, 1997.

[14] D. L. Alspach and H. W. Sorensen, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Transactions on Automatic Control*, vol. 17, no. 4, pp. 439–448, August 1972.

[15] G. Kitagawa, "Non-Gaussian state-space modelling of non-stationary time series," *Journal of the American Statistical Association*, vol. 82, pp. 1032–1063, 1987.

[16] C. M. Kreucher and K. Kastella, "Multiple-model nonlinear filtering for low-signal ground target applications," *Proceedings of The Fifteenth International Aerosense Symposium*, vol. 4380, pp. 1–12, 2001.

[17] N. Bergman, "Recursive Bayesian estimation: Navigation and tracking applications," *Ph.D. dissertation*, 1999.

[18] N. J. Gordon, D. J. Salmond, , and A. F. M. Smith, "A novel approach to non-linear and non-Gaussian Bayesian state estimation," *IEE Proceedings on Radar and Signal Processing*, vol. 140, pp. 107–113, 1993.

[19] S. Musick, J. Greenewald, C. M. Kreucher, and K. Kastella, "Comparison of particle method and finite difference nonlinear filters for low SNR target tracking," *Proceedings of The Fourth Annual Conference on Information Fusion*, 2001.

[20] D. Crisan and A. Doucet, "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736–746, 2002.

[21] A. Doucet, N. de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer Publishing, 2001.

[22] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," *IEEE International Conference on Robotics and Automation*, May 1999.

[23] M. S. Arulampalam and B. Ristic, "Comparison of the particle filter with range parameterized and modified polar ekfs for angle-only tracking," *Proceedings of SPIE*, vol. 4048, pp. 288–299, 2000.

[24] M. Mallick, "Comparison of EKF, UKF, and PF for UGS and GMTI sensors," *The 2003 Defense Science and Technology Workshop*, July 2003.

[25] Y. Bar-Shalom and W. D. Blair, *Multitarget-Multisensor Tracking: Applications and Advances, Volume III*. Artech House, 2000.

[26] L. D. Stone, T. L. Corwin, and C. A. Barlow, *Bayesian Multiple Target Tracking*. Artech House, 1999.

[27] S. Mori, C. Y. Shong, E. Tse, and R. P. Wishner, "Tracking and classifying multiple targets without a prior identification," *IEEE Transactions on Automatic Control*, vol. AC31, no. 5, pp. 401–409, 1986.

[28] M. I. Miller, A. Srivastava, and U. Grenander, "Conditional mean estimation via jump-diffusion processes in multiple target tracking/recognition," *IEEE Transactions on Signal Processing*, vol. 43, no. 11, pp. 2678–2690, 1995.

[29] K. Kastella, "Event averaged maximum likelihood estimation and mean-field theory in multitarget tracking," *IEEE Transactions on Automatic Control*, vol. 50, no. 6, pp. 1070–1073, 1995.

[30] R. P. S. Mahler, "A unified foundation for data fusion," *In the Seventh Joint Service Data Fusion Symposium*, pp. 154–174, 1994.

[31] C. Hue, J.-P. Le Cadre, and P. Perez, "Tracking multiple objects with particle filtering," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 3, pp. 791–812, 2002.

[32] R. Karlsson and F. Gustafsson, "Monte Carlo data association for multiple target tracking," *Proceedings of The IEE Workshop on Target Tracking : Algorithms and Applications*, 2001.

[33] H. A. P. Blom and E. A. Bloem, "Joint IMMPDA particle filter," *Proceedings of the 6th International Conference on Information Fusion*, 2003.

[34] J. Vermaak, S. Goodsill, , and P. Pérez, "Monte Carlo filtering for multi-target tracking and data association," *To appear in IEEE Transactions on Aerospace and Electronic Systems*, 2005.

[35] M. Isard and J. MacCormick, "BraMBLe: A Bayesian multiple-blob tracker," *Proceedings of the 8th International Conference on Computer Vision*, vol. 2, pp. 24–31, 2001.

[36] M. Orton and W. Fitzgerald, "A Bayesian approach to tracking multiple targets using sensor arrays and particle filters," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 216–223, 2002.

[37] S. Maskell, M. Rollason, N. Gordon, and D. Salmond, "Efficient particle filtering for multiple target tracking with application to tracking in structured images," *Procedings of SPIE Conference on Signal and Data Processing of Small Targets*, vol. 4728, pp. 251–262, 2002.

[38] H. Tao, H. S. Sawhney, and R. Kumar, "A sampling algorithm for tracking multiple objects," *Workshop on Vision Algorithms 1999*, pp. 53–68, 1999.

[39] C. Hue, J.-P. Le Cadre, and P. Perez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 309–325, 2002.

[40] D. Schulz, D. Fox, and J. Hightower, "People tracking with anonymous and id-sensors using rao-blackwellised particle filter," *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[41] A. Doucet, B.-N. Vo, C. Andrieu, and M. Davy, "Particle filtering for multi-target tracking and sensor management," *The Fifth International Conference on Information Fusion*, 2002.

[42] S. Musick and R. Malhotra, "Chasing the elusive sensor manager," *Proceedings of NAECON*, pp. 606–613, May 1994.

[43] J. Liu, P. Cheung, L. Guibas, and F. Zhao, "A dual-space approach to tracking and sensor management in wireless sensor networks," *ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[44] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 462–472, August 1990.

[45] R. Popoli, *in Multitarget-Multisensor Tracking: Advanced Applications*, Y. Bar-Shalom, Ed. Artech House, 1992, vol. II.

[46] K. J. Hintz and E. S. McVey, "Multi-process constrained estimation," *IEEE Transactions on Man, Systems, and Cybernetics*, vol. 21, no. 1, pp. 434–442, January/February 1991.

[47] K. J. Hintz, "A measure of the information gain attributable to cueing," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 2, pp. 237–244, 1991.

[48] W. Schmaedeke and K. Kastella, "Event-averaged maximum likelihood estimation and information-based sensor management," *Proceedings of SPIE*, vol. 2232, pp. 91–96, June 1994.

[49] K. Kastella, "Discrimination gain for sensor management in multitarget detection and tracking," *IEEE-SMC and IMACS Multiconference CESA*, vol. 1, pp. 167–172, 1996.

[50] ——, "Discrimination gain to optimize classification," *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, vol. 27, no. 1, January 1997.

[51] R. Mahler, "Global optimal sensor allocation," *Proceedings of the Ninth National Symposium on Sensor Fusion*, vol. 1, pp. 167–172, 1996.

[52] F. Zhao, J. Shin, and J. Reich, "Information-driven dynamic sensor collaboration," *IEEE Signal Processing Magazine*, pp. 61–72, March 2002.

[53] S. Tong and D. Koller, "Active learning for parameter estimation in Bayesian networks," *Proceedings of the Thirteenth Annual Conference on Neural Information Processing Systems*, pp. 647–653, December 2000.

[54] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," *Machine Learning*, vol. 28, pp. 138–168, 1997.

[55] X. S. Zhou and T. S. Huang, "Relevance feedback in image retrieval : A comprehensive review," *ACM Multimedia Journal*, vol. 8, no. 6, pp. 536–544, April 2003.

[56] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos, "The Bayesian image retrieval system, pichunter: Theory, implementation, and psychophysical experiments," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 20–37, January 2000.

[57] D. Geman and B. Jedynak, "An active testing model for tracking roads in sattelite images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 1, pp. 1–13, January 1995.

[58] J. Denzler and C. M. Brown, "Information theoretic sensor data selection for active object recognition and state estimatation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 145–157, February 2002.

[59] M. A. Sipe and D. Casasent, "Feature space trajectory methods for active computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1634 – 1643, December 2002.

[60] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[61] A. Rényi, "On measures of entropy and information," *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics, and Probility*, vol. 1, pp. 547–561, 1961.

[62] V. Krishnamurthy and D. Evans, "Hidden markov model multiarm bandits: A methodology for beam scheduling in multitarget tracking," *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 2893–2908, December 2001.

[63] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden markov model sensors," *IEEE Transactions on Signal Processing*, vol. 50, no. 6, pp. 1382–1397, June 2002.

[64] D. P. Bertsekas and D. Castañon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, no. 1, pp. 89–108, 1999.

[65] D. Castañon, "Approximate dynamic programming for sensor management," *Proceedings of the 1997 IEEE Conference on Decision and Control*, 1997.

[66] ——, "Optimal search strategies for dynamic hypothesis testing," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 25, no. 7, pp. 1130–1138, 1995.

[67] R. Malhotra, "Temporal considerations in sensor management," *Proceedings of the IEEE 1995 National Aerospace and Electronics Conference, NAECON 1995*, vol. 1, pp. 86–93, May 1995.

[68] A. Chhetri, D. Morrell, and A. Papandreou-Suppappola, "The use of particle filtering with the unscented transform to schedule sensors multiple steps ahead," *Procedings of International Conference on Acoustics, Speech, and Signal Processing 2004*, 2004.

[69] J. Shin, L. Guibas, and F. Zhao, "A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks." *Proceedings of 2nd International Workshop on Information Processing in Sensor Networks*, April 2003.

[70] K. Kastella, "Joint multitarget probabilities for detection and tracking," *Proceedings of SPIE Acquisition, Tracking and Pointing XI*, 1997.

[71] R. E. Bethel and G. J. Paras, "A PDF multisensor multitarget tracker,," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 153–168, January 1998.

[72] I. R. Goodman, R. P. S. Mahler, and H. T. Nguyen, *Mathematics of Data Fusion.* Kluwer Academic Publishers, 1997.

[73] S. Musick, K. Kastella, and R. Mahler, "A practical implementation of joint multitarget probabilities," *SPIE Proceedings*, vol. 3374, pp. 26–37, 1998.

[74] K. Kastella, "A maximum likelihood estimator for report-to-track association," *In Proceedings of SPIE*, vol. 1954, pp. 386–393, 1993.

[75] H. M. Shertukde and Y. Bar-Shalom, "Tracking of crossing targets with imaging sensors," *IEEE Transactions on Aeropace and Electronic Systems*, vol. 27, no. 4, pp. 582–592, 1991.

[76] E. W. Kamen, "Multiple target tracking based on symmetric measurement functions," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 371–374, 1992.

[77] C. M. Kreucher, A. O. Hero III, and K. Kastella, "Multiple model particle filtering for multitarget tracking," *Proceedings of the Twelfth Annual Workshop on Adaptive Sensor Array Processing (ASAP)*, March 2004.

[78] G. W. Stimson, *Introduction to Airborne Radar.* SciTech Publishing, 1998.

[79] B. E. Tullsson, "Monopulse tracking of rayleigh targets: A simple approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 27, no. 3, pp. 520–531, 1991.

[80] C. H. Gowda and R. Viswanatha, "Performance of distributed CFAR test under various clutter amplitudes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 4, pp. 1410–1419, 1999.

[81] C. M. Kreucher, K. Kastella, and A. O. Hero III, "Multitarget tracking using a particle filter representation of the joint multitarget probability density," *to appear in IEEE Transactions on Aerospace and Electronic Systems*, 2005.

[82] ——, "Tracking multiple targets using a particle filter representation of the joint mulitarget proabbility density," *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, 2003.

[83] M. Morelande, C. M. Kreucher, and K. Kastella, "Multitarget track initiation using particle filters," *In Preparation*, 2005.

[84] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, June 1999.

[85] T. Hastie, R. Tibshirani, , and J. Friedman, *The Elements of Statistical Learning.* Springer Verlag, 2001.

[86] J. S. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.

[87] P. Tichavsky, C. Muravchik, and A. Nehorai, "Posterior Cramer-Rao bounds for discrete-time nonlinear filtering," *IEEE Transactions on Signal Processing*, vol. 46, no. 5, pp. 1386–1396, May 1998.

[88] B. Ristic and M. S. Arulampalam, "Tracking a manoeuvring target using angle-only measurements: algorithms and performance," *Signal Processing*, vol. 83, pp. 1223–1238, 2003.

[89] A. O. Hero III, B. Ma, O. Michel, and J. Gorman, "Applications of entropic spanning graphs," *IEEE Signal Processing Magazine (Special Issue on Mathematics in Imaging)*, vol. 19, no. 5, pp. 85–95, 2002.

[90] A. Dembo and O. Zeitouni, *Large Deviations Techniques and Applications*. Springer-Verlag, 1998.

[91] R. Blahut, *Principles and Practice of Information Theory*. Addison-Wesley, 1987.

[92] A. O. Hero III, B. Ma, O. Michel, and J. D. Gorman, "Alpha divergence for classification, indexing and retrieval," *Technical Report 328, Comm. and Sig. Proc. Lab. (CSPL), Dept. EECS, University of Michigan*, 2001.

[93] C. M. Kreucher, K. Kastella, and A. O. Hero III, "Sensor management using an active sensing approach," *Signal Processing*, vol. 85, no. 3, pp. 607–624, March 2005.

[94] ——, "Information based sensor management for multitarget tracking," *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, 2003.

[95] K. Kastella and S. Musick, "The search for optimal sensor management," *Proceedings of the SPIE Symposium on Aerospace/Defense Sensing and Controls*, vol. 2759, pp. 318–329, April 1996.

[96] C. M. Kreucher, K. Kastella, and A. O. Hero III, "Efficient methods of non-myopic sensor management for multitarget tracking," *Proceedings of the 2004 IEEE Conference on Decision and Control*, pp. 722–727, December 2004.

[97] G. Tesauro and G. R. Galperin, "On-line policy improvement using Monte Carlo search," *The 1996 Neural Information Processing Systems Conference*, 1996.

[98] M. Kearns, Y. Mansour, and A. Y. Ng, "A sparse sampling algorithm for near-optimal planning in large markov decision processes," *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999.

[99] Y. He and E. K. P. Chong, "Sensor scheduling for target tracking in sensor networks," *Proceedings of the 2004 IEEE Conference on Decision and Control*, pp. 734–748, December 2004.

[100] C. M. Kreucher and A. O. Hero III, "Non-myopic approaches to scheduling agile sensors for multitarget detection, tracking, and identification," *The 2005 IEEE Conference on Acoustics, Speech, and Signal Processing Special Section on Advances in Waveform Agile Sensor Processing*, 2005.

[101] C. Watkins, "Learning from delayed rewards," *Ph.D. Thesis, University of Cambridge*, 1989.

[102] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[103] C. M. Kreucher, D. Blatt, A. O. Hero III, and K. Kastella, "Adaptive multi-modality sensor scheduling for detection and tracking of smart targets," *to appear in Digital Signal Processing*, 2005.