Title
Search and Tracking of an Unknown Number of Targets by a Team of Autonomous Agents Utilizing Time-evolving Partition Classification

Author
Wood, Jared Gregory

Publication Date
2011

Peer reviewed|Thesis/dissertation

Search and Tracking of an Unknown Number of Targets by a Team of Autonomous Agents Utilizing Time-evolving Partition Classification

by

Jared Gregory Wood

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor J. Karl Hedrick, Chair
Professor Francesco Borrelli
Professor Raja Sengupta
Professor Pieter Abbeel

Fall 2011

Search and Tracking of an Unknown Number of Targets by a Team of Autonomous Agents Utilizing Time-evolving Partition Classification

Abstract

Search and Tracking of an Unknown Number of Targets by a Team of Autonomous Agents Utilizing Time-evolving Partition Classification

by

Jared Gregory Wood

Doctor of Philosophy in Engineering – Mechanical Engineering

University of California, Berkeley

Professor J. Karl Hedrick, Chair

The advancement of computing technology has enabled the practical development of intelligent autonomous systems. Intelligent autonomous systems can be used to perform difficult sensing tasks. One such sensing task is to search for and track targets over large geographic areas. Searching for and tracking targets over geographic areas has important applications. These applications include search and rescue, boarder patrol, and reconnaissance. Inherent in applications such as these is the need for high accuracy of observation while performing repetitive, mundane tasks. Additionally, in many of these applications is inherent danger. Fortunately, computers are excellent at performing repetitive, mundane tasks. Also, no loss of life is threatened if intelligent autonomous systems are used for these applications. It is then important to develop computational methods for performing autonomous search and tracking of targets over geographic areas.

In order to perform autonomous search and tracking, intelligent autonomous systems often consist of a team of agents with sensors for making observations. Examples of agents include ground robots, unmanned aerial vehicles, unmanned underwater vehicles, and unmanned spacecraft. The current state-of-the-art approach to accomplish search and tracking decomposes the problem into two parts. The first part is target location estimation. The second part is agent path planning based on the target estimation. As the number of targets increases, the estimation space increases. And when the number of targets is unknown, the dimension of the estimation space is unknown. For the case of general non-parametric estimation, path planning over such a space then becomes a difficult task.

The focus of this dissertation is to develop an alternative approach for autonomous search and tracking. This alternative approach simplifies agent path planning when the number of targets is unknown. This simplification is accomplished in two steps. The first step is to reduce the space over which path planning is performed. This is done by transforming the target location estimation into a target density distribution. The target density distribution is defined over the space of the geographic area. So, the dimension of the target density

distribution is potentially much lower than the dimension of the target location estimation space. The second step is to change the objective of search and tracking. In state-of-the-art search and tracking, the objective is to search for and track targets. When the number of targets is unknown, this objective becomes difficult to define. However, over the geographic area, it is possible to define a finite set of regions that change with time. These regions are defined by analyzing the target density distribution. Given this finite set of regions, the objective is then changed to search for and track these regions. By doing this, the objective is shifted from searching and tracking an unknown number of targets to searching and tracking a known number of regions.

Four types of regions are identified in this dissertation. These types of regions are 1) exploration regions, 2) searching regions, 3) tracking regions, and 4) target nullity regions. Exploration regions are those regions over which little information is available, such as regions with uniform distribution of target density. Searching regions are those regions over which significant information is available to guide a search effort. Tracking regions are those regions over which the information available is very certain, specifying near exact target location. Target nullity regions are those regions that have been searched and in which no targets have been observed. In this dissertation, a method for classifying these regions through time is presented. This method is called time-evolving partition classification. Methods are also presented to optimize agent paths over these types of regions.

The search and tracking approach developed in this dissertation was tested for the case when agents are fixed wing aircraft. This approach was tested thoroughly in simulations. Additionally, several components of this approach were tested in experiments. These results are presented. Based on these results, the approach developed in this dissertation performs better than state-of-the-art approaches. This increase in performance applies to the case when the number of targets is unknown and target estimation is general and non-parametric.

To my family

Through my graduate studies my parents were always there, supporting me and encouraging me to keep on pushing. And while I wrote this dissertation, my wife Mackenzie was always there to cheer me up and keep me balanced. I will always be grateful for my family's love and support.

Contents

List of Figures

Acknowledgments

I want to thank my advisor, Professor Karl Hedrick, for the many years of giving me the opportunity to work on the projects of the Center of Collaborative Control of Unmanned Vehicles (C3UV). In these projects I was able to implement algorithms I had developed and test them in real flight experiments on board autonomous fixed-wing aircraft. All of the flight data presented in this dissertation was produced during flight experiments for the development of the many C3UV projects on which I worked. I also want to thank Professor Raja Sengupta for giving me the opportunity to work on the Scan Eagle project for Shell. In this project many of my algorithms were implemented to successfully automate a Scan Eagle aircraft, which turned out to be an interesting challenge and ultimately a rewarding experience. This project provided much of the flight data presented in this dissertation. Finally I want to thank my fellow lab mates with whom I worked many long, difficult hours in various labs and in the desert, testing and developing hardware and software. Some times we slept, and some times we did not. The flight data presented in this dissertation would not have been obtained had it not been for the hard work and collaboration of my lab mates.

Chapter 1

Introduction

1.1 Background

Search and tracking targets is an activity that has engaged several aspects of society. From intelligence gathering and ground mission support to reconnaissance and search and rescue, the need often arises to be able to search for and track targets of interest whether they be security threats or groups needing rescue and support. Technologies geared toward providing various forms of search and tracking continue to improve. In general, the need is to have greater efficiency and increased coverage of observations in order to have faster and more reliable information. The most challenging type of search and tracking is that in which the number of targets is unknown, the field of view of each observation is much smaller than the total search area, and future observations are dynamically constrained to past observations. In many of the situations involving search and tracking, greater efficiency and increased coverage can be achieved by utilizing a network of intelligent autonomous sensing agents. For example, a team of autonomous aircraft with sensors and intelligence to analyze their observations to autonomously plan their future search paths.

The problem of searching a large area to find and track an unknown number of targets is challenging especially when the field of view of one observation is much smaller than the entire search area. Yet this problem arises in many life scenarios including search and rescue, reconnaissance, and area patrol and surveillance. Each of these scenarios consists of

- A large area in which the existence or location of targets is uncertain.
- The observation field of view of one searcher is small relative to the search area.
- The number of targets is often uncertain.
- The time to conduct search is limited.

A large search area and small observation field of view require intelligent search planning because targets may transition to regions that have already been searched. First inclination

might suggest that previously developed guaranteed, conservative, systematic search procedures [74] should be performed. Yet, the speed of target detection is important since some scenarios consist of establishing safety and others consist of saving lives. Because of the large area, the search must then additionally be biased toward checking in regions with greater likelihood of detecting targets as well as providing a time evolving estimate of regions that can be considered searched. As observed in the search and rescue community, one approach is to maximize the number of skilled searchers involved. In this manner more area can be observed in a single instance and systematic searches can be performed. When the number of targets is known, a search can be terminated once all targets have been found. However, the number of targets is often unknown, as in area patrol, surveillance, and reconnaissance. Uncertainty in the number of targets further complicates a search. One complication is that the entire area must be searched. Another is that there must be a way to determine when the area has been sufficiently searched so that the search can be terminated.

The approach taken in this work toward solving this problem consists of the following realizations.

- There must be a time evolving search estimation that captures regions of possible target existence and regions that can be considered searched.
- There must be multiple skilled searchers involved that cooperate to find the targets.
- The searcher paths must be based on the search estimation.

The standard approach is to estimate the target locations and use this estimate for the search estimation [91]. Searcher paths can then be determined by direct optimization over the target estimate distribution [8, 9, 17]. The work that will be presented builds upon this approach. However, the method presented decomposes the search estimation into 1) target estimation and 2) a time evolving partitioning to account for the type of search that should be performed over each region of the surveillance area. The classification of this partitioning is accomplished by analyzing the target estimation. Additionally, it is assumed that the search is accomplished by a team of skilled, intelligent searching agents. The goal is then to develop autonomous search estimation and cooperative team path planning in order to detect all possible targets and be able to specify when the search area has been fully searched.

1.2 Components of Autonomous Search and Tracking

The type of search planning addressed in this work accounts for search and tracking missions in which the following attributes are present.

1. A single observation field of view is small relative to the entire surveillance area.
2. The placement of observations is dynamically constrained to the previous placement.

3. Multiple searchers are utilized in order to increase the observation coverage.

4. Searchers are autonomous agents who determine an appropriate search or tracking strategy.

5. The number of targets to search and track is unknown.

The purpose of a search and tracking mission is to find all targets that are present in a given surveillance area and then to track the positions of all targets. The problem is then to find an approach that accomplishes this purpose. This purpose is typically accomplished by decomposing this problem into two layers. These layers are

1. Target state estimation

2. Path planning based on target state estimate

For attributes 1 through 4 this decomposition has been shown to work well for scenarios wherein there is either a single target [47, 51, 89, 8] or a known number of multiple targets [8, 11, 10, 92, 5, 9, 16, 17, 44, 100]. In these scenarios, target state estimation consists of computing well defined probability distributions for each target. Consequently, appropriate search or tracking strategies for each target or for any region of the surveillance area can be determined. However, for the scenario in which the number of targets is unknown (attribute 5), there is no longer a well defined estimate for all possibly existing targets, although some of the multi-target methods referenced above provide some heuristic means to accomplish tracking for an unknown number of targets. In any case, there is consequently no immediately apparent strategy for search or tracking for any region in the surveillance area.

1.2.1 Adding Learning to the Problem Decomposition

Because not all targets are known, estimates for all targets are no longer available. The approach taken in this work is then to shift the focus of path planning from being target-based to being region-based. By doing this, strategies for searching or tracking each region of the surveillance area is then well defined. But to accomplish this, the problem is further decomposed as shown in Fig. 1.1 by adding a layer of partition learning. The layers of the problem decomposition are then

1. Target density estimation

2. Target partition learning

3. Path planning based on target partitions

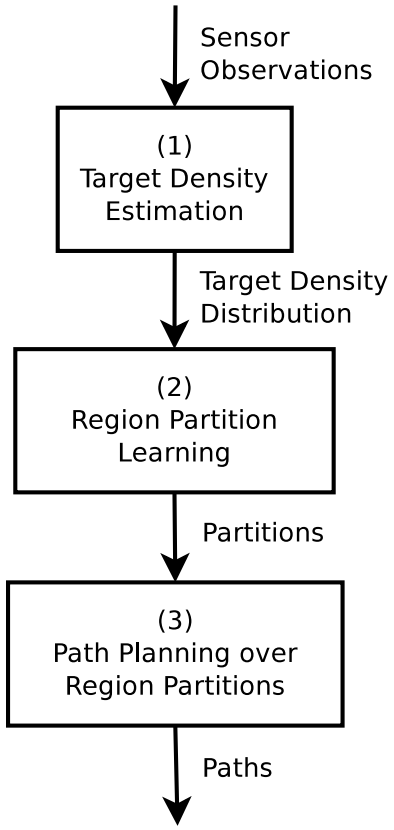


Figure 1.1: Structure of autonomous search and tracking planning by decomposition into target density estimation, region partition learning, and path planning over region partitions.

In the partition learning layer, the surveillance area is partitioned into disjoint regions in which varying levels of search or tracking are the required type of path planning. These partitions, as well as the target density estimation, vary over time to account for updated information collected from observations. Note that the problem decomposition reflects the shift from target-based to region-based path planning because 1) estimation is now an estimation of target density rather than that of estimating each target's state and 2) path planning is now based on the partitions defined over the surveillance area.

1.3 Objectives

Addressing the need for greater efficiency and increased observation coverage to provide faster and more reliable information for search and tracking, this work focuses on the case in which there is an unknown number of targets, the field of view of each observation is much

smaller than the total surveillance area, and future observations are dynamically constrained to past observations. Additionally, the search and tracking is assumed to be performed by a team of autonomous agents who autonomously determine their search paths. To construct a solution, a new structure for autonomous estimation and search planning is presented. This structure decomposes the search and tracking problem into three layers, which are

1. Time-evolving target density estimation
2. Time-evolving partition learning
3. Path planning based on time-evolving partitions

The general objective is to provide methods for accomplishing autonomous search planning according to this decomposition of the search and tracking problem. To accomplish this objective, methods to accomplish each of the layers will be presented.

The first objective is then to present methods for accomplishing time-evolving target density estimation. Target density estimation is not a new topic. Much work has been done by researchers in this field to develop methods for target density estimation. These methods will be presented.

The second objective is to present methods for time-evolving partition learning. To accomplish this, a partition classifier will be presented that operates over a time evolving estimate of target density defined over the surveillance area. To develop this classifier 1) features must be determined that characterize regions in the surveillance area based on the target density distribution, 2) the types of regions must be defined based on the characteristic features, and 3) the structure of the classifier must be determined in order to correctly partition the surveillance area into search, tracking, exploration, and null target regions. These four types of regions will be classified because the type of agent control should be different for each. In search regions, there is some degree of target information however the exact positions of targets is not yet known. So an agent should choose paths that guide it to observe lots of area but that also lead it to locations of highest potential for detecting targets. In tracking regions, the position of targets is well known and an agent should choose paths to aggressively keep targets in observation. In exploring regions, little to no information is known about potential target locations. Consequently, in these regions an agent should choose paths conservatively in order to maximize possible target detections. Null target regions are important for maintaining which regions should not be re-observed because no targets exist in those regions. In this manner search effort is focused on regions of possible target existence.

The third objective is observation and path planning methods based on time-evolving partitions. The development of the planning methods consists of 1) task allocation route planning over the partitions and 2) partition level distribution based path optimization. In order to optimize agent routes over partitions, cost functions are constructed that are easily measured and correctly capture the time varying effort to visit particular partitions. Then,

in order to search, track, or explore a sequence of allocated partitions, an agent must choose paths and points of observation by optimizing over disjoint partition level target density distributions.

Lastly, results will be presented for the overall autonomous search planning methods. Development of the methods presented contribute new technology in order to enable autonomous search and tracking for the challenging case of there being an unknown number of targets to find, in which observation fields of view are much smaller than the surveillance area, and future observations are dynamically constrained to past observations. The research is motivated by the need to have greater efficiency and increased observation coverage to provide faster and more reliable information.

1.4 Related Work

Path planning to search an area using a team of autonomous agents is a subject that several research groups have addressed. A brief review of approaches taken by other groups and how this work contributes new material is presented here.

The major branches of relevant autonomous agent team path planning are

1. Task oriented route planning
2. Probability distribution oriented receding horizon planning

These separate branches of work focus on different aspects of desired team behavior in order to search an area of interest.

1.4.1 Task Oriented Route Planning

The task oriented route planning branch of work focuses on large scale team route optimization and tight control over how and which regions of the search area are searched. Consequently this branch of work breaks team path planning into a finite set of tasks. Each of the tasks represents some self contained element of work for which the required effort of accomplishment is easily measured. A commanding user can then specify exactly how an area should be searched [41, 38]. For example, there may initially be some set of points that need to be observed. The observation of one of the points can then be one task. The set of observation tasks can then be sent to the team of agents. As the agents progress toward observing the points, other tasks can then be input, such as observing some border within the search area. In order to guarantee completion of all tasks, each task must have an easily measured effort or time of completion. Consequently tasks are very strict and agents will only do exactly what has been specified. There is little to no control over the quality of information gained. Returning to the previous example, once a point is observed, that task is accomplished. The agent has no control over whether or not it should observe neighboring

points based on the results of the observation. All information gathering intelligence is left with the commanding user.

The problem of optimally visiting a set of points has been well studied, starting from the case in which there is a single agent, referred to as the Traveling Salesman Problem [65, 6]. The generalization of this problem to multiple agents has also been well studied and is referred to as the Vehicle Routing Problem [30, 97]. The classic Vehicle Routing Problem assumes the set of points to visit are stationary and that agents are fully capable of complete information retrieval throughout their routes. Yet, the classic problem is still computationally complex and requires approximations or heuristics to solve quickly [97]. Recent advancement of mobile robotics and applications involving teams of cooperating robots [22] has brought new researchers to consider developing application specific variations on the Vehicle Routing Problem, such as UAV team cooperation in accomplishing mission objectives [25]. In these new variations, the environment is more complex, information retrieval is far from complete, and the set of points to visit often varies with time. Consequently, current approaches attempt to solve this complex optimization problem by developing either heuristics with rules to directly solve the optimization [1] or approximations to decompose the optimization and solve nicer optimizations with theoretical guarantees [40].

In either of these frameworks, possibly time varying cost functions must be defined that correctly measure the time or effort to visit a particular task. One part of this work is to discuss what types of cost functions should be utilized for the three types of partitions (that were previously defined above), which are

- Searching partitions
- Tracking partitions
- Exploring partitions

Adequate cost function definitions for each of these types of partitions must be based on predicted time to reach particular levels of uncertainty as well as the effort to perform the required actions. Methods for selecting these cost function definitions have been provided in the literature. They will be discussed later.

1.4.2 Probability Distribution Oriented Receding Horizon Planning

Quality of information is the exact focus of probability distribution oriented receding horizon methods. These methods are inherently probabilistic and do not attempt to break up team path planning into tasks. Instead, a commanding user inputs a probability distribution over the search area that biases which points in the area are of high importance. This provides initial guidance on where an agent should go as it searches the area. However, there is no strict control over which points are observed through time. Instead, the agents are given

the intelligence to detect features of importance so that the agents can modify the search on their own as they observe points in the search area. The team's paths are then chosen to maximize the quality of information that is obtained.

These approaches directly optimize over the estimated target probability distribution. Consequently estimation and path planning are coupled. So, to describe the different approaches, the type of target estimation will first be discussed and then the corresponding types of path planning will be presented.

Estimation

Estimation for classic target tracking applications is well established [8]. These types of estimation are well characterized by Kalman filtering, including various extensions such as Sigma Point filtering. Extension of these types of estimation to multiple sensors and multiple targets is also well established [9, 17]. Yet these methods rely on high frequency target detections as is typical in radar applications because the entire search area can be scanned quickly. In applications where the observation field of view is much smaller than the entire search area, this high frequency of target detections disappears. Instead, most observations will be that no target has been detected, even though there may be several targets within the search area. It has consequently been established that likelihood functions based on no-detection observations are required in these types of applications. The classic types of target tracking are then no longer applicable [91]. For these types of applications, estimation algorithms rely directly on numerical implementation of the definition of recursive Bayesian Estimation which consists of Bayesian updates (normalization with observation likelihood) and time prediction (applying the law of total probability to the future). Extensions of this type of estimation are also well established for the case of multiple targets and multiple sensors [92]. These extensions generally require the implementation of a track table to account for a running estimate of target association and to address the possible hypotheses for which detection belongs to which target. However, for these types of estimation, target tracking for a large number of targets becomes very computationally demanding. To further complicate matters, when the number of targets is unknown, the estimation problem ceases to be feasible without implementation of heuristics [70]. This infeasiblity, yet necessity, to have a theoretically guaranteed method for estimating the location of an unknown number of targets with small observation fields of view led to the development of methods for estimating target density [70]. This type of estimation preserves the definition of recursive Bayesian estimation but extends it to account for the estimation of random target sets instead of variables.

This work does not attempt to contribute a new approach for target estimation. Instead, this work builds upon the estimation of a target density distribution [70]. To do this, the density distribution is analyzed to generate time evolving disjoint partitions within the search area. These partitions are determined by a classifier that runs through each step in time. Consequently, target estimation is extended to include this layer of time evolving partition

classification. Each partition then specifies a particular type of required search, tracking, or exploration in order to detect targets. Portions of this work have been presented by the author already [104, 103]. In this dissertation, this work will be presented with more detail as well as more developments and results.

Estimation Dependent Path Planning

Probability distribution oriented receding horizon planning methods can be discussed considering three types of search and tracking, as determined by the types of applicable target estimation. The types of search are

1. There is only one target
2. There is a known number of multiple targets
3. There is an unknown number of targets

For each search type, some applicable existing path planning approaches will be mentioned.

For the first case, a single probability distribution can be used to estimate the target position, so implementations of standard recursive Bayesian estimation work well. Consequently, since there's a single probability distribution, paths can be optimized directly over it [21, 60]. And there are several information metrics that can be used for the path optimization, such as probability of detection [21], entropy gain [68], mutual information [43, 48], Kullback-Leibler divergence [27], and other generalizations of information [28, 78]. Many path planning strategies have been developed for this case. The effectiveness of the search can be improved by utilizing a team of cooperating agents [94, 18, 102, 37, 43, 48, 81]. To achieve some level of cooperation there are several approaches. One approach is to have each agent fuse observations from all other agents and choose a path that optimizes over the probability distribution [18, 43, 48]. This approach is termed *coordinated* because the agents do not directly consider the paths of other agents. Another approach is to additionally have each agent receive the predicted paths of all other agents and perform a decentralized optimization for the best path [19, 36]. This approach is termed *cooperative*. Because of the vast amount of communication and computation required in fully cooperative approaches, alternative approximately cooperative approaches have been developed [94, 73]. The path an agent chooses to follow for each of these methods ultimately is determined by optimization over a probability distribution of the predicted target location. At this point typically either single or multiple-step finite horizon path optimization determines the path for the agent to follow. However, there are performance limitations with finite-horizon optimization. Ways to get around these limitations are proposed in [105, 94].

For the second case, variations on Bayesian estimation have been developed, such as multiple hypothesis tracking [92]. The most naive approach is to maintain multiple independent probability distributions (one for each target) and assume that the target association

of observations is given. However, the target association must be addressed in reality and methods for this case cease to be purely Bayesian estimation because they maintain track tables and require some form of target association [92, 70]. Yet in most of these methods a set of distributions are constructed (one for each target) and the path planning optimization strategies developed for the first case can be extended to account for optimization over multiple probability distributions [102, 37].

For the third case, the complexity of maintaining a probability distribution for each target (including unknown targets) becomes difficult. This difficulty inspired development of a generalization of the standard recursive Bayesian estimation framework from random variables to random sets that provides a target density distribution over the search area instead of multiple probability distributions [70]. With this estimation method, there's one distribution over the search space that combines all targets into an estimate of target density. Consequently, the path planning strategies developed for the first two cases are no longer applicable. For this case, path planning strategies have been developed that choose paths by maximizing the expected number of targets in the search area [69]. However, analysis of the target density distribution can determine regions within the search area that correspond to 1) regions of possible target existence, 2) regions of almost certain target existence, 3) regions that require more searching to gain information, and 4) regions of almost certainly no targets. The search area can then be partitioned into disjoint subsets characteristic of these regions.

This work also addresses the third case of estimation dependent path planning, in which there is an unknown number of targets. However, instead of attempting to develop another method that sticks purely to probability distribution oriented receding horizon optimization, the approach of this work develops a planning structure in which agent route planning and probability distribution path optimization are combined. This approach requires new methods for partitioning the search area over time into regions characteristic of search, tracking, and exploration. The partitions are then allocated to the agents and observation and agent paths are planned by optimizing over disjoint sets of partition level target density distributions. To do this, formal feature based definitions of search, tracking, and exploration partitions must be constructed and cost functions must be developed that represent the effort to visit search, tracking, and exploration partitions. The methods for optimizing over sets of distributions must also be developed. This work is novel and contributes to the field of autonomous agent path planning, particularly by addressing the case of when the number of targets is unknown and the field of view of observations is small relative to the search area.

1.5 Contributions of this Work

As will be detailed later, the contribution of this work is an approach toward solving the search and tracking path planning problem for the case when the number of targets is unknown. This approach has shown to find targets and more efficiently search regions of

the surveillance area in a shorter amount of time than state-of-the-art methods. The steps taken to accomplish this can be summarized as

1. Shift of search and tracking from being target based to being region based.
2. Decomposition of search and tracking into three components.
3. Learning over the target density distribution.
4. Time-evolving partition classification.
5. Path planning strategies over partitions.

The typical approach toward search and tracking is to generate estimates of each target detected and then to plan agent paths based on the target estimates. However, in reality information for some targets will be very uncertain, no information will exist for other targets, and the number of targets will not be known. The first contribution of this work is then to shift the focus of search and tracking away from the targets directly. The focus is then turned toward the types of regions that exist in the surveillance area. Search and tracking is then performed by considering the types of regions that exist and what type of path planning should be done for each region. Consequently, search and tracking is accomplished by defining regions of characteristic information and then searching over these various regions to gather information in ways tailored to the types of existing regions.

Search and tracking is typically decomposed into two components. These components are 1) target track estimation and 2) agent path planning based on the target track estimation. Although the target track estimation is often further decomposed into detection and tracking components, and similarly path planning is often further decomposed, the estimation and path planning still stand as the only two components. This two level decomposition is typical because the focus of search and tracking is typically directly on the target estimates. Considering that in this work the focus is on the types of regions that exist in the surveillance area, the second contribution of this work is the decomposition of search and tracking into three components. Three components are needed to account for the step of defining regions within the surveillance area. This is done to better address the aspects of search and tracking for an unknown number of targets by a team of autonomous agents with sensors that have small fields of view relative to the surveillance area. The three components of this composition are

1. Target density estimation.
2. Surveillance area partition learning.
3. Path planning over surveillance area partitions.

The first component is to estimate the target density distribution. This distribution, as will be detailed later, essentially combines an estimate of the exploration map with all target estimates to form an estimate of possible target density within regions of the surveillance area. The second component is surveillance area partition learning based on the target density distribution. This component is the step of search and tracking in which characteristic regions within the surveillance area are defined. This is the main focus of this work. The third component is agent path planning over the surveillance area partitions that were defined in the second component. Given that the surveillance area is partitioned, these partitions can be viewed as tasks to allocate to the team of agents. As such, in this work the path planning is further decomposed into 1) agent routing over partitions and 2) partition level path optimization based of partition level target density distributions.

As mentioned above, partition learning is the second component of search and tracking decomposition as presented in this work. The third contribution of this work is then methods for learning these partitions at each moment in time. This learning is referred to as time-evolving partition classification because the type of learning employed in this work is that of an unsupervised cascade of classifiers. This contribution is the main innovation presented in this work. All other contributions revolve around this.

The final theoretical contribution of this work is methods for planning agent paths over the time-evolving partitions. As mentioned above, path planning is decomposed into 1) agent routing over partitions and 2) partition level path optimization based on partition level target density distributions. In this work it is assumed that the agent routing step of path planning is given. Indeed, much development has been done to establish agent routing algorithms. Consequently, all focus of path planning was placed on partition level path optimization. Methods for accomplishing this level of path planning will be presented later.

In addition to the contribution specified above, the approach presented in this work has also been implemented and its performance analyzed. As will be shown later, the results of the performance analysis suggest that the methods developed in this work perform well to enable search and tracking for an unknown number of targets by a team of autonomous agents with sensors that have small fields of view relative to the surveillance area.

1.6 Summary

In this work is presented an approach for accomplishing general autonomous search and tracking missions consisting of the attributes

1. A single observation field of view is small relative to the entire surveillance area.
2. The placement of observations is dynamically constrained to the previous placement.
3. Multiple searchers are utilized in order to increase the observation coverage.

4. Searchers are autonomous agents who determine an appropriate search or tracking strategy.

5. The number of targets to search and track is unknown.

To accomplish this, the problem is decomposed into three layers as shown in Fig. 1.1. The layers are

1. Time-evolving target density estimation

2. Time-evolving partition learning

3. Path planning based on time-evolving partitions

Each of these layers will be addressed in depth. Performance results will then be presented.

Chapter 2

Target Density Estimation

2.1 Overview

Target estimation is the first layer of the problem decomposition, as presented in Fig. 1.1 block (1). Target estimation is accomplished by estimating the target density distribution. Although the notion of target density estimation may not be new to this work, it is required in the search and tracking partition classification presented later. As will be discussed below, it is not necessary to directly estimate target density because it can be computed from any number of other target tracking estimation methods. However, in some scenarios, direct estimation of target density may actually be preferable. Yet, the exact estimation method implemented in a search and tracking problem depends on what assumptions can be made by the possible targets within the surveillance area. The purpose of this chapter is to review target track estimation, highlighting the types of estimation that should be considered for various types of target scenarios. The type of estimation to perform will be based on assumptions that can be made regarding possible target cases, such as whether it can be assumed that there is only one target, or multiple. Details of how to fuse observation information to target estimation will be presented when sensor observations are performed by a visual spectrum camera mounted on a gimbal on board an aircraft. In order to present target tracking estimation, including the target density distribution, the topics discussed in this chapter will be presented in the following order.

1. General estimation
    (a) Bayesian estimation
    (b) Recursive Bayesian estimation
2. Target tracking
    (a) Single target

(b) Known number of multiple targets

(c) Unknown number of targets

i. Target density distribution

ii. Random finite sets

iii. Multitarget calculus

iv. Recursive Bayesian estimation for random finite sets

3. Summary

2.2 Estimation

The problem of estimation deals with how to infer the true values of numbers by making observations directly or indirectly of those numbers. The estimation problem then incorporates

- Designing the estimation structure and observation process.
- Determining a mathematical procedure for estimating the true values of the numbers.

The design of the observation process depends on the nature of the estimation problem. In the simplest case, the number to estimate is directly observable. However, there is always error in observation. So, for this case the observation process design simply requires modeling of this observation error. For example, given that θ is the number to estimate, it may be determined that this observation error can be adequately modeled by a Gaussian distribution $N(\theta, \sigma)$ about the true value θ. Then each observation is a sample $Z \sim N(\theta, \sigma)$. With the assumption that observations are independent of each other and all have the same distribution (independent and identically distributed (IID)), this fully specifies the estimation structure and observation process for this simple case. The next step is then to determine mathematical procedure for estimating the true value of the number θ. One option is to select an estimator to infer the true value of θ. For this case, such an estimator could be chosen to be the sample mean defined by

$$\hat{\theta} = \frac{1}{N} \sum_{i=1:N} z_i. \tag{2.1}$$

Notice that this example emphasizes that θ is not random and it's value is estimated by inputting observation samples into an estimator. This type of estimation process is characteristic of Frequentist methods which is a field of statistical inference that emphasizes selection of estimators purely on data [57, 67]. Notice however that the selection of the estimator chosen above may have seemed natural yet arbitrary. To counter this, Frequentist methods attempt to determine optimal estimators by comparing estimator performance through the

expected value of some selected loss function. The expected value of an estimator's loss function is known as the estimator's risk. Estimators can then be selected to yield minimum risk.

Although Frequentist methods of statistical inference are very useful in many contexts, the estimation problems encountered in this work are better treated by utilizing methods known as Bayesian statistical inference [13, 79, 98].

2.2.1 Bayesian Estimation

Bayesian statistical estimation emphasizes the fact that the true value of the number θ appears random to an observer, although its value is not really random. Bayesian methods accordingly attempt to model this apparent randomness by determining a probability distribution over θ, given observations z. This distribution is then $P(\theta|z)$. To obtain this distribution, Bayes' Theorem is utilized, which is defined as

$$P(\theta|z) = \frac{P(z|\theta)P(\theta)}{P(z)}. \tag{2.2}$$

Recalling the Law of Total Probability, $P(z)$ becomes

$$P(z) = \int_{\theta\in\Theta} P(z|\theta)P(\theta)d\mu(\theta), \tag{2.3}$$

and Bayes' Theorem can then be rewritten as

$$P(\theta|z) = \frac{P(z|\theta)P(\theta)}{\int_{\theta\in\Theta} P(z|\theta)P(\theta)d\mu(\theta)}. \tag{2.4}$$

Furthermore, note that the conditional probability $P(z|\theta)$ simply expresses the likelihood of the observation z given a value of the unknown θ. Because of the normalization, this likelihood need not be a probability and can more generally be written as $L(\theta; z)$. The likelihood is often written in this form to stress that it need not be a probability and to show that the independent variable of the function is θ, not z (since z is the observation and θ is allowed to vary). Bayes' Theorem can then be modified to be

$$P(\theta|z) = \frac{L(\theta; z)P(\theta)}{\int_{\theta\in\Theta} L(\theta; z)P(\theta)d\mu(\theta)}. \tag{2.5}$$

Under the Bayesian estimation paradigm the example presented above can be represented by the graphical model in Fig. 2.1. This graph represents the total probability

$$P(\theta, Z) = P(\theta) \prod_{i=1:N} P(Z_i|\theta). \tag{2.6}$$

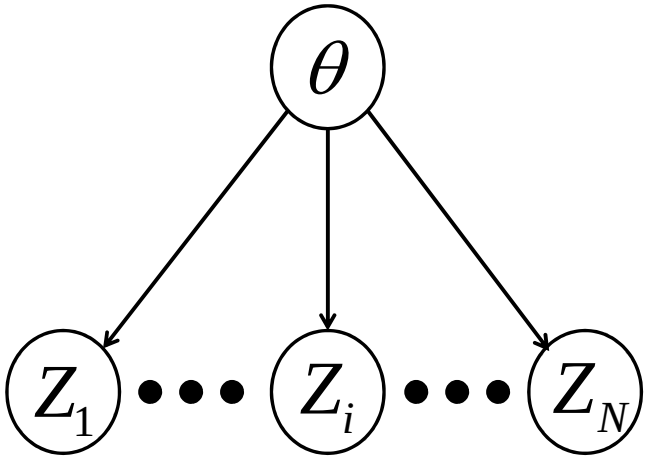


Figure 2.1: Total probability of the unknown number θ and observations Z_i

After performing the observations $(Z_i = z_i, ..., Z_N = z_N)$, the distribution over θ conditioned on these observations is then

$$\begin{aligned} P(\theta|z_{1:N}) &= \frac{P(\theta, z_{1:N})}{P(z_{1:N})} \\ &= \frac{P(\theta)\prod_{i=1:N} P(z_i|\theta)}{P(z_{1:N})}. \end{aligned} \tag{2.7}$$

Notice that the likelihood function for this example is $L(\theta; z_{1:N}) = \prod_{i=1:N} P(z_i|\theta)$. In the Frequentist method it was required to determine an estimator and loss function to determine its performance. Here it can been seen that for Bayesian methods it is required to provide some prior distribution over the number θ as well as to perform a possibly computationally heavy integration for the normalization.

2.2.2 Recursive Bayesian Estimation

Typically the estimation structure is more complicated than the simple estimation problem presented above in Fig. 2.1. For target tracking purposes, the number to estimate is generally time varying since a target tends to move. Within the estimation structure this tendency of the target to move should be modeled in order to provide a more accurate estimation. To do this, relabel the number to estimate as a time varying vector X_t. X is used to highlight the case of target tracking in which this vector consists of at least the position of the target. Then the graphical model of the total probability becomes that presented in Fig. 2.2.

The total probability represented in Fig. 2.2 is

$$P(X_{1:T}, Z_{1:T}) = P(X_0) \prod_{t=1:T} P(X_t|X_{t-1})P(Z_t|X_t). \tag{2.8}$$

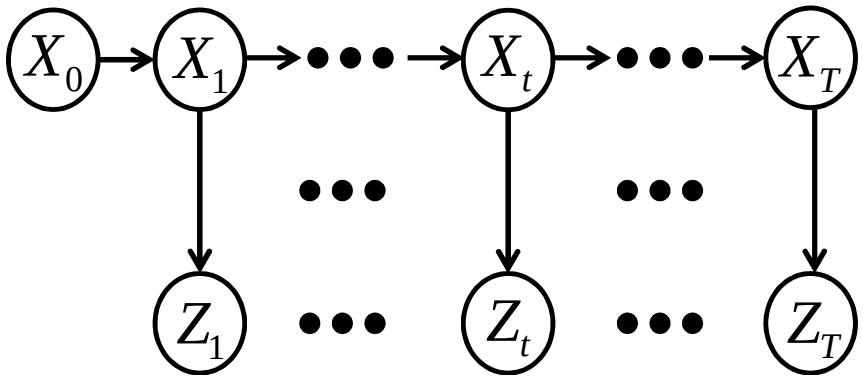


Figure 2.2: Total probability of the unknown time varying position vector X_t and observations Z_t

Consider T to be the current time of the estimation algorithm and the set of observations $(Z_1, ..., Z_T) = (z_1, ..., z_T)$ to be given. The desired distribution at time T is then $P(X_T|z_{1:T})$. To get this distribution, notice that it can be written as

$$\begin{aligned} P(X_T|z_{1:T}) &= \frac{P(X_T|z_{1:T-1})P(z_T|X_T)}{P(z_T|z_{1:T-1})} \\ &= \frac{P(X_T|z_{1:T-1})L(X_T; z_T)}{P(z_T|z_{1:T-1})}, \end{aligned} \tag{2.9}$$

where $P(X_T|z_{1:T-1})$ is the prediction of X_T given all past observations $z_{1:T-1}$ and serves as the prior distribution at time T, and $L(X_T; z_T) = P(z_T|X_T)$ is the likelihood function. Note that the prior distribution requires computing a prediction step. This prediction step is accomplished by utilizing the state transition probability $P(X_T|X_{T-1})$, which is the probability of the target movement. Utilizing this transition probability the prediction is accomplished by

$$P(X_T|z_{1:T-1}) = \int_{x_{T-1} \in S} P(x_{T-1}|z_{1:T-1})P(X_T|x_{T-1})d\mu(x_{T-1}), \tag{2.10}$$

which is implementation of the Law of Total Probability. Notice that in Eq. 2.10 the posterior distribution at the past time is utilized to perform the prediction. Phrasing the time varying estimation problem according to Eq. 2.9 is known as recursive Bayesian estimation because the estimation is performed recursively in time between successive identical prediction and update steps. Recursive Bayesian estimation can then be summarized as a two step process consisting of

1. Prediction from past posterior.

2. New observation update from prediction.

These steps are then performed by

$$P(X_T|z_{1:T-1}) = \int_{x_{T-1}\in S} P(x_{T-1}|z_{1:T-1})P(X_T|x_{T-1})d\mu(x_{T-1}),$$
$$P(X_T|z_{1:T}) = \frac{P(X_T|z_{1:T-1})L(X_T;z_T)}{P(z_T|z_{1:T-1})}, \tag{2.11}$$

where the normalization term is computed by

$$P(z_T|z_{1:T-1}) = \int_{x_T\in S} P(x_T|z_{1:T-1})L(x_T;z_T)d\mu(x_T). \tag{2.12}$$

2.3 Target Tracking

Because target tracking problems are generally based on estimating the time varying track of targets, target tracking methods generally utilize recursive Bayesian estimation. Estimation for target tracking is principally complicated by the available assumptions that can be placed on the type and number of targets. Estimation for target tracking then falls into one of three categories. These three categories are

- Single target
- Known number of multiple targets
- Unknown number of targets

2.3.1 Single Target

Estimation for target tracking when it is known that there is only one target within the surveillance area is well developed because it fits perfectly within the framework of recursive Bayesian estimation of random variables or vectors [47, 51, 89, 8]. For single target estimation Fig. 2.2 perfectly specifies the structure of the estimation problem and the target state estimation can be solved by Eq. 2.11.

Although the estimation structure is well defined and the estimation problem can be readily posed as recursive Bayesian estimation, the estimation problem is far from solved. Two key design choices must be made. From Eq. 2.11 these two design choices are apparent. These two choices are the design of the

- State transition probability $P(X_T|X_{T-1})$.
- Observation likelihood function $L(X_T;z_T)$.

If these two design choices are not well determined, the estimation may either completely fail or just not perform well. As an example of a poorly defined likelihood function, consider observations the author made during autonomous UAV flight experiments of a ScanEagle aircraft at Camp Roberts, California in the fall of 2010 while working as part of C3UV [83]. Observations were performed by a visual spectrum gimballed camera mounted on the ScanEagle aircraft. At that point in time autonomy computations had to be performed on the ground due to lack of payload space and inability to interface directly with the on board autopilot and electronics. Consequently all processing had to be performed on ground and all information had to be passed via wireless transmission between the ground station and the aircraft. Before doing any flight experiments, the accuracy of computer vision algorithms was tested on the ground. Through these tests it was observed that the computer vision was very accurate. So the observation likelihood function was designed to incorporate this high certainty of observations. However, after performing several flight experiments, extremely jittery target localizations were observed. With each target detection the estimation algorithm would output the estimated target position with high certainty. Yet the peak of the estimate varied significantly and was rarely close to the true target location. The first consideration for the cause of this error was time synchronization between the camera video feed and the aircraft telemetry packets. In fact, lack of synchronization did produce mild error. However, even after synchronizing of the video feed with the telemetry packets this error was still observed after flight experiments. Eventually the frequency of the telemetry packets was measured. These measurements showed that the telemetry packets provided new aircraft and gimbal state and orientation data only at about 2 Hz, which was extremely slow for autonomy algorithms of fixed wing aircraft to operate well. The magnitude of gimbal orientation change within half a second was then tested. These tests showed that within half a second the gimbal could change orientation at a maximum of about 17°. So observations could occur with an error of 17°! So although the computer vision target detections within a frame were very accurate, this accuracy was drastically deteriorated by the possible gimbal orientation error. In order to improve the estimation algorithm, this error had to be incorporated into the observation likelihood function.

Likelihood Function Design

Most design work of the estimation structure typically goes to designing the likelihood function. After all, the likelihood function is the means whereby information enters the estimation algorithm. Some algorithms go as far as to neglect all together the transition probability and focus completely on the collection of likelihood functions from sensor observations.

In the simplest of cases, a sensor observation can be modeled by a Gaussian distribution. And often, even when an observation is not technically Gaussian, observations are modeled by Gaussians so as to preserve Gaussianity of the posterior distribution that is being estimated. Preservation of Gaussianity is done to enable analytical algorithms based on methods similar to the Kalman Filter. For example, it may occur that an observation of a target is made

directly in the space of the target. It may further occur that this observation is straight forward and only has error in location due to uncertainty in the observer's position. Then the observation can be modeled by a Gaussian likelihood function with some variance to account for noise in observation.

However, in this work all sensor observations are produced by a visual spectrum camera on board an aircraft. Consequently all observations occur within the image plane of the camera rather than within the target position space. Furthermore, uncertainty in aircraft position and orientation, as well as camera gimbal mount orientation relative to the aircraft must be projected through the image plane observation. The resulting likelihood function is highly nonlinear and necessarily non-Gaussian.

The process of designing a sensor observation likelihood function will be shown by presenting the design of likelihood functions for observations made by visual spectrum cameras mounted on aircraft. Likelihood functions for the case of a fixed camera on board an aircraft have been developed in [58, 95]. The likelihood functions developed here follow the development presented in [58, 95] but expand it to model likelihood functions for gimballed cameras on board aircraft.

The first note to make for likelihood functions generated from sensor observations made on board aircraft is that most observations will necessarily be observations in which *no* target is detected. This is because, given a surveillance area, until a target is within visibility, a observations will not contain target detections. Yet in order to still update the target estimate, these no-detection observations must be utilized in some form of likelihood function. This means that there are two types of observations and two corresponding types of likelihood functions. For the case in which a target is detected, the observation consists of an instantiation of the set $Z = \{D, U\}$, where D is a binary random variable specifying whether or not a target is detected (D for a detection and $\neg D$ for when no target is detected), and U is a random vector specifying the image coordinates of a detected target. These two observations and likelihood functions are then

1. Target detected: $P(u|X, \hat{T}, \hat{R})$.
2. No target detected: $P(\neg D|X, \hat{T}, \hat{R})$.

where X is the position of the target and $\hat{T}$ and $\hat{R}$ are estimates of the true camera position and orientation (T and R). $\hat{T}$ and $\hat{R}$ define distributions over T and R. The meaning of these estimates can be understood through a simple example. In this example, there is a random variable Y that is assumed to be normally distributed with mean μ and variance σ. An estimator $\hat{\mu}$ of μ is then designed with which the definition of some distribution $P(\mu|\hat{\mu})$ is determined. So, with μ being a parameter of the distribution over Y, $\hat{\mu}$ is then a hyper-parameter. In a similar manner, this work assumes that, given estimates $\hat{T}$ and $\hat{R}$, there can be determined accompanying distributions $P(T|\hat{T})$ and $P(R|\hat{R})$. Along this line of thought, T and R can be viewed as being parameters of the likelihood functions, and $\hat{T}$ and $\hat{R}$ can be viewed as being hyper-parameters. Alternatively the likelihood functions could

be conditional on the true camera position and orientation, rather than on estimates of the camera position and orientation. In deed, the design of likelihood functions conditional on the true camera position and orientation form part of the likelihood function development below. So these forms of the likelihood functions will be developed, and could potentially be used directly. However, if likelihood functions conditional on the true camera position and orientation are used directly, then the estimation algorithm has to estimate the camera position and orientation in addition to the target position. Here it is assumed that estimates of the camera position and orientation are provided by an estimation algorithm *separate* from the target position estimation algorithm. For example, this would be the case in a system where an autopilot provides the camera position and orientation estimate. The camera position and orientation estimation error is then accounted for in the likelihood functions developed here, as will be apparent below. Additionally note that the target detected likelihood function is actually an un-normalized probability density function over X, however the no-detection likelihood function is actually a probability function. For simplicity in formulas, they are both represented as $P(\cdot)$. The reason for the no-detection likelihood function being a probability function and *not* a probability density function will become clear later when this likelihood function is developed. According to these two types of observations and likelihood functions, the update step in recursive Bayesian estimation becomes

1. Target detected, $z_T = \{D, u\}$: $P(X_T|z_{1:T}) \propto P(X_T|z_{1:T-1})P(u|X_T, \hat{T}, \hat{R})$.
2. No target detected, $z_T = \{\neg D, \emptyset\}$: $P(X_T|z_{1:T}) \propto P(X_T|z_{1:T-1})P(\neg D|X_T, \hat{T}, \hat{R})$.

Fig. 2.3 shows the relation between the target and camera. Fig. 2.4 shows the camera's perspective of points in the world [35, 86]. By standard notation, the camera coordinate frame is taken to have z pointing from the camera in line with the camera's direction of view. x and y are then arbitrary but parallel to the camera's image plane. x and y are then taken to be up and right relative to the camera. The camera's image plane is taken to be the plane defined by $z = 1$ in the camera coordinate frame. As represented in Fig. 2.4, a point in the camera's image plane can then be described in camera frame coordinates as

$$u = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.13}$$

Recall that R is the rotation matrix representing the orientation of the camera frame relative to the world frame. This rotation matrix actually consists of several sub-rotations. Decomposing the rotation matrix into these sub-rotation matrices, the rotation matrix can be written as

$$R = R_{GB\to C} R_{G\to GB} R_{B\to G} R_{N\to B} R_{W\to N}, \tag{2.14}$$

where $R_{W\to N}$ is the rotation from the world coordinate frame to the navigation coordinate frame, $R_{N\to B}$ is the rotation from the navigation coordinate frame to the aircraft body

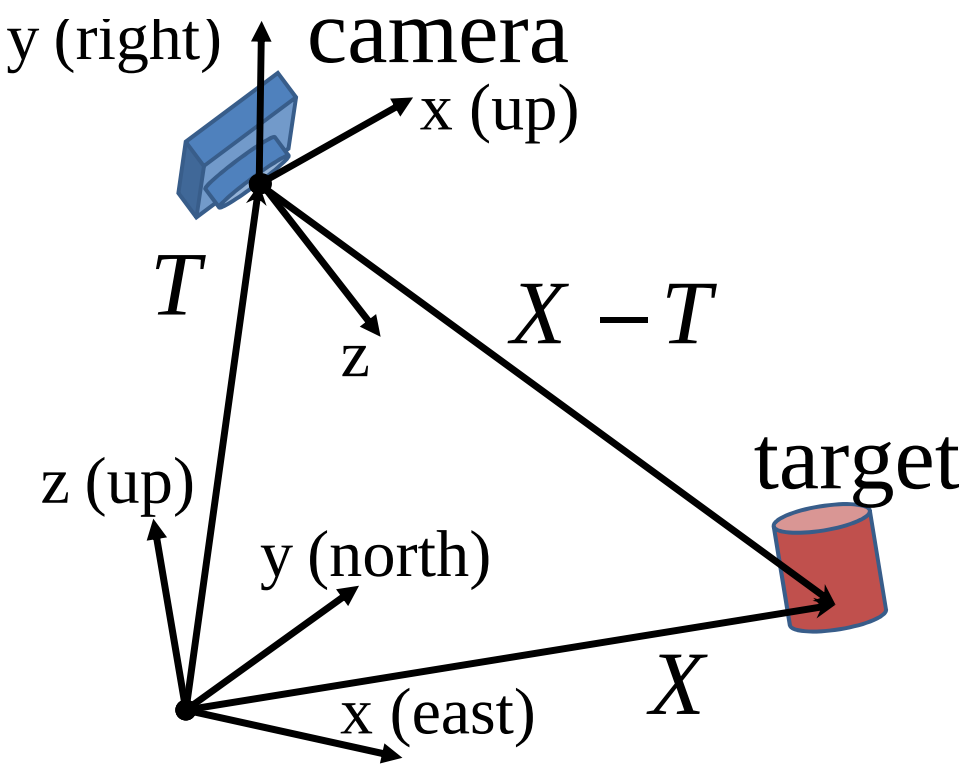


Figure 2.3: Geometric relation between a camera on board an aircraft and a target on the ground, assuming a flat earth model.

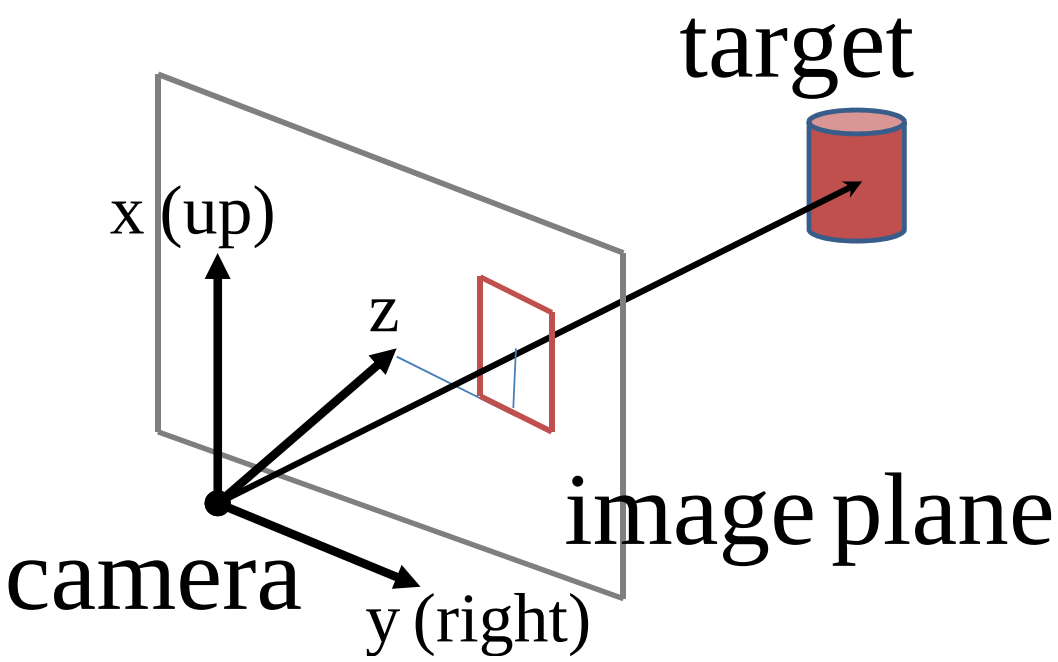


Figure 2.4: Perspective projection relation between a point in the camera image and the corresponding point in the world.

coordinate frame, $R_{B\to G}$ is the rotation from the aircraft body coordinate frame to the gimbal coordinate frame, $R_{G\to GB}$ is the rotation from the gimbal coordinate frame to the gimbal body coordinate frame, and $R_{GB\to C}$ is the rotation from the gimbal body coordinate frame to the camera coordinate frame. In this work the world coordinates are taken to be (east, north, up), navigation coordinates are taken to be (north, east, down), aircraft body coordinates are taken to be (forward, rightward, downward from belly), gimbal coordinates are taken to be (upward from top, rightward, forward), gimbal body coordinates are taken to be the same as gimbal coordinates expect panned and tilted, and camera coordinates were defined above previously. The standard taken in this work is that all rotations are performed on coordinate frames rather than on points in coordinate frames. Consequently the rotation matrices are defined as

$$R_{W\to N} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{2.15a}$$

$$R_{N\to B} = R_x(\phi)R_y(\theta)R_z(\psi), \tag{2.15b}$$

$$R_{B\to G} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{2.15c}$$

$$R_{G\to GB} = R_y(\beta)R_x(\alpha), \tag{2.15d}$$

$$R_{GB\to C} = I, \tag{2.15e}$$

where $R_x(\phi)$, $R_y(\theta)$, $R_z(\psi)$ correspond with roll, pitch, and yaw sub-rotations. Roll, pitch, and yaw are the standard navigation orientation coordinates which specify a body's orientation relative to the standard (north, east, down) inertial navigation frame. These rotations are performed in the order of yaw, then pitch, and then roll. The roll, pitch, and yaw sub-rotation matrices are equal to

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}, \tag{2.16a}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{2.16b}$$

$$R_z(\phi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.16c}$$

$R_y(\beta)$ and $R_x(\alpha)$ are tilt and pan sub-rotations corresponding with the orientation of the gimbal relative to the body of the aircraft. The order of these sub-rotations is first pan and then tilt. The definition of pan and tilt depends on the chosen default orientation of the

gimbal. Often, to protect a camera that would be mounted on the gimbal, a gimbal's default, initial orientation is straight forward. Accordingly, it is assumed that this is the unrotated orientation of the gimbal. Following the camera frame convention of coordinate frames, as mentioned above, the gimbal z coordinate is then set to be in the direction in which the gimbal is pointing. Pan is then a rotation about x, which initially points upward from the top of the aircraft. And tilt is then a rotation about y. Tilt and pan sub-rotation matrices are then equal to

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{2.17a}$$

$$R_x(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.17b}$$

where α is the pan and β is the tilt of the gimbal. Combining the roll, pitch, and yaw of the aircraft body orientation relative to the navigation coordinate frame and the tilt and pan of the gimbal body relative to the gimbal coordinate frame, these sub-rotation matrices become:

$$\begin{aligned} R_{N\to B} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\cos\theta \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \end{bmatrix}, \end{aligned} \tag{2.18a}$$

$$\begin{aligned} R_{G\to GB} &= \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos\beta\cos\alpha & \cos\beta\sin\alpha & -\sin\beta \\ -\sin\alpha & \cos\alpha & 0 \\ \sin\beta\cos\alpha & \sin\beta\sin\alpha & \cos\beta \end{bmatrix}. \end{aligned} \tag{2.18b}$$

With the rotation matrix solidified, a point in the camera image can then be rotated to world frame coordinates as

$$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = R^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{2.19}$$

Now consider the position of the camera in the world frame and the target appearing in the camera image. X is the position of the target in world frame coordinates. This point

projects onto the camera image at camera image coordinates $[\ x \quad y \quad 1\ ]^T$. This is interpreted as a vector pointing from the camera to the point in the camera image. Transforming the reference frame from camera frame coordinates to world frame coordinates, this vector can be described by the coordinates $[\ r_x \quad r_y \quad r_z\ ]^T$, as shown in Eq. 2.19. Assuming a flat earth, the camera's height above the ground is h. r_z and h then provide the depth of the point that is projected onto the camera image. As seen in Fig. 2.3, the vector $X - T$ is then

$$X - T = -\frac{h}{r_z} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \tag{2.20}$$

where it is realized that, if the camera is airborne and observing the ground, in world frame coordinates the z-axis coordinate is negative. So $r_z < 0$. We then get that the target's position is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ h \end{bmatrix} - \frac{h}{r_z} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}. \tag{2.21}$$

Notice that $Z = 0$, which again emphasizes that a flat earth model is assumed. Utilizing Eq. 2.21, given the vehicle position and a target detection in the camera image, the target's position in the world can be computed. The relation specified in Eq. 2.21 is utilized in the development of the camera observation likelihood functions.

With the camera geometry and perspective defined, the likelihood functions can now be developed. Recall that there are two types of observations, one for when a target is detected and one for when no target is detected. Target existence likelihood is drastically different for these two types of observations. Hence, there are two likelihood functions. These two likelihood functions are

1. Detected target position likelihood function.

2. Missed detection likelihood function.

Essentially, the first of these likelihood functions is very intuitive. It is a function that, given a detected target image position, the actual position probability density function is estimated taking into account uncertainty in camera position and orientation. The second likelihood function is less intuitive. It is a function which, given that no target is detected, estimates likelihood of a missed detection. This function is necessarily *not* a probability density function. Instead, it returns a probability function (not density) that does not integrate to 1 (as would a probability density). In order to understand this function a bit more, consider the case of when the camera's position and orientation are known perfectly and the camera has perfect detection accuracy. Then any point within the surveillance area that is within the camera's field of view would have a missed detection probability of zero. Similarly, any point outside of the camera's field of view would have a missed detection probability of one.

Moving on, first consider the detected target position likelihood function. Note that when a target is detected, this really means that *a* detection has occurred, whether this detection be of a true or false target. Consequently the detected "target" likelihood function $P(u|X,\hat{T},\hat{R})$ must account for whether or not the detection is true or false. Allow the binary random variable D_{true} to represent this true or false target detection (with value D_{true} when a true detection and value $\neg D_{\text{true}}$ when false). In $P(u|X,\hat{T},\hat{R})$, u is the detected "target"'s position in image coordinates (with the x and y axes being in the same direction as in the camera frame). This function is a probability density function specifying the likelihood of the target's position in camera frame coordinates. $P(u|X,\hat{T},\hat{R})$ cannot be determined directly due to the fact that it accounts for true or false target detection. However, $P(u|D_{\text{true}},X,\hat{T},\hat{R})$ is a quantity that can eventually be determined directly because it is given that the detection is a true detection (thus removing this degree of uncertainty). To utilize the fact that this quantity can eventually be determined, the law of total probability is applied to $P(u|X,\hat{T},\hat{R})$ over the variable D_{true} by noticing that

$$P(U,D_{\text{true}}|X,\hat{T},\hat{R}) = P(U|D_{\text{true}},X,\hat{T},\hat{R})P(D_{\text{true}}|X,\hat{T},\hat{R}). \tag{2.22}$$

The law of total probability then provides the likelihood function to be

$$\begin{aligned} P(u|X,\hat{T},\hat{R}) &= \sum_{D_{\text{true}}=\{D_{\text{true}},\neg D_{\text{true}}\}} P(u,D_{\text{true}}|X,\hat{T},\hat{R}) \\ &= P(u|D_{\text{true}},X,\hat{T},\hat{R})P(D_{\text{true}}|X,\hat{T},\hat{R}) \\ &\quad + P(u|\neg D_{\text{true}},X,\hat{T},\hat{R})P(\neg D_{\text{true}}|X,\hat{T},\hat{R}). \end{aligned} \tag{2.23a}$$

Each of the quantities in Eq. 2.23 are directly determinable. As mentioned above, $P(u|D_{\text{true}},X,\hat{T},\hat{R})$ is the position likelihood function of a target given that it is known that the target was correctly detected. $P(u|D_{\text{true}},X,\hat{T},\hat{R})$ is a probability density function. The determination of this quantity will be presented below because it requires significant computation. Paired with $P(u|D_{\text{true}},X,\hat{T},\hat{R})$ is $P(D_{\text{true}}|X,\hat{T},\hat{R})$, which is the probability of a target detection being correct given the target's true position and the estimate position and orientation of the camera. Note that $P(D_{\text{true}}|X,\hat{T},\hat{R})$ is *not* a probability density function. Also, this quantity can realistically be regarded as a constant even though it should technically be dependent on image resolution [58]. Allowing $P(D_{\text{true}}|X,\hat{T},\hat{R})$ to be a constant places the assumption on the computer vision algorithm that it will not generate target detections when image resolution is below an acceptable threshold. $P(u|\neg D_{\text{true}},X,\hat{T},\hat{R})$ is the counterpart of $P(u|D_{\text{true}},X,\hat{T},\hat{R})$, so it is not only a likelihood of a false detection (or false alarm), but it is also a probability density function. Because little to nothing is known about this false alarm probability density function, it is taken as a uniform distribution over the image [58], so it is represented as

$$P(u|D_{\text{true}},X,\hat{T},\hat{R}) = \begin{cases} \frac{1}{\text{image size}} & u \in \text{Image}(\hat{T},\hat{R}), \\ 0 & \text{otherwise}, \end{cases} \tag{2.24}$$

where Image$(\cdot,\cdot,\cdot)$ defines the set of points in the image plane that are within the camera image. The final quantity is $P(\neg D_{\text{true}}|X,\hat{T},\hat{R})$. This quantity is the probability of a target detection being incorrect given the true target position and estimate camera position and orientation. Before attempting to determine this quantity, recall that D_{true} is a binary random variable. Consequently $P(\neg D_{\text{true}}|X,\hat{T},\hat{R}) = 1 - P(D_{\text{true}}|X,\hat{T},\hat{R})$.

Now, to determine the probability density function $P(u|D_{\text{true}},X,\hat{T},\hat{R})$, notice that the estimate camera position and orientation are given. The camera estimate position and orientation are the major source of uncertainty in the likelihood function. Similar to the determination of $P(u|X,\hat{T},\hat{R})$, this uncertainty makes $P(u|D_{\text{true}},X,\hat{T},\hat{R})$ not directly determinable because $\hat{T}$ and $\hat{R}$ have uncertainty and this uncertainty is accounted for in $P(u|D_{\text{true}},X,\hat{T},\hat{R})$. As such, the law of total probability must once again be utilized to marginalize over this degree of uncertainty by integration. To do this note that, according to the chain rule of conditional probabilities,

$$\begin{aligned} P(U,T,R|D_{\text{true}},X,\hat{T},\hat{R}) &= P(U|D_{\text{true}},X,T,R)P(T,R|D_{\text{true}},X,\hat{T},\hat{R}) \\ &= P(U|D_{\text{true}},X,T,R)P(T,R|\hat{T},\hat{R}), \end{aligned} \tag{2.25}$$

where it is assumed that T and R are independent of D_{true} and X, given the estimates $\hat{T}$ and $\hat{R}$. Eq.2.25 can be further decomposed by assuming that the camera position and orientation are independent given estimators of the camera position and orientation. With this assumption of independence Eq. 2.25 then becomes

$$P(U,T,R|D_{\text{true}},X,\hat{T},\hat{R}) = P(U|D_{\text{true}},X,T,R)P(T|\hat{T})P(R|\hat{R}). \tag{2.26}$$

Observe that the quantity $P(U|D_{\text{true}},X,T,R)$ is now directly determinable. $P(U|D_{\text{true}},X,T,R)$ is a probability density function of a detected target's position in image plane coordinates given that the detection is the true target and given the true target position and true camera position and orientation. In fact, there is no longer any uncertainty in U, given D_{true}, X, T, and R. According to perspective geometry of the camera, Eq. 2.21, this probability density function must be

$$P(u|D_{\text{true}},X,T,R) = \delta\left(u - f_{\text{image}}(X,T,R)\right), \tag{2.27}$$

where the transformation performed by $f_{\text{image}}(X,T,R)$ can be understood by first defining the variable X_c as

$$X_c = R(X-T). \tag{2.28}$$

So X_c is simply the vector $X-T$ as viewed in the camera frame coordinates. This should not be confused with the image plane coordinates. In fact, the image plane coordinates are exactly what is output by $f_{\text{image}}(X,T,R)$. Allowing the components of X_c to be denoted by $X_c = \begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T$, $f_{\text{image}}(X,T,R)$ then outputs the two-dimensional vector $\begin{bmatrix} X_c/Z_c & Y_c/Z_c \end{bmatrix}^T$. These are the image plane coordinates as computed given a target position X and camera position T.

Putting everything together, implementing the law of total probability the probability density function for the detected target position likelihood function, given that the detection is truly of the target, becomes

$$\begin{aligned} P(u|D_{\text{true}}, X, \hat{T}, \hat{R}) &= \int_{R,T} P(u|D_{\text{true}}, X, T, R) P(T|\hat{T}) P(R|\hat{R}) d\mu(T) d\mu(R) \\ &= \int_{R,T} \delta\left(u - f_{\text{image}}(X, T, R)\right) P(T|\hat{T}) P(R|\hat{R}) d\mu(T) d\mu(R). \end{aligned} \tag{2.29}$$

Ideally Eq. 2.29 would be computed as is. However, following the approach taken in [58], the approach taken in this work was to linearize $f_{\text{image}}(X, T, R)$ about $\hat{T}$ and $\hat{R}$. As pointed out in [58], this linearization results in little loss of accuracy under typical levels of uncertainty over R. Then, with this linearization of $f_{\text{image}}(X, T, R)$, assuming $P(T|\hat{T})$ and $P(R|\hat{R})$ are Gaussian, the convolution of Eq. 2.29 is straight forward to compute analytically. Utilizing the assumption that $P(T|\hat{T})$ and $P(R|\hat{R})$ are Gaussian, $P(u|D_{\text{true}}, X, \hat{T}, \hat{R})$ becomes a linear function in the camera image frame coordinates u. However, the estimation algorithm estimates the target position X. So, despite the linearization of $f_{\text{image}}(X, T, R)$, $P(u|D_{\text{true}}, X, \hat{T}, \hat{R})$ is still a *nonlinear* function in X. As such, the likelihood function is nonlinear, because it is a function of X.

The computation of Eq. 2.29 can be simplified even further by making the assumptions

- The target's position uncertainty can be decomposed into three independent axes.
- The target's orientation uncertainty can be decomposed into the angles that define the rotation matrix.

According to [58], these assumptions are reasonable. With these assumptions, $P(T|\hat{T})$ and $P(R|\hat{R})$ can be decomposed as

$$P(T|\hat{T}) = P(T_1|\hat{T}_1) P(T_1|\hat{T}_2) P(h|\hat{h}) \tag{2.30a}$$

$$P(R|\hat{R}) = P(\phi|\hat{\phi}) P(\theta|\hat{\theta}) P(\psi|\hat{\psi}) P(\alpha|\hat{\alpha}) P(\beta|\hat{\beta}), \tag{2.30b}$$

where T_1, T_2, and h are the three axes of decomposition of the camera position uncertainty. T_1 is the direction in which the camera is moving and T_2 is the direction perpendicular to the camera motion. Then the detected target position probability density function $P(u|D_{\text{true}}, X, \hat{T}, \hat{R})$ in Eq. 2.29 becomes

$$\begin{aligned} P(u|D_{\text{true}}, X, \hat{T}, \hat{R}) = \int_{T_1,T_2,h,\phi,\theta,\psi,\alpha,\beta} & \delta\left(u - f_{\text{image}}(X, T, R)\right) P(T_1|\hat{T}_1) P(T_2|\hat{T}_2) P(h|\hat{h}) \\ & P(\phi|\hat{\phi}) P(\theta|\hat{\theta}) P(\psi|\hat{\psi}) P(\alpha|\hat{\alpha}) P(\beta|\hat{\beta}) d\mu(T_1) d\mu(T_2) \\ & d\mu(h) d\mu(\phi) d\mu(\theta) d\mu(\psi) d\mu(\alpha) d\mu(\beta). \end{aligned} \tag{2.31}$$

The linear approximation of $f_{\text{image}}(X, T, R)$ is accomplished by taking a first order Taylor series approximation about the free variables that compose the rotation matrix R and the camera position T [58]. The free variables over which this Taylor series is defined are

- Camera position variables: T_1, T_2, and h.
- Vehicle orientation variables: ϕ, θ, ψ.
- Camera relative to vehicle orientation variables: α and β.

Note that the value of X is input to the likelihood function, so it can be treated as a known constant here. The first order Taylor series approximation of $f_{\text{image}}(X, T, R)$ about the point $\left(X, \hat{T}, \hat{R}\right)$ is then:

$$\begin{aligned} f_{\text{image}}(X,T,R) \approx f_{\text{image}}(X,\hat{T},\hat{R}) &+ (T_1 - \hat{T}_1)\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (T_2 - \hat{T}_2)\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (h - \hat{h})\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial h}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (\phi - \hat{\phi})\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial \phi}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (\theta - \hat{\theta})\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial \theta}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (\psi - \hat{\psi})\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial \psi}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (\alpha - \hat{\alpha})\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial \alpha}\right|_{T=\hat{T},R=\hat{R}} \\ &+ (\beta - \hat{\beta})\left.\frac{\partial f_{\text{image}}(X,T,R)}{\partial \beta}\right|_{T=\hat{T},R=\hat{R}} \end{aligned} \tag{2.32}$$

The gradients involved in the Eq. 2.32 require some development. Each gradient will now be derived. The first two gradients are $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}$ and $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}$. To compute these gradients, first recall that

$$f_{\text{image}}(X,T,R) = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}, \tag{2.33}$$

where $X_c = R(X - T)$, as defined in Eq. 2.28. Then the chain rule of differentiation provides

that the two gradients $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}$ and $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}$ can be written in terms of X_c as

$$\begin{aligned}\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1} &= \frac{\partial}{\partial T_1}\begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial X_c}{\partial T_1}\frac{1}{Z_c} - \frac{X_c}{Z_c^2}\frac{\partial Z_c}{\partial T_1} \\ \frac{\partial Y_c}{\partial T_1}\frac{1}{Z_c} - \frac{Y_c}{Z_c^2}\frac{\partial Z_c}{\partial T_1} \end{bmatrix}, \end{aligned} \tag{2.34a}$$

$$\begin{aligned}\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2} &= \frac{\partial}{\partial T_2}\begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial X_c}{\partial T_2}\frac{1}{Z_c} - \frac{X_c}{Z_c^2}\frac{\partial Z_c}{\partial T_2} \\ \frac{\partial Y_c}{\partial T_2}\frac{1}{Z_c} - \frac{Y_c}{Z_c^2}\frac{\partial Z_c}{\partial T_2} \end{bmatrix}. \end{aligned} \tag{2.34b}$$

With the gradients decomposed in this manner, the form of $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}$ and $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}$ can now be determined by computing the gradients of $\mathbf{X}_c$, instead of $f_{\text{image}}(X,T,R)$, and then inputting these gradients into Eqs. 2.34.

Now to derive the form of the gradients $\frac{\partial \mathbf{X}_c}{\partial T_1}$ and $\frac{\partial \mathbf{X}_c}{\partial T_2}$, recall that T_1 is in the forward direction of the camera's motion and T_2 is perpendicular to this direction but within the same vertical plane parallel to the ground plane. As such, they can be defined by

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = R_{\text{heading}}(\gamma)\begin{bmatrix} T_x \\ T_y \end{bmatrix}, \tag{2.35}$$

where $R_{\text{heading}}(\gamma)$ is the two-dimensional rotation matrix defined by the heading with $\gamma = \pi/2 - \text{yaw} = \pi/2 - \psi$, where the yaw is taken as the heading. Inverting Eq. 2.35 results in a form which can be used to compute the first two gradients. The inversion provides

$$\begin{aligned} T_{xy} &= R_{\text{heading}}^T T_{12} \\ &= \begin{bmatrix} \cos\gamma & -\sin\gamma \\ \sin\gamma & \cos\gamma \end{bmatrix} T_{12}, \end{aligned} \tag{2.36}$$

where $T_{xy} = \begin{bmatrix} T_x & T_y \end{bmatrix}^T$ and $T_{12} = \begin{bmatrix} T_1 & T_2 \end{bmatrix}^T$. The chain rule of differentiation then provides the solutions to the first two gradients in Eq. 2.32. These first two gradients on $\mathbf{X}_c$ are

$$\begin{aligned}\frac{\partial \mathbf{X}_c}{\partial T_1} &= \frac{\partial T_x}{\partial T_1}\frac{\partial \mathbf{X}_c}{T_x} + \frac{\partial T_y}{\partial T_1}\frac{\partial \mathbf{X}_c}{T_y} \\ &= \cos\gamma\frac{\partial \mathbf{X}_c}{T_x} + \sin\gamma\frac{\partial \mathbf{X}_c}{T_y}, \end{aligned} \tag{2.37a}$$

$$\begin{aligned}\frac{\partial \mathbf{X}_c}{\partial T_2} &= \frac{\partial T_x}{\partial T_2}\frac{\partial \mathbf{X}_c}{T_x} + \frac{\partial T_y}{\partial T_2}\frac{\partial \mathbf{X}_c}{T_y} \\ &= -\sin\gamma\frac{\partial \mathbf{X}_c}{T_x} + \cos\gamma\frac{\partial \mathbf{X}_c}{T_y}, \end{aligned} \tag{2.37b}$$

from which it can be observed that these first two gradients on $\mathbf{X}_c$ can be written compactly as

$$\begin{bmatrix} \frac{\partial \mathbf{X}_c}{\partial T_1} \\ \frac{\partial \mathbf{X}_c}{\partial T_2} \end{bmatrix} = R_{\text{heading}}(\gamma) \begin{bmatrix} \frac{\partial \mathbf{X}_c}{T_x} \\ \frac{\partial \mathbf{X}_c}{T_y} \end{bmatrix}. \tag{2.38}$$

One final step is required to have the first two gradients $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}$ and $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}$ completely derived. This final step is to determine $\frac{\partial \mathbf{X}_c}{\partial T_x}$ and $\frac{\partial \mathbf{X}_c}{\partial T_y}$, which are directly computed and equal to

$$\begin{aligned} \frac{\partial \mathbf{X}_c}{\partial T_x} &= \frac{\partial}{\partial T_x} R(X - T) \\ &= - \begin{bmatrix} R_{11} \\ R_{21} \\ R_{31} \end{bmatrix}, \end{aligned} \tag{2.39a}$$

$$\begin{aligned} \frac{\partial \mathbf{X}_c}{\partial T_y} &= \frac{\partial}{\partial T_y} R(X - T) \\ &= - \begin{bmatrix} R_{12} \\ R_{22} \\ R_{32} \end{bmatrix}. \end{aligned} \tag{2.39b}$$

The first two gradients $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}$ and $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}$ have now been completely derived, although their derivation was somewhat long and tedious. Moving on, recall the application of the chain rule of differentiation in Eq. 2.34 to obtain $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_1}$ and $\frac{\partial f_{\text{image}}(X,T,R)}{\partial T_2}$ in terms of $\mathbf{X}_c$. Equivalent applications of the chain rule of differentiation are used to obtain the remaining gradients in Eq. 2.32 in terms of $\mathbf{X}_c$. This is done by letting

$$\begin{aligned} \frac{\partial f_{\text{image}}(X,T,R)}{\partial x} &= \frac{\partial}{\partial x} \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial X_c}{\partial x}\frac{1}{Z_c} - \frac{X_c}{Z_c^2}\frac{\partial Z_c}{\partial x} \\ \frac{\partial Y_c}{\partial x}\frac{1}{Z_c} - \frac{Y_c}{Z_c^2}\frac{\partial Z_c}{\partial x} \end{bmatrix}, \end{aligned} \tag{2.40}$$

where x is then replaced by the appropriate variable to get the desired gradient in terms of $\mathbf{X}_c$. With Eq. 2.40 established to function for each of the remaining gradients in Eq. 2.32, the remaining gradients that still require derivation are

- $\frac{\partial \mathbf{X}_c}{\partial h}$
- $\frac{\partial \mathbf{X}_c}{\partial \phi}$
- $\frac{\partial \mathbf{X}_c}{\partial \theta}$

- $\frac{\partial \mathbf{X}_c}{\partial \psi}$
- $\frac{\partial \mathbf{X}_c}{\partial \alpha}$
- $\frac{\partial \mathbf{X}_c}{\partial \beta}$

The first of these gradients is

$$\begin{aligned} \frac{\partial \mathbf{X}_c}{\partial h} &= \frac{\partial}{\partial h} R(X - T) \\ &= - \begin{bmatrix} R_{13} \\ R_{23} \\ R_{33} \end{bmatrix}, \end{aligned} \tag{2.41}$$

and the gradient $\frac{\partial \mathbf{X}_c}{\partial \phi}$ then has the form:

$$\begin{aligned} \frac{\partial \mathbf{X}_c}{\partial \phi} &= \frac{\partial}{\partial \phi} R(X - T) \\ &= \left(\frac{\partial R}{\partial \phi} \right) (X - T) \\ &= R_{B \to C} \left(\frac{\partial R_{N \to B}}{\partial \phi} \right) R_{W \to N} (X - T) \end{aligned}$$

The gradients over the other navigation Euler angles pitch and yaw have the same form. This leads to

$$\frac{\partial \mathbf{X}_c}{\partial x} = R_{B \to C} \left(\frac{\partial R_{N \to B}}{\partial x} \right) R_{W \to N} (X - T), \tag{2.42}$$

where x is set to either of the vehicle orientation Euler angles. The partial derivative is directly computed for any particular vehicle orientation Euler angle. As an example, consider the partial derivative for the case of the roll angle ϕ. The derivative is more readily computed by decomposing the rotation matrix $R_{N \to B}$ into the yaw, pitch, and roll sub-rotation matrices. Then, for the case of the roll angle, the partial derivative is computed as:

$$\begin{aligned} \frac{\partial R_{N \to B}}{\partial \phi} &= \left(\frac{\partial R_x(\phi)}{\partial \phi} \right) R_y(\theta) R_z(\psi) \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sin\phi & \cos\phi \\ 0 & -\cos\phi & -\sin\phi \end{bmatrix} R_y(\theta) R_z(\psi) \end{aligned} \tag{2.43}$$

In similar manner, the remaining partial derivatives of $R_{N \to B}$ can be computed. The three

partial derivatives of $R_{N\to B}$ are then

$$\frac{\partial R_{N\to B}}{\partial \phi} = \begin{bmatrix} 0 & 0 & 0 \\ \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \cos\phi\cos\theta \\ \cos\phi\sin\psi - \sin\phi\sin\theta\cos\psi & -\cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\theta \end{bmatrix}, \tag{2.44a}$$

$$\frac{\partial R_{N\to B}}{\partial \theta} = \begin{bmatrix} -\sin\theta\cos\psi & -\sin\theta\sin\psi & -\cos\theta \\ \sin\phi\cos\theta\cos\psi & \sin\phi\cos\theta\sin\psi & -\sin\phi\sin\theta \\ \cos\phi\cos\theta\cos\psi & \cos\phi\cos\theta\sin\psi & -\cos\phi\sin\theta \end{bmatrix}, \tag{2.44b}$$

$$\frac{\partial R_{N\to B}}{\partial \psi} = \begin{bmatrix} -\cos\theta\sin\psi & \cos\theta\cos\psi & 0 \\ -\cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & 0 \\ \sin\phi\cos\psi - \cos\phi\sin\theta\sin\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & 0 \end{bmatrix}. \tag{2.44c}$$

The partial derivatives on the total rotation matrix over the gimbal pan and tilt angles are derived in a similar fashion. The decomposition for the partial derivative computation is

$$\frac{\partial R}{\partial x} = R_{GB\to C}\left(\frac{\partial R_{G\to GB}}{\partial x}\right) R_{W\to}, \tag{2.45}$$

where x is replaced with either of the pan or tilt relative gimbal orientation angles. The partial derivatives of $R_{G\to GB}$ over the pan and tilt Euler angles are then

$$\frac{\partial R_{G\to GB}}{\partial \beta} = \begin{bmatrix} -\sin\beta\cos\alpha & -\sin\beta\sin\alpha & -\cos\beta \\ 0 & 0 & 0 \\ \cos\beta\cos\alpha & \cos\beta\sin\alpha & -\sin\alpha \end{bmatrix}, \tag{2.46a}$$

$$\frac{\partial R_{G\to GB}}{\partial \alpha} = \begin{bmatrix} -\cos\beta\sin\alpha & \cos\beta\cos\alpha & 0 \\ -\cos\alpha & -\sin\alpha & 0 \\ -\sin\beta\sin\alpha & \sin\beta\cos\alpha & 0 \end{bmatrix}. \tag{2.46b}$$

Derivation of all of the components that make up the gradients in the linear approximation of the transformation $f_{\text{image}}(X, R, T)$ in Eq. 2.32 have now been completed. This linear approximation, with the independence and Gaussian assumptions, then enables decomposing the integral of Eq. 2.31 into a series of one-dimensional convolutions. Observing Eq. 2.31, there is a convolution for each camera position and orientation degree of freedom that has uncertainty and consequent error from measurement. If the uncertainty for each of these position and orientation degrees of freedom errors is assumed to be derived from a Gaussian distribution, then each convolution can be performed analytically and a closed form solution for the integral in Eq. 2.31 can be obtained. Approximating these errors with Gaussian distributions is commonly done in inertial navigation systems, so these assumptions were considered reasonable.

The convolutions of Eq. 2.31, after linear approximation of $f_{\text{image}}(X, R, T)$ according to Eq. 2.32, results in a Gaussian distribution in camera image frame coordinates u, whose

mean is simply the image projection transformation on the estimator values of $\hat{R}$ and $\hat{T}$, assuming all of the error distributions have zero mean (unbiased). Note, however, that if there is any bias in the error distributions, then the mean will be shifted.

With the development of the detected target position likelihood function, there is still one piece that requires mention. This piece is that of considering the domain of the likelihood function and the range of the target position X. For any point in the surveillance area $X \in S$, from Eq. 2.28, $X_c(X) = (x_c, y_c, z_c)$ is the vector pointing from the camera to the point X, as viewed in the camera frame. Note that the coordinate z_c is in the direction in which the camera points. z_c must be positive for the likelihood function to be well defined. The domain of the likelihood function, intersected with the surveillance area S, can then be defined as $\{x \in S : z_c(x) \geq 0\}$. The likelihood function domain can be computed from the intersection of the camera image plane and the ground plane. This intersection defines a line on the ground plane within the surveillance area. This line divides the surveillance area into two regions. The region in the direction in which the camera points is the domain of the likelihood function intersected with the surveillance area. The other region is outside the domain of the likelihood function. Initially, this may not seem like a problem because the intersection of these planes is well outside the camera's field of view. However, return again to the application of the likelihood function. Its application is in the target position estimation algorithm. The estimation algorithm calls the likelihood function as a function of target position X. The range of X has no respect for the domain of the likelihood function. Consequently, the case will often arise that the estimation algorithm calls the likelihood function with a value for X that is outside the likelihood function's domain. For example, if the estimation algorithm is grid based, then the estimation space is fixed and will often consist of points outside the likelihood function's domain. As such, it is necessary to account for the likelihood function domain. There are likely many methods for accounting for the domain of the likelihood function. The method taken here is to extend the likelihood function domain to include all of the surveillance area. To do this, all points outside the likelihood function domain are given a value equal to the minimum value of the likelihood function. The minimum value of the likelihood function is used so that the likelihood function can still be normalized (since the likelihood function is not normalized). The minimum value is found by computing the intersection of the camera image plane and the ground plane.

The intersection line is the line obtained by intersecting the two planes $z_c = 1$ and $Z = 0$. These are the camera image plane and ground plane, respectively. Recall that the relation between the camera and a point on the ground is

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} T_x \\ T_y \\ h \end{bmatrix} + R \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}. \tag{2.47}$$

Rearranging, one gets

$$R \begin{bmatrix} X - T_x \\ Y - T_y \\ Z - h \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}, \tag{2.48}$$

from which can be extracted the equation

$$z_c = R_{3,1}(X - T_x) + R_{3,2}(Y - T_y) + R_{3,3}(Z - h). \tag{2.49}$$

Eq. 2.49 then defines the intersection specific to the camera image plane and the ground plane by setting $z_c = 1$ and $Z = 0$ to get

$$1 + R_{3,3}h + R_{3,1}T_x + R_{3,2}T_y = R_{3,1}X + R_{3,2}Y. \tag{2.50}$$

Now what is required is a point along this intersection that has minimum likelihood function value. To get this point, another line on the ground plane is required. This line is chosen to be the line defined between the camera and the point of observation of the camera on the ground plane. If the observation point is (X_0, Y_0), then this line is

$$Y - T_y = \frac{Y_0 - T_y}{X_0 - T_x}(X - T_x). \tag{2.51}$$

Rearranging this provides

$$(Y_0 - T_y)T_x - (X_0 - T_x)T_y = (Y_0 - T_y)X - (X_0 - T_x)Y. \tag{2.52}$$

Combining Eq. 2.50 and Eq. 2.52 yields

$$A \begin{bmatrix} X \\ Y \end{bmatrix} = b, \tag{2.53}$$

where

$$A = \begin{bmatrix} R_{3,1} & R_{3,2} \\ Y_0 - T_y & -(X_0 - T_x) \end{bmatrix}, \tag{2.54}$$

and

$$b = \begin{bmatrix} 1 + R_{3,3}h + R_{3,1}T_x + R_{3,2}T_y \\ (Y_0 - T_y)T_x - (X_0 - T_x)T_y \end{bmatrix}. \tag{2.55}$$

Because A is square and full rank, the point of minimum likelihood function value is then

$$\begin{bmatrix} X \\ Y \end{bmatrix} = A^{-1}b. \tag{2.56}$$

The point defined by Eq. 2.56 then provides a well defined point on the ground whose likelihood function value is the minimum value and can be used as the value for any point on the ground that has an ill defined likelihood function value.

Putting this all together, the likelihood function for the case when a positive detection is made can be constructed. Unfortunately not every scenario can be presented here. However, a sample detected target position likelihood function is presented in Fig. 2.5 for error parameters that have been used in some flight experiments [83]. This likelihood function is for the case when a detection is made at (0.1, 0.1) in image frame coordinates. In this figure, the error standard deviations on the camera position and orientation are

- ψ (yaw) error: 0.1 rad.
- θ (pitch) error: 0.05 rad.
- ϕ (roll) error: 0.05 rad.
- T_f (longitudinal) error: 13.6 meters.
- T_l (lateral) error: 5.5 meters.
- T_h (height) error: 5 meters.

Additionally, the vehicle/camera height was 100 meters above ground and the roll, pitch, and yaw were all zero. The gimbal pan was $\pi/4$ and the tilt was $\pi/6$ off of straight down.

Now to highlight the effects of the various position and orientation errors on the likelihood function shape, a few plots of the likelihood function will be presented for different error parameters. For each of these plots the same camera position and orientation and detection image coordinates are used. First, a baseline likelihood function is presented in Fig. 2.6. For this likelihood function, the error parameters are all small with values

- ψ (yaw) error: 0.01 rad.
- θ (pitch) error: 0.01 rad.
- ϕ (roll) error: 0.01 rad.
- T_f (longitudinal) error: 4 meters.
- T_l (lateral) error: 4 meters.
- T_h (height) error: 4 meters.

Note that the effect of the position and orientation errors depends on orientation of the camera and the position of the detection in the camera image. For example, consider the case of the camera pointing straight down from the vehicle and the vehicle at zero rotation. Additionally, let the detection be in the center of the image, directly below the vehicle. In this case, yaw error will have no effect because any change in yaw will result in the same detection location. Height error will also have no effect because any change in height will result in the same detection location.

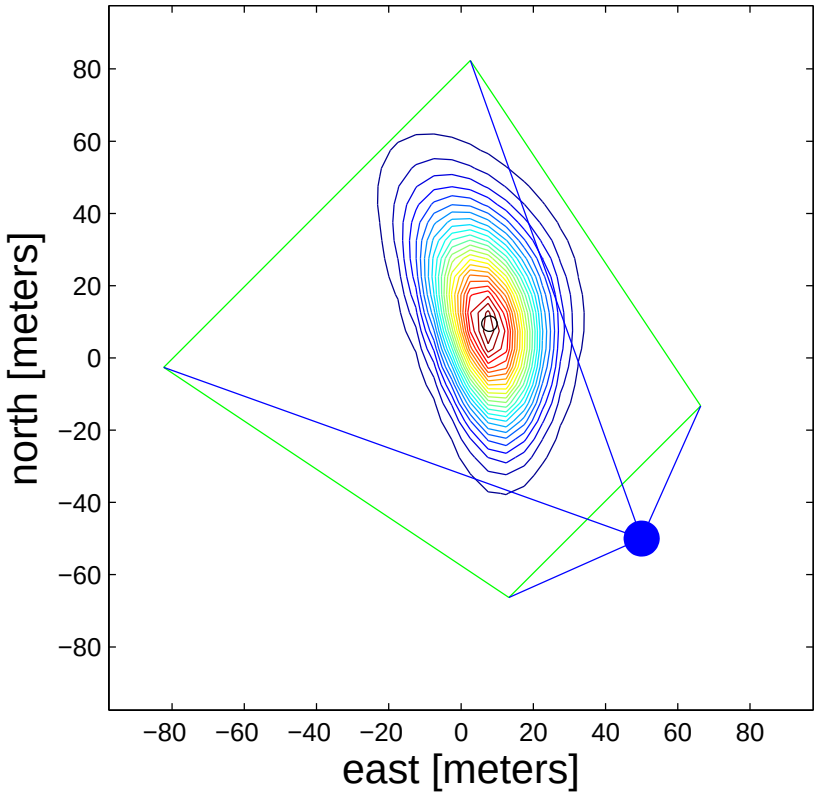


Figure 2.5: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.1, 0.05, 0.05); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (13.6, 5.5, 5). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event.

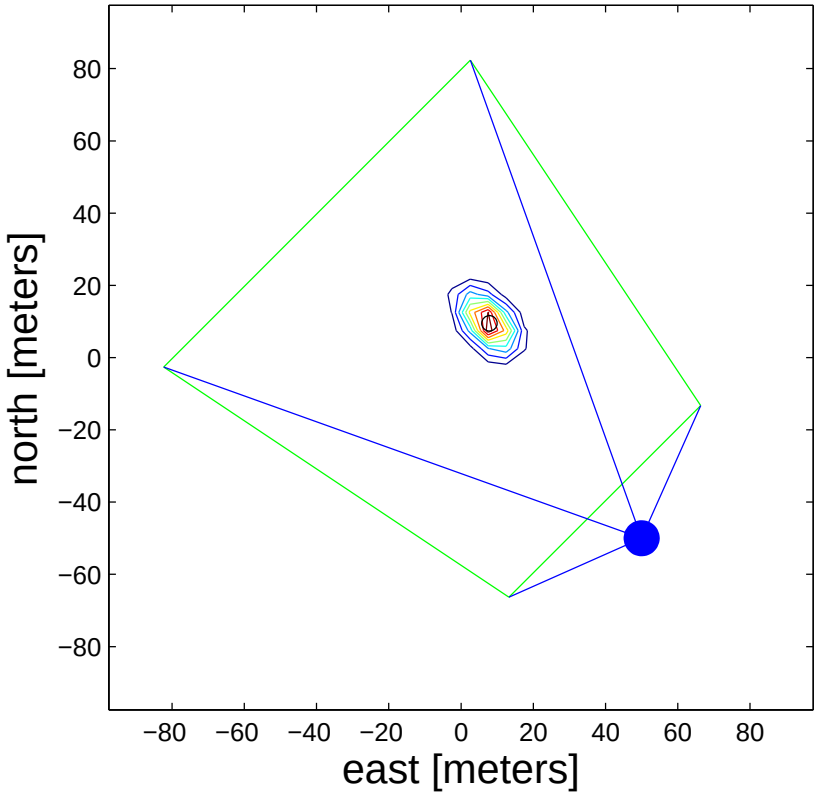


Figure 2.6: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.01, 0.01, 0.01); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (4, 4, 4). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Note that this figure is a baseline likelihood function, with minimal error, for comparison of the effects of the various error parameters. As such, observe that because there is low uncertainty in the camera position and orientation that the likelihood function uncertainty is low.

The effect of the vehicle orientation on the likelihood function will now be considered. To start off, consider the effect of the yaw error. Setting the yaw error standard deviation to 0.5 rad and leaving the other errors the same as the baseline likelihood function in Fig. 2.6, the likelihood function shape becomes that presented in Fig. 2.7. From Fig. 2.7 it is apparent that the effect of yaw error is to give the likelihood function a crescent shape. This makes sense because the yaw is rotation about the axis perpendicular to the ground plane.

Now consider the effect of pitch error on the likelihood function. Setting the pitch error standard deviation to 0.1 rad and leaving the other errors the same as the baseline likelihood function, the resulting likelihood function is presented in Fig. 2.8. Observe in this figure that the effect of pitch error is to stretch out the likelihood function in the direction of the vehicle.

Next consider the effect of roll error on the shape of the likelihood function. To observe the roll effect, the roll error standard deviation is set to 0.07 rad while leaving the remaining errors the same as that for the baseline likelihood function. Fig. 2.9 presents this likelihood function. Observe from this figure that roll error stretches the likelihood function shape in the direction lateral to the vehicle direction.

Now consider the effect of camera position error on the shape of the likelihood function. First consider the effect of longitudinal error on the likelihood function shape. Fig. 2.10 presents the effect of longitudinal error on the shape of the likelihood function. Observe that the effect is similar to the effect of pitch error. However, larger longitudinal errors can be tolerated than pitch errors. This is because a small angle of rotation results in a large deviation on the ground, whereas a small translation results in a small translation on the ground.

Next, consider the effect of lateral error on the shape of the likelihood function. Fig. 2.11 presents the likelihood function for the case when the lateral error is set to 10 m, and the other errors are kept the same as for the baseline likelihood function. Observe in Fig. 2.11 that the effect of lateral error on the shape of the likelihood function is to stretch it in the direction lateral to the direction of the vehicle. This is similar to the effect of the roll angle error in this case. This is because the roll angle is zero for this case. If the roll angle were non-zero, then it would stretch one lateral direction (e.g., the leftward direction) more than the other (e.g., the rightward direction). Yet, in this case lateral error would still have the same effect as the case when there is zero roll.

As a final consideration, consider the effect of height error on the shape of the likelihood function. To observe the effect of height error, set the height error standard deviation to 10 m while leaving the remaining errors the same of for the baseline likelihood function. The resulting likelihood function is presented in Fig. 2.12. From this figure it is apparent, for this case, height error flattens out stretches in both directions the likelihood function.

Moving on, the likelihood function for the case of observations corresponding to no targets detected must be developed. In this case the observation is $Z = \{\neg D, \emptyset\}$. And the likelihood function for this case is $P(\neg D|X, \hat{T}, \hat{R})$, which should be interpreted as a function of probability, *not* of probability density.

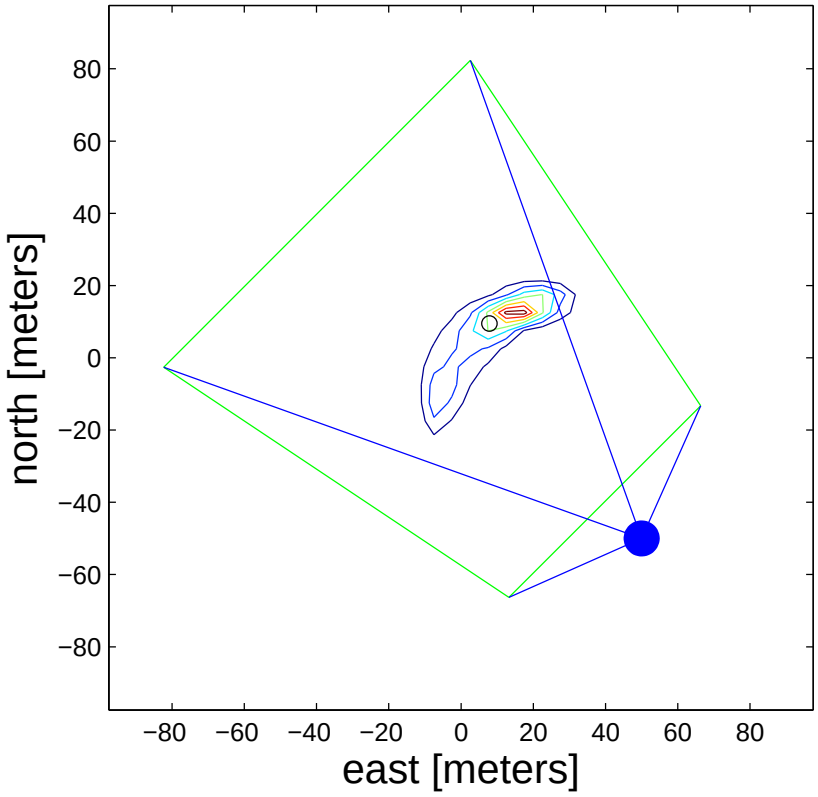


Figure 2.7: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.5, 0.01, 0.01); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (4, 4, 4). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Comparing this figure to Fig. 2.6, note that this figure highlights the effect of yaw error on the likelihood function. Observe that yaw error causes a crescent shape in the likelihood function.

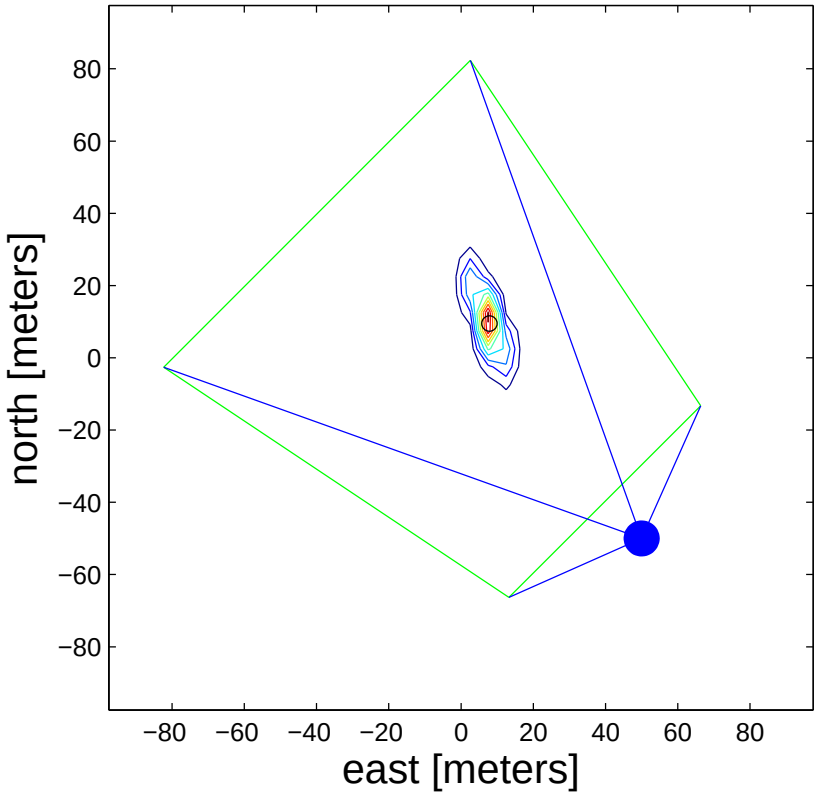


Figure 2.8: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.01, 0.1, 0.01); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (4, 4, 4). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Comparing this figure to Fig. 2.6, note that this figure highlights the effect of pitch error on the likelihood function. Observe that pitch error stretches the likelihood function in the direction of the aircraft.

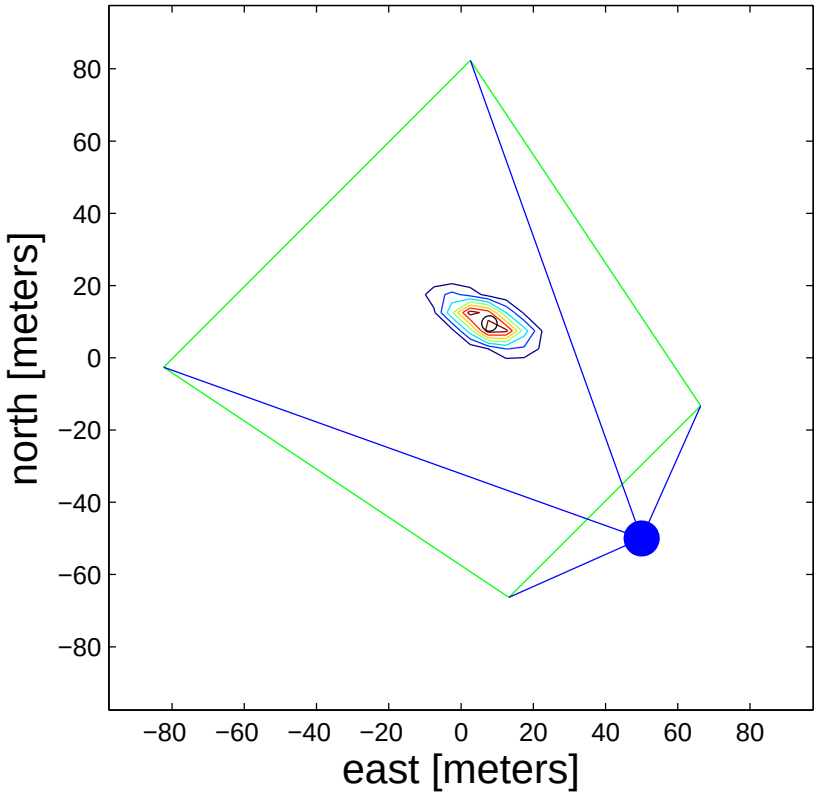


Figure 2.9: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.01, 0.01, 0.07); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (4, 4, 4). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Comparing this figure to Fig. 2.6, note that this figure highlights the effect of roll error on the likelihood function. Observe that roll error stretches the shape of the likelihood function in the direction lateral to the direction of the vehicle.

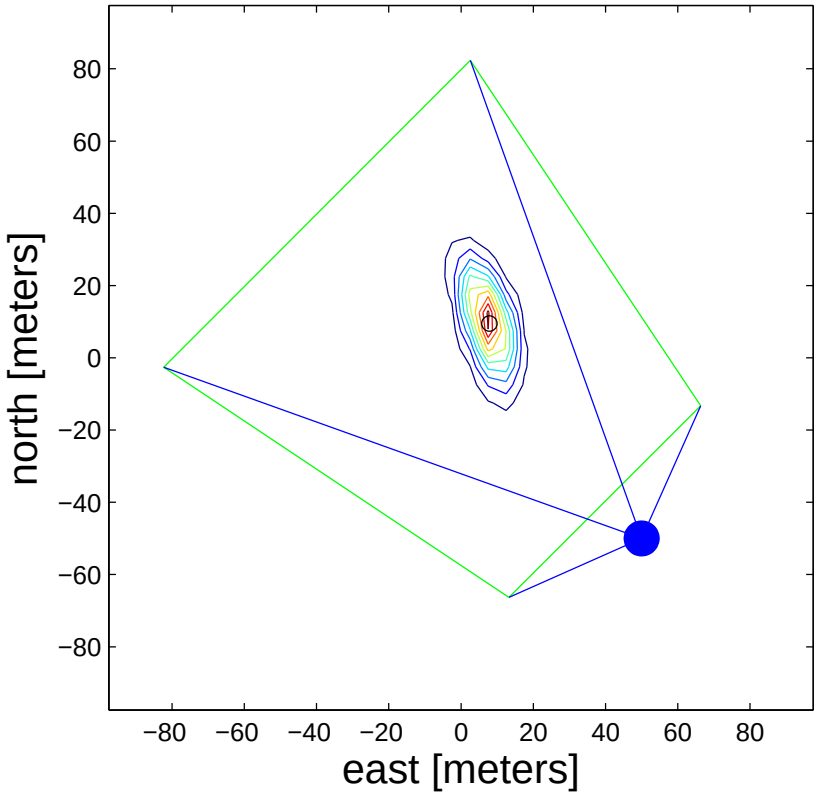


Figure 2.10: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.01, 0.01, 0.01); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (10, 4, 4). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Comparing this figure to Fig. 2.6, note that this figure highlights the effect of longitudinal error on the likelihood function. Observe that longitudinal error stretches the likelihood function in the direction of the vehicle.

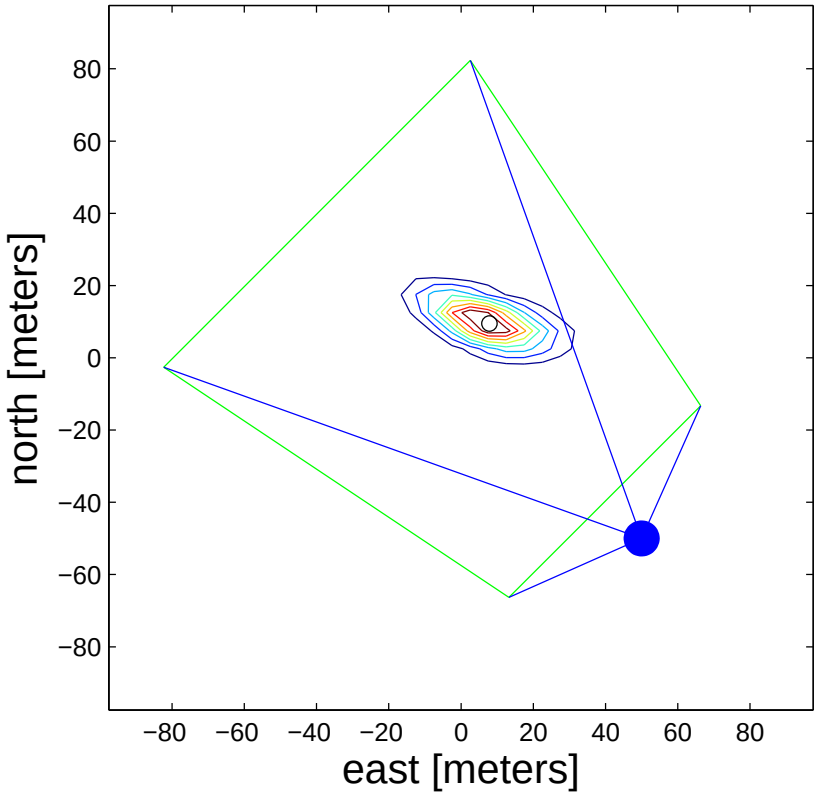


Figure 2.11: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.01, 0.01, 0.01); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (4, 10, 4). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Comparing this figure to Fig. 2.6, note that this figure highlights the effect of lateral error on the likelihood function. Observe that lateral error stretches the likelihood function in the direction lateral to the direction of the vehicle.

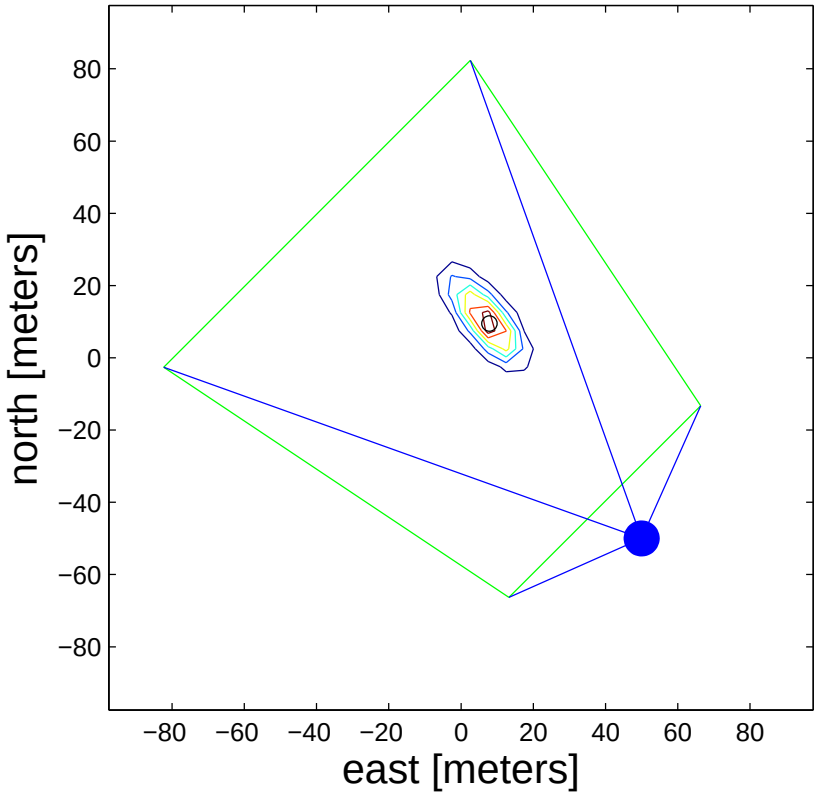


Figure 2.12: Sample detected target position likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.01, 0.01, 0.01); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (4, 4, 10). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. The detection image position is (0.1, 0.1) in image frame coordinates. This projects onto the world as the black circle plotted in this figure. Then about this position is defined the possible position of the true target that signaled the detection event. Comparing this figure to Fig. 2.6, note that this figure highlights the effect of height error on the likelihood function. Observe that height error flattens out and stretches in both directions the likelihood function.

To compute $P(\neg D|X,\hat{T},\hat{R})$, the approach is to instead compute $P(D|X,\hat{T},\hat{R})$ then

$$P(\neg D|X,\hat{T},\hat{R}) = 1 - P(D|X,\hat{T},\hat{R}). \tag{2.57}$$

Doing this transforms the derivation of the no-detection likelihood into that of a derivation of the detection likelihood, which is straight-forward to derive because it is essentially based on computation of the sensor field of view [58]. The derivation follows closely with that presented in [58], with differences consisting mainly with the computation of uncertainty because in [58] the camera orientation is assumed fixed and here it is assumed to move according to a gimbal's pan and tilt. Most of these differences have already been derived above for the case of the detected target likelihood function derivation (for example, with the rotation, rotation partial derivatives, and integration over uncertainty derivations). Although there are few further differences that have not already been clarified, to have a complete likelihood function presentation here, the no-detection likelihood function will now be derived.

The first step in the no-detection likelihood function derivation is integrate out the uncertainty in $P(D|X,\hat{T},\hat{R})$. This is accomplished by utilizing the Law of Total Probability to marginalize over the hidden variables T and R. But to do this it must first been mentioned that, according to the chain rule of conditional probabilities,

$$P(D,T,R|X,\hat{T},\hat{R}) = P(D|X,T,R)P(T,R|\hat{T},\hat{R}). \tag{2.58}$$

Recalling the decompositional assumptions made above, Eq. 2.58 becomes

$$\begin{aligned} P(D,T,R|X,\hat{T},\hat{R}) &= P(D|X,T,R)P(T,R|\hat{T},\hat{R}) \\ &= P(D|X,T,R)P(T_1|\hat{T}_1)P(T_2|\hat{T}_2)P(h|\hat{h}) \\ &\quad P(\phi|\hat{\phi})P(\theta|\hat{\theta})P(\psi|\hat{\psi})P(\alpha|\hat{\alpha})P(\beta|\hat{\beta}). \end{aligned} \tag{2.59}$$

The marginalization is then accomplished by integration as

$$\begin{aligned} P(D|X,\hat{T},\hat{R}) &= \int_{T,R} P(D|X,T,R)P(T,R|\hat{T},\hat{R})d\mu(T)d\mu(R) \\ &= \int_{T_1,T_2,h,\phi,\theta,\psi,\alpha,\beta} P(D|X,T,R)P(T_1|\hat{T}_1)P(T_2|\hat{T}_2)P(h|\hat{h}) \\ &\qquad P(\phi|\hat{\phi})P(\theta|\hat{\theta})P(\psi|\hat{\psi})P(\alpha|\hat{\alpha})P(\beta|\hat{\beta}) \\ &\qquad d\mu(T_1)d\mu(T_2)d\mu(h)d\mu(\phi)d\mu(\theta)d\mu(\psi)d\mu(\alpha)d\mu(\beta). \end{aligned} \tag{2.60}$$

In the process of evaluating this integral, recall the integration performed in Eq. 2.31 to derive the detected target image position probability density function. This integration was performed on the delta function of the estimator values for T and R. Consequently, the probability distribution obtained from Eq. 2.31 yields the probability distribution $P(T,R|\hat{T},\hat{R})$. So, once the integration of Eq. 2.31 is performed to yield the Gaussian distribution $P(T,R|\hat{T},\hat{R})$, the integration of Eq. 2.60 is accomplished by convolving $P(D|X,T,R)$ with this Gaussian distribution. Putting these pieces together, the detection likelihood is computed by

1. Computing the Gaussian distribution $P(T, R|\hat{T}, \hat{R})$ by performing the analytical integration in Eq. 2.31.

2. Convolving the Gaussian distribution $P(T, R|\hat{T}, \hat{R})$ with $P(D|X, T, R)$.

So all that is required is the form of $P(D|X, T, R)$. To obtain this, define a new random variable V that represents whether or not the target is actually within the sensor field of view. So V takes on the values V when the target is within the sensor field of view and $\neg V$ when the target is *not* within the sensor field of view. Once again utilizing the Law of Total Probability, $P(D|X, T, R)$ becomes

$$\begin{aligned} P(D|X,T,R) &= \sum_{V=\{V,\neg V\}} P(D,V|X,T,R) \\ &= \sum_{V=\{V,\neg V\}} P(D|V,X,T,R)P(V|X,T,R) \\ &= P(D|V,X,T,R)P(V|X,T,R) + P(D|\neg V,X,T,R)P(\neg V|X,T,R), \end{aligned} \quad (2.61)$$

where $P(D|V, X, T, R)$ is the probability of a detection given the target is within the sensor field of view, the true target position, and the true camera position and orientation; $P(V|X, T, R)$ is the probability of the target being in the sensor field of view given the true target position and camera position and orientation; $P(D|\neg V, X, T, R)$ is the probability of a detection given that the target is not within the sensor field of view (also called the false detection rate per image); and $P(\neg V|X, T, R)$ is the probability of the target *not* being within the sensor field of view.

The false detection rate, $P(D|\neg V, X, T, R)$, is a constant based on the already determined constant probability of correct detection above. The probability of visibility, $P(V|X, T, R)$, is

$$\begin{aligned} P(V|X,T,R) &= \begin{cases} 1 & f_{\text{image}}(X,R,T) \in \text{Image}(T,R) \\ 0 & \text{otherwise} \end{cases} \\ &= \prod_{i\in\{\text{left,right,top,bottom}\}} \text{Step}_i\left(f_{\text{image}}(X,T,R),T,R\right), \end{aligned} \quad (2.62)$$

where the one dimensional step functions Step_i are

$$\text{Step}_{\text{left}}(u,T,R) = \text{I}\left(u \in \left(X_{\min}^{\text{image}}, \infty\right]\right), \quad (2.63a)$$

$$\text{Step}_{\text{right}}(u,T,R) = \text{I}\left(u \in \left(-\infty, X_{\max}^{\text{image}}\right]\right), \quad (2.63b)$$

$$\text{Step}_{\text{top}}(u,T,R) = \text{I}\left(u \in \left(-\infty, Y_{\max}^{\text{image}}\right]\right), \quad (2.63c)$$

$$\text{Step}_{\text{bottom}}(u,T,R) = \text{I}\left(u \in \left(Y_{\min}^{\text{image}}, \infty\right]\right). \quad (2.63d)$$

The probability of the target not being within the sensor field of view is then simply $P(\neg V|X,T,R) = 1 - P(V|X,T,R)$. The final quantity to be derived is $P(D|V,X,T,R)$. $P(D|V,X,T,R)$ may initially seem like it should be equal to 1. However, just because the target is within the sensor field of view doesn't mean that it should be detected. As a camera is rotated away from being perpendicular to the ground plane, its field of view grows very large. Consequently, larger regions of the ground plane are projected onto the camera image pixels. As such, many of the camera image pixels will have resolution far too low to be useful. A couple immediate notes can be made from this knowledge. First, in planning paths for the camera and planning ground points to view, they should be chosen to maintain satisfactory image resolution. Second, $P(D|V,X,T,R)$ is heavily dependent on the image resolution. This leads to $P(D|V,X,T,R)$ having the form

$$\begin{aligned} P(D|V,X,T,R) &= P(D|V,\text{Res}(X,T,R)) \\ &= f_D(\text{Res}(X,T,R)), \end{aligned} \tag{2.64}$$

where the resolution $\text{Res}(X,T,R)$ is readily computed [58] and the functional format $f_D(\cdot)$ can be obtained by use of Bayes' Rule by

$$\begin{aligned} f_D(r) &= P(D|V,r) \\ &= \frac{P(r|D,V)P(D|V)}{P(r|V)} \\ &= \frac{P(r|D,V)P(D|V)}{\sum_{D\in\{D,\neg D\}} P(r|D,V)P(D|V)}. \end{aligned} \tag{2.65}$$

Putting this all together, the likelihood function for the case when *no* detection is made can be constructed. Unfortunately not every scenario can be presented here. However, a sample no-detection likelihood function is presented in Fig. 2.13. In this figure, the error standard deviations on the camera position and orientation are

- ψ (yaw) error: 0.1 rad.
- θ (pitch) error: 0.05 rad.
- ϕ (roll) error: 0.05 rad.
- T_f (longitudinal) error: 13.6 meters.
- T_l (lateral) error: 5.5 meters.
- T_h (height) error: 5 meters.

Additionally, the vehicle/camera height was 100 meters above ground and the roll, pitch, and yaw were all zero. The gimbal pan was $\pi/4$ and the tilt was $\pi/6$ off of straight down.

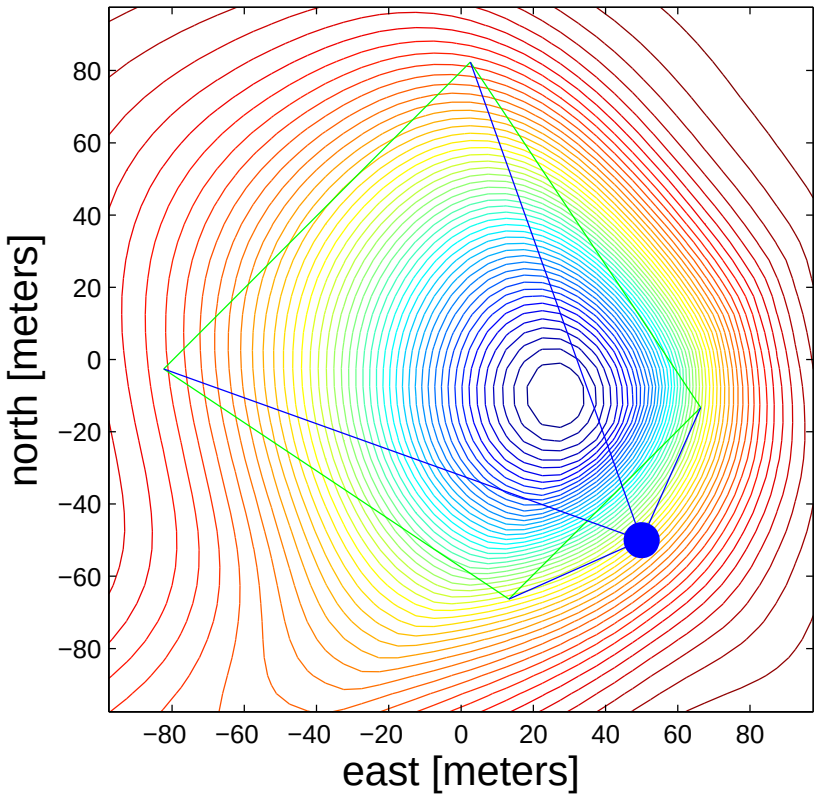


Figure 2.13: Sample no-detection likelihood function for the case when the camera is positioned at 100 meters above ground; the roll, pitch, and yaw are all zero; the gimbal pan is $\pi/4$ rad and the tilt is $\pi/6$ rad off of straight down; the standard deviations of the vehicle orientation (roll, pitch, yaw) errors are (0.1, 0.05, 0.05); and the standard deviations of the vehicle position (longitudinal, lateral, height) errors are (13.6, 5.5, 5). In this figure the filled circle represents the vehicle/camera position and the green trapezoidal lines on the ground represent an *approximation* of the camera field of view. Note that the contours in this figure correspond to a local minimum, rather than a local maximum. This means that, within the field of view, the probability of target existence is low, whereas outside the field of the target existence has probability equal to one. Applying this likelihood function in the estimation then will shift the probability away from the space over which the field of view has observed.

Transition Probability Design

As far as predicting the motion of a target goes, the worst case scenario is the one in which all that can be done is to assume some bounds on the speed at which the target may move. In this case there is very little known about where the target will transition and the prediction step really only becomes that of a step to add uncertainty to the target track estimate. The simplest means of adding this uncertainty is to apply a Gaussian transition probability. This is known generally by a diffusion model for the transition probability, and is determined simply by assigning the variance of the Gaussian distribution that models the diffusion. According to this model all that is assumed of the possible motion of the target is that it can proceed in any direction it wishes or even stay put. For this case of assumed possible target motion the prediction step becomes

$$\begin{aligned} P(X_T|z_{1:T-1}) &= \int_{x_{T-1}\in S} P(x_{T-1}|z_{1:T-1})P(X_T|x_{T-1})d\mu(x_{T-1}) \\ &= \int_{x_{T-1}\in S} P(x_{T-1}|z_{1:T-1})N(0,\tau^2)d\mu(x_{T-1}) \\ &= P(x_{T-1}|z_{1:T-1}) * N(0,\tau^2). \end{aligned} \tag{2.66}$$

Although this diffusion model appears to not have much detail of possible target motion, it is used in many estimation frameworks. In fact it is the model utilized in the Kalman Filter because if $P(x_{T-1}|z_{1:T-1})$ and $P(X_T|x_{T-1})$ are both Gaussian, then their convolution will be Gaussian, and the estimation can be accomplished by analytically computing the mean and variance of the Gaussian distribution.

However, even when little is known about the possible motion of the target, a simple diffusion model is typically not used for the case when it is assumed that the target is definitely moving. In this case a diffusion model places too much emphasis on the probability of the target staying in the same place. Instead, probability is shifted outward radially to form a donut shaped transition distribution as shown in Fig. 2.14. This distribution is defined mathematically with polar coordinates (r,θ) by

$$\begin{aligned} r &\sim N(\mu,\sigma^2), \\ \theta &\sim Uniform([0,2\pi)). \end{aligned} \tag{2.67}$$

With this definition of the transition probability the target is assumed to move according to

$$x_T = x_{T-1} + r\begin{bmatrix}\cos\theta \\ \sin\theta\end{bmatrix}. \tag{2.68}$$

The parameters of this donut shaped transition probability distribution can be physically defined by making assumptions on the type of target to be tracked. For example, if the target were a pedestrian, then one could assume that an average pedestrian would travel at 2 m/s ($\mu = 2$) and a fast pedestrian would travel around 4 m/s or faster ($\sigma = 2$).

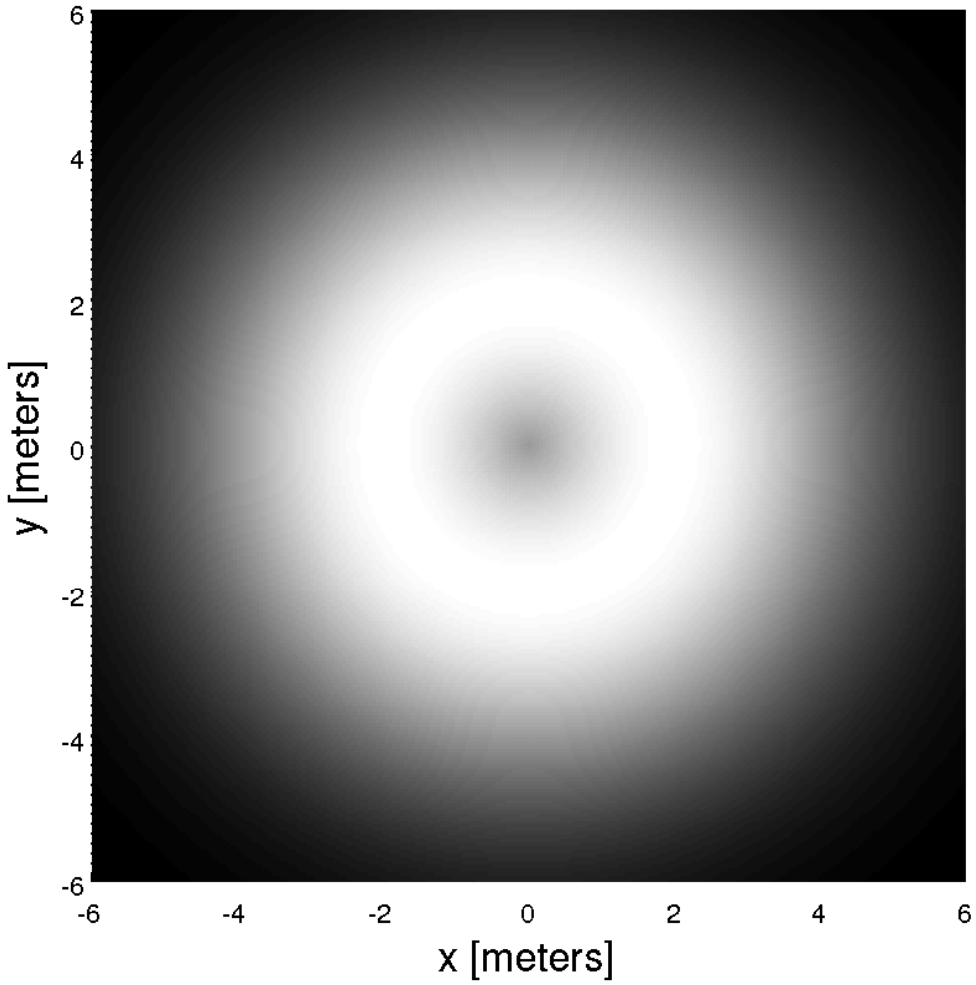


Figure 2.14: Transition probability distribution with the assumption that the target is definitely moving, so probability is shifted radially outward away from the distribution mean. This is what gives this type of transition model the name of a donut, because the center has little mass, as well as far from the center.

As more information is known about the assumptions that can be made on the possible target motion, the target transition probability distribution can be further refined. For example, it may be known that, given the target's past orientation, the direction it takes is constrained to be within $\pi/4$ of on its past orientation. However, notice that with more assumptions typically comes the requirement to estimate more details of a target's state. With the diffusion model for possible target transition, all that was required for target state estimation was the target position (say, two dimensions). Whereas if assumptions can be made on the target's motion by additionally considering the target's orientation, then the orientation must be included in the state vector (say, three dimensions now). Furthermore, the donut transition model could be improved by having the target speed mean and variance of the distribution estimated (say, four dimensions now). Then combining the orientation with the speed mean and variance estimation the target state to be estimated would have five dimensions for a two dimensional position vector.

The examples of possible target transition probability distributions presented so far have been derived from available assumptions on possible target motion based purely upon states of the target itself. However, more can be said about possible target motion. A target's motion may not only be dependent upon its own state but also on the state of the environment [20] and even other objects [45, 4, 49, 39, 50, 46, 52, 23]. For the case of the target state transition being dependent on the state of the environment, the target transition model could be a function of the state of the environment. For example, consider the possible speed at which a target could travel on various terrain such as unobstructed asphalt, tall grass, scree and boulder fields, a sheer cliff, knee high water, sand, etc. If the terrain is known, then the possible speed of a target definitely depends on the type of terrain upon which it travels. Consider Fig. 2.15, which plots mean target speed as a function of terrain, and the terrain being a function of location. Fig. 2.15 is an example of how speed can be made dependent on the environment. Other environmental effects can be readily modeled as well, such as currents in the ocean, the grade of a road or hillside upon which a target is found.

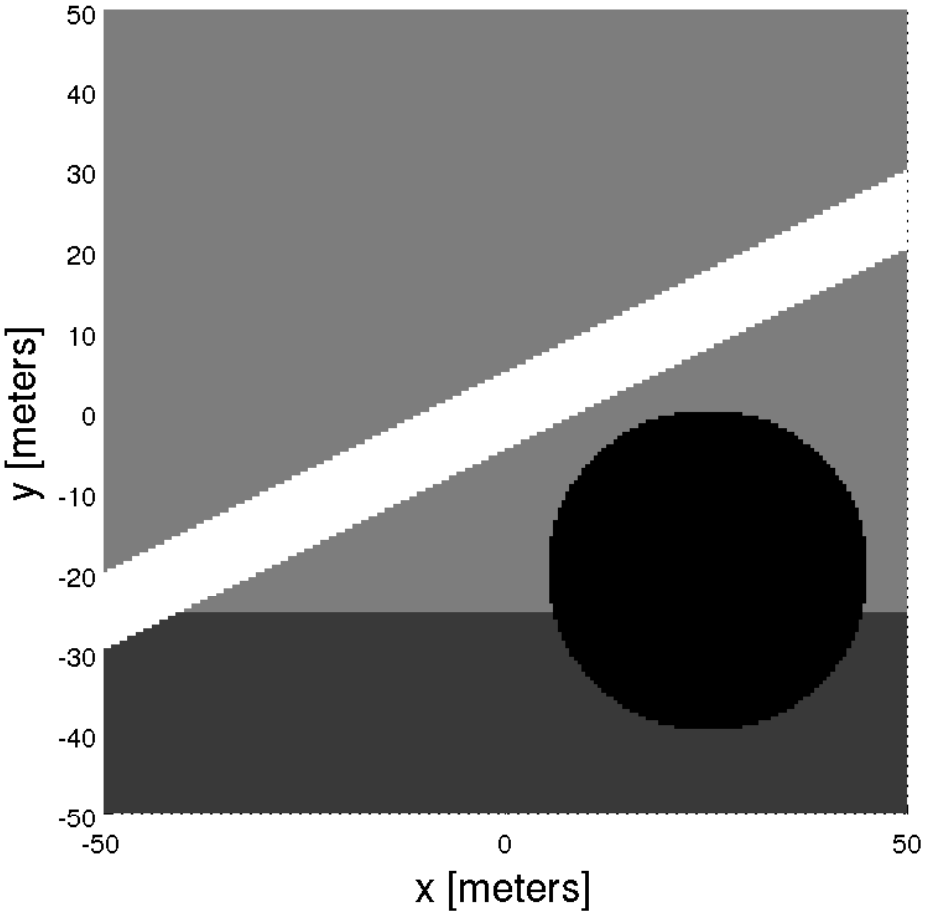


Figure 2.15: Mean target speed as a function of terrain, which terrain is a function of location. In this figure is shown the mean speed based on the type of terrain in a given location. In the image can be depicted the maximum speed achievable in the terrain to be along a road. In most regions off of the road the achievable speed is half that on the road. However, there are two regions in which the speed achievable is much less. These two regions are one that is circular. This region is the slowest. For example, it could be a large boulder that's difficult to climb. The other region is that region which covers the bottom of the image. An example of this region could be a mountain hillside.

Types of Single Target Estimation

The type of estimation performed for single target estimation depends on the details of the estimation structure. If both the likelihood function and target transition probability are modeled by simple Gaussian distributions, then a Kalman Filter, or other linear estimation methods, can be utilized to perform estimation because their solutions are analytical and easy to compute. In fact, this ease of computation often leads implementations of many non-Gaussian estimation problems to be approximated by Gaussian estimation. In many cases approximation by Gaussian distributions is very reasonable.

However, for the estimation structure utilized in this work, the Kalman Filter was not an option even in scenarios in which a target was assumed to be stationary and that it was known that there was only one target. The reason for this inability to implement the Kalman Filter was that all sensor observations were performed by a visual spectrum camera on board aircraft. The drastically non-Gaussian nature of camera-based observations obtained on board aircraft required methods of general recursive Bayesian estimation of not only non-Gaussian probability distributions but distributions that are non-parametric as well. For these general recursive Bayesian estimation structures, there are generally two types of estimation methods. These two types of estimation methods are

- Grid-based estimation.
- Sequential sample-based estimation.

Grid-based estimation methods are generally a discretization of a possibly continuous-space estimation problem. For example, in grid-based estimation methods a target's space of possible positions must be discretized into a finite number of cells. The standard grid-based estimation methods consequently consist of defining

1. A finite rectangular space for the estimation of target position.
2. A finite number of grid cells within the estimation space dimensions.
3. A possibly terrain dependent convolution mask of dimensions much smaller than the estimation space for the target transition probability.
4. A function for defining the likelihood grid for sensor observations of size equal to the estimation space.

The power of grid-based estimation methods is that values are guaranteed to persist for each cell in the grid and arbitrarily complex distributions can be represented. However, two main disadvantages of grid-based estimation methods are

- The convolution for the prediction step utilizes the target transition probability mask and is a computationally intensive procedure.

- The space over which the grid is defined is fixed, so targets are not allowed to leave the initial estimation space.

Some work has been done to address the issue of the grid space being fixed [63]. In this work [63], the grid is dynamically reconfigured to account for regions within the surveillance area that are more important. In fact, it is not even required for the grid cells to be rectangular. Instead, a finite element approach may be utilized. In this method the grid can expand and change shape to account for a target possibly leaving the domain of its initial grid space. Although rigorously powerful and capability of very detailed estimation, this method still results in heavy computation.

To counter the heavy computation involved in the prediction step convolution integral, and to avoid the restrictions of a finite sized estimation space, sequential sample-based estimation methods have been developed [7, 32, 33, 31]. Sequential sample-based estimation methods are typically known by one or more of the names Sequential Monte Carlo, particle-based methods, or particle filtering. Particle-based methods consist of a finite set of samples (called particles) whose collection present a description of the underlying probability distribution. Because this set consists of a collection of samples, the estimation space is not finite (as it was with grid-based methods) and the prediction step is computationally simple. Consequently, if estimation is performed by particle-based methods, the two disadvantages of grid-based methods are avoided. However, because particle-based estimation methods consist of a finite set of samples, as the probability distribution increases in complexity, problems are encountered consisting of

- Insufficiency of the number of samples in the set to describe the probability distribution.
- Lack of samples in many regions of the estimation space.
- Sensor observations not contributing to the estimation because they are in regions with no samples of the probability distribution.

In general, these problems involve the underlying issue of there being a lack of samples in the set that describes the distribution. This issue can even be encountered in regions where the underlying probability distribution has significant probability. For example, consider the scenario in which a target is being tracked. At some point in time the target stops and for some reason detections of that target cease for a time span longer than expected. As time progresses with no detections, the prediction step causes the samples to deviate far from position of the target. After the samples have deviated away from the target, a new target detection occurs. However, because there are no samples in the vicinity of the target position, the observation does little, if anything, to update the samples and bring them back to the position of the target. Consequently the set of samples diverges from the true probability distribution and the particle-based estimation likely fails completely to estimate the target position. This complete failure due to lack of a sufficient number of samples is an inherent disadvantage of particle-based estimation methods and a reason to be weary of

implementing a particle-based estimation method. Much work has been done to overcome this problem with particle-based estimation methods [31]. Realizing the capabilities of both grid-based and sample-based estimation methods, some approaches have even been developed that combine grid-based and sample-based methods [64].

2.3.2 Known Number of Multiple Targets

Target tracking for a known number of multiple targets can be accomplished by extending the methods developed for single target tracking. This is fortunate because significant development for the single target tracking case has been accomplished [47, 51, 89, 8]. However, significant estimation complexity is added by having multiple targets. This complexity is due to the question of how to fuse information form observations when the observations may be generated from any one or many of the targets. And it is not known from which target an observation has been made. The classical approach is to break the target tracking estimation problem into

1. Data association.

2. Information fusion, or estimation, given data association.

As far as the estimation step in concerned, the easiest approach would be to assume the data association is given and has no error. Then each of the target state could be estimated independently of each other. Then the problem of multiple target tracking estimation would become that of multiple single target tracking estimation problems. However, there will necessarily be error in the data association. So this error must be accounted in the estimation step. Accounting for data association uncertainty, the classical approach to accomplishing these two steps of multiple target tracking is a general method known as Multiple Hypothesis Tracking (MHT), for which variations, several details and special cases, and approximations have been well developed and analyzed [8, 11, 10, 92, 5, 9, 16, 17, 44, 100].

Recall that the methods for single target tracking were derived purely from the statistical formalism of Bayesian inference. Much of the elements of Multiple Hypothesis Tracking can be derived from Bayesian inference formalism. However, in order to perform the data association step, methods such as track tables are required, thus breaking the purely Bayesian inference formalism that was used to establish single target tracking. Instead, reliance on heuristics are typically required.

Yet, multiple target tracking can be derived completely from a Bayesian inference formalism [92, 15, 53, 54, 55, 56, 62, 101]. A completely general recursive Bayesian estimation problem can be defined by first noticing that what needs to be estimated is the set of target state processes $\{X^1, X^2, ..., X^N\}$ over time t, where each target state lives within some state space χ. Then, given a set of observations $\{z_1, z_2, ..., z_t\}$ over time, the recursive estimation problem begins with determining a probability distribution over the set of states at time t

given the set of observations up to time t. This probability distribution is

$$P(X_t^1, X_t^2, ..., X_t^N | z_1, z_2, ..., z_t) \equiv P(X_{1:t}^{1:N} | z_{1:t}). \tag{2.69}$$

It is more common in the target tracking literature to use a much more concise notation for the set of states [92]. This more concise notation is achieved by concatenating the set of state processes into one process S_t defined as

$$S_t = \begin{bmatrix} X_t^1 \\ X_t^2 \\ \vdots \\ X_t^N \end{bmatrix}. \tag{2.70}$$

So it is then consequential that S_t lives in the state space χ^N. Realizing that the quantity to estimate is once again just a random process, it is immediate that the recursive Bayesian estimation problem can be stated as

$$P(S_t | z_{1:t-1}) = \int_{s_{t-1} \in \chi^N} P(S_t | s_{t-1}) P(s_{t-1} | z_{1:t-1}) d\mu(s_{t-1}), \tag{2.71a}$$

$$P(S_t | z_{1:t}) \propto L(S_t; z_t) P(S_t | z_{1:t-1}), \tag{2.71b}$$

where Eq. 2.71a is the prediction step and Eq. 2.71b is the observation update step. Although this approach technically defines the general recursive Bayesian estimation problem, writing the multitarget tracking problem in this manner hides the inherent complexity. First, the dimension of this estimation problem is N times the dimension of the single target tracking problem dimension. Immediately the computational complexity is much worse, with the severity depending on the estimation algorithm chosen. Next, consider the structure of the transition probability distribution. To visualize this structure observe its graphical model as presented in Fig. 2.16. From Fig. 2.16, observe that there must be modeled inter-dependencies between a target at time t and the states of each of the other targets at time $t-1$. So the transition probability distribution is much more complicated than it was for the single target tracking problem. However, unless the presence of significant inter-dependence is known, typically there is assumed to be no target transition probability inter-dependence.

Now consider the likelihood function $L(S_t; z_t)$ as used in the observation update step in Eq. 2.71b. This likelihood function is much more complicated than the one for single target tracking. Yet, recall the derivation of likelihood functions for the case of observations being generated from camera images that was presented for the single target tracking problem above. That derivation was long and complicated. For real world sensors with significant complexity of measurements, likelihood functions are difficult to design well. Now imagine doing this for the much larger estimation dimension of multitarget tracking. The fact is, it's feasible and definitely worth investigation. However, this complexity is one of the motivations for performing multitarget tracking estimation by classical methods similar to Multiple

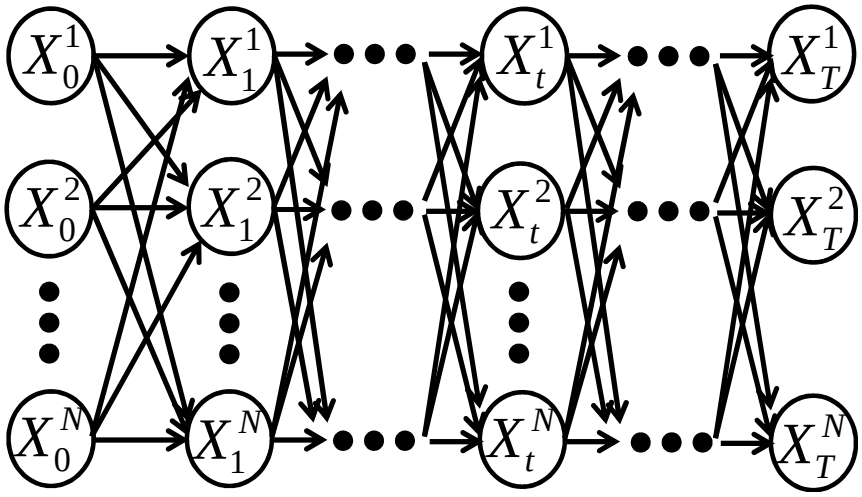


Figure 2.16: Graphical model of the multitarget transition probability distribution.

Hypothesis Tracking. Because in these methods, after observations have been associated to targets, target states are updated according to the exact likelihood functions that were designed for single target tracking. One way of understanding the main difference between classical Multiple Hypothesis Tracking type estimation methods and pure Bayesian type estimation methods is that the classical methods place the burden of work on data association whereas pure Bayesian type methods place the burden of the work on the likelihood function design. However, if it is desired to be able to specify the problem in an elegant manner, than the purely Bayesian type methods would be preferred. However, when it comes down to it, in real world applications either type of methods will likely have their fair share of heuristics.

Note that the approach that has been presented here, although for multiple targets, was for a fixed number of targets. However, within the framework presented, a somewhat rigid approach toward tracking an unknown number of targets can be derived. To do this allow the target state space to include the empty set $\emptyset$. The target state space is then $\chi \cap \emptyset$. Then include several initially empty state valued targets in the target concatenation vector S. As more targets are detected, empty state valued targets can then transition from empty to having some value in χ. Similarly, as targets leave, they can then take on the empty value. Incorporation of this new possibility for target states into the estimation algorithm involves modification at both the transition probability distribution and the likelihood function. For the transition probability distribution, target birth and death functions are required for transitions between empty state values and non-empty state values. Modification at the likelihood function requires the consideration of detections or signals coming from a new target. Before a new target can take on estimation value, it must be initialize with a prior distribution.

However, this approach for handling an unknown number of targets is necessarily rigid because the number of targets in the state vector must be preset. Consequently, if there happen to be more targets than spots for allocation in the state vector, then this approach will fail. On the other hand, if too many spots are provided for target allocation, then there will be significant overhead in the estimation algorithm accounting for empty target spots that are never filled.

2.3.3 Unknown Number of Targets

Typical engineering estimation applications, especially in target tracking [8, 9, 17], perform estimation over random vectors or processes (which consist of time sequences of random vectors) [14, 82]. A random vector can be viewed as being part of a deterministic set that has only one element (the element being the vector itself). The estimation theoretical formalism that has been presented so far has consisted of random vector/process estimation. When the number of targets is unknown, the estimation theoretical formalism for target tracking breaks down because the random vector consisting of a concatenation of target states that is estimated no longer has a known dimension. This is because the number of target states to concatenate onto the random vector is unknown. Depending on application, it may be possible to heuristically account for this by various methods [92, 9]. And much work has been done to develop heuristics to accomplish target tracking for an unknown number of targets for applications in which these heuristics can be reasonably designed, such as when using fixed radar systems for tracking traffic as it passes over the field of view of the sensor [16].

The strength of these methods is that, because they rely on random vectors, there can always be determined a probability distribution over the possible states of every target accounted for in the vector. So each target can be separated, by some fashion (perhaps even heuristic), from the rest and a precise estimate of that target's most likely state can be determined. However, for the work presented here, all that is required of the estimation algorithm is to output an estimate of the target density over the surveillance area. Although this target density estimate can be readily computed from random vector target tracking estimation methods, these methods are not necessary.

Target Density Distribution

Throughout the presentation of the methods developed in this paper, such as time-evolving partitioning for path planning, the existence of a target density distribution [104] is frequently mentioned. Consider the surveillance area S and some subarea $A \subset S$ of the surveillance area. The target density can be defined by a distribution $f(x)$ that is defined over the surveillance area as

$$\mathrm{E}\, N_A = \int_{x \in A \subset S} f\,(x)\, d\mu(x), \tag{2.72}$$

where N_A is the expected number of targets in the subarea A. Notice that the target density distribution provides the estimated number of targets within regions of the surveillance area. The expected total number of targets in the search area is then

$$\mathbf{E}N_S = \int_{x \in S} f(x)\, d\mu(x). \tag{2.73}$$

Observing Eq. 2.72, notice that the expected number of targets within a region can be evaluated simply by integrating the target density distribution over the region. The ability to quickly evaluate the number of expected targets within a region by integrating over a target density distribution is used frequently in time-evolving partition classification (also called dynamic partition classification) [104, 103] that will be presented in this paper.

It is not necessary to directly estimate the target density distribution. For example, in the event that the target states are estimated by a random vector based target tracking estimation method, a target density distribution can easily be computed. In the simplest case it is known that there is only one target. The estimation algorithm then outputs the probability distribution $P(X)$. The target density distribution is then simply $f(x) = P(x)$. When multiple targets are estimated, there are generally two cases. These cases are

1. The targets are estimated independently of each other.

2. The targets are inter-dependent.

In the first case, separate target distributions $\{P(X^1), ..., P(X^N)\}$ can be readily determined. So the target density distribution can be computed as $f(x) = \sum_i P(x^i)$. The second case is much more difficult. In the second case, marginalization can be utilized to achieve separate distributions used to compute the target density distribution.

Note that the dimension of the space over which the target density distribution is defined is equal to the dimension of a single target's position, which in this paper is typically assumed to be a two dimensional planar area representing the ground. Consequently, the space upon which the target density distribution is defined is much lower than the estimation space of random vector based multitarget tracking methods. Because the dimension of the space over which the target density distribution is defined is small, and the same for any number of targets, direct estimation of the target density distribution poses itself as an attractive option when the number of possible targets is unknown. There may be several ways to estimate a target density distribution. One such approach will be presented below by first discussing random sets [42] and then discussing an estimation method based on random sets [70].

Random Finite Sets

In order to preserve theoretical formalism as Bayesian inference and to enable target tracking in a more general unknown number of targets setting, instead of estimating a random *vector* of target states, a random *set* [42, 76, 75] of target states can be estimated. The difference

between a random vector and a random set is that a random vector has a deterministic number of elements, whereas a random set has a random number of elements, perfect for the case of an unknown number of targets.

Here let a random vector be denoted by $\mathbf{Y}$, with instantiation $\mathbf{Y} = \mathbf{y}$, where it is typically assumed that $\mathbf{y} \in R^N$ (with N being the number of elements in the vector). Then let a random finite set be denoted by Ψ, with instantiation $\Psi = Y$, where Y is a finite subset that can include $\emptyset$ or $\{\mathbf{y}_1, ..., \mathbf{y}_n\}$ where $n \geq 1$. Now, considering a region $A \subseteq S \subseteq R^N$ in the surveillance area, the statement $\Psi \subseteq A$ means $\{\mathbf{y}_1, ..., \mathbf{y}_n\} \subseteq A$, or all vectors in the set Ψ live in the region A.

Multitarget Calculus

In order to make use of random finite set notation in estimation, methods of calculus similar to those used for random vectors are required. These methods are called here multitarget calculus [70], simply to distinguish them from those used for single target estimation. Among the full set of methods, those that will be presented here are

- Set integrals.
- Multiobject density functions.
- Multiobject probability distributions.

Recall a set Y as defined above. A function $f(Y)$ on this set has set integral over a region A defined [70] by:

$$\begin{aligned}\int_{Y \subseteq A} f(Y)\delta Y :&= \sum_{n=0}^{\infty} \frac{1}{n!} \int_{\mathbf{y}_1 \in A, ..., \mathbf{y}_n \in A} f(\{\mathbf{y}_1, ..., \mathbf{y}_n\}) d\mathbf{y}_1 \cdots d\mathbf{y}_n \\ &= f(\emptyset) + \int_{\mathbf{y} \in A} f(\{\mathbf{y}\}) d\mathbf{y} + \frac{1}{2} \int_{\mathbf{y}_1, \mathbf{y}_2} f(\{\mathbf{y}_1, \mathbf{y}_2\}) d\mathbf{y}_1 d\mathbf{y}_2 + \cdots \end{aligned} \tag{2.74}$$

This leads to considering special forms required for the set function to satisfy Eq. 2.74. In fact, $f(Y)$ must be a multiobject density function. A multiobject density function is simply a set function for which, if $Y \subseteq S$ where S has unit of measurement u, then $f(Y)$ has unit of measurement $1/u^{\|Y\|}$. A special case of a multiobject density function is the case of a multiobject probability density function. A multiobject probability density function is a multiobject density function that satisfies

- $f(Y) \geq 0$.
- $\int_{Y \subseteq S} f(Y)\delta Y = 1$.

When the set function is a probability density function, the set function is denoted $f_\Psi(Y)$ or $f(\Psi = Y)$, and then it can be written that

$$\text{Prob}(\Psi \subseteq A) = \int_{Y \subseteq A} f_\Psi(Y)\delta Y. \tag{2.75}$$

Now, because a random finite set may have any number of elements in it, a probability distribution over the number of elements is useful. Let this probability distribution be called the cardinality distribution [70]. For the number of elements n, the cardinality is obtained by integrating over the space of n targets by:

$$\begin{aligned} P_\Psi(n) &:= P_{|\Psi|}(n) \\ &:= \text{Prob}(|\Psi| = n) \\ &:= \int_{|\Psi|=n} f_\Psi(Y)\delta Y \\ &= \frac{1}{n!}\int_{\mathbf{y}_1 \in S, ..., \mathbf{y}_n \in S} f_\Psi(\{\mathbf{y}_1, ..., \mathbf{y}_n\}) d\mathbf{y}_1 \cdots d\mathbf{y}_n \end{aligned} \tag{2.76}$$

It is also useful to get a measure for the probability of some $\mathbf{y}$ to be in the set Ψ. To determine this, all of the other possible elements W in Ψ must be marginalized. This is done by

$$\begin{aligned} \text{Prob}(\mathbf{y} \in \Psi) &:= \int_{\mathbf{y} \in \Psi} f_\Psi(Y)\delta Y \\ &= \int_{W \subseteq S} f_\Psi(\{\mathbf{y}\} \cup W)\delta W. \end{aligned} \tag{2.77}$$

Recursive Bayesian Estimation for Random Finite Sets

Recall the structure of recursive Bayesian estimation. The structure consists of

- Prediction step: convolution with transition probability.
- Observation Update step: likelihood function fusion.

This same structure can be utilized for estimation with random finite sets. Doing this involves defining the prediction step and associated multiobject transition probability or motion model, as well as the observation update step with associated multiobject likelihood function.

Let the time sequence of target states be denoted by

$$X^t := \{X_1, X_2, ..., X_t\}, \tag{2.78}$$

where each $X_i = \{\mathbf{x}_i^1, ..., \mathbf{x}_i^n\} \in \mathfrak{X}$ is a finite subset of target states at time i, instantiated from the finite random target state set Ψ_x. Similarly, let the time sequence of observations be denoted by

$$Z^t := \{Z_1, Z_2, ..., Z_t\}, \tag{2.79}$$

where each $Z_i = \{\mathbf{z}_i^1, ..., \mathbf{z}_i^m\} \in \mathfrak{Z}$ is a finite subset of observations made at time i. The multitarget transition density and multitarget likelihood function are represented by

- Transition density: $f_{\Psi_x}(X_t|X_{t-1})$.
- Likelihood function: $f(Z_t|X_t)$.

Multitarget recursive Bayesian estimation can then be defined by

$$f_{\Psi_x}(X_t|Z^{t-1}) = \int_{X_{t-1}\subseteq S} f_{\Psi_x}(X_t|X_{t-1}) f_{\Psi_x}(X_{t-1}|Z^{t-1}) \delta X_{t-1}, \tag{2.80a}$$

$$f_{\Psi_x}(X_t|Z^t) = \frac{f(Z_t|X_t) f_{\Psi_x}(X_t|Z^{t-1})}{f(Z_t|Z^{t-1})}, \tag{2.80b}$$

where Eq. 2.80a is the prediction step and Eq. 2.80b is the observation update step. In Eq. 2.80b, the normalizer is defined by

$$f(Z_t|Z^{t-1}) = \int_{X_t\subseteq S} f(Z_t|X_t) f_{\Psi_x}(X_t|Z^{t-1}) \delta X_t. \tag{2.81}$$

So for random finite sets, recursive Bayesian estimation can be performed by utilizing the set integral as defined in Eq. 2.74.

Approaches toward designing multitarget likelihood functions for the observation update step, Eq. 2.80b, and multitarget transition densities for the prediction step, Eq. 2.80a, will not be presented here. Their development is beyond the scope of the work presented here. Such design methods can be found in the literature [70].

It was mentioned previously that in some possible target scenarios it is useful to directly estimate the target density distribution. It was then alluded that this approach presented for estimating target tracks of an unknown number of targets would provide a means for directly estimating the target density distribution. So far it is not clear what connection there is between multitarget recursive Bayesian estimation as presented in Eq. 2.80 and the target density distribution as presented in Eq. 2.72. The target density distribution is a function specifying the density of targets within a region. However, Eq. 2.80 is actually a probability density distribution, although it is referred to as a multiobject probability density distribution. So, if Eq. 2.80 were implementable as presented, then the target density distribution would still not be directly estimated by this method.

However, there can be made a direct connection between multitarget recursive Bayesian estimation and the target density distribution. To make this connection, consider what a

statistical first moment would be for a random finite set with an associated multitarget probability density function. For the case of a random vector $\mathbf{Y}$ with instantiation $\mathbf{y} \in \mathfrak{Y}$, its statistical first moment is its expected value as defined by

$$\mathrm{E}\,\mathbf{Y} := \int_{y \in \mathfrak{Y}} \mathbf{y} P(\mathbf{y}) d\mu(\mathbf{y}). \tag{2.82}$$

It is tempting to use this same formula for defining the expected value for a random finite set Ψ with instantiation the finite subset Y, and associated multiobject probability density function $f_\Psi(Y)$. However, because addition of finite subsets has no definition [70], this formula must be modified. What is required is a transformation that takes a finite subset as input and outputs an equivalent vector [70]. The most common transformation to use for this [29] is the subset delta function defined as

$$\delta_Y(\mathbf{y}) := \begin{cases} 0 & \text{if } Y = \emptyset, \\ \sum_{\mathbf{u} \in Y} \delta_{\mathbf{u}}(\mathbf{y}) & \text{otherwise,} \end{cases} \tag{2.83}$$

where $\delta_{\mathbf{u}}(\mathbf{y})$ is the Dirac delta function centered at $\mathbf{y} = \mathbf{u}$. Now, denoting the random finite set statistical first moment by $D_\Psi(\mathbf{y})$ [70], the expected value of a random finite set is then

$$\begin{aligned} D_\Psi(\mathbf{y}) &:= \mathrm{E}\,\delta_\Psi(\mathbf{y}) \\ &= \int_{Y \subseteq S} \delta_Y(\mathbf{y}) f_\Psi(Y) \delta Y, \end{aligned} \tag{2.84}$$

where S is the entire state space of $\mathbf{y}$.

Now notice that the expectation defined in Eq. 2.84 produces a function on $\mathbf{y}$ that is itself a function outputting density, given input point $\mathbf{y}$ within a single object state space. As such, $D_\Psi(\mathbf{y})$ *is* the target density distribution, which is historically called the Probability Hypothesis Density (PHD) [42, 70].

Realizing that the random finite set expectation $D_\Psi(\mathbf{y})$ is the target density distribution, it is apparent that, for the purposes of time-evolving partition classification as presented in this work, there is no need for estimating the full multitarget probability density function. Estimation of $D_\Psi(\mathbf{y})$ is perfectly adequate.

In fact, it should be pointed out that implementation of the estimation of the complete multitarget probability distribution is not generally tractable, as presented in Eq. 2.80. Consequently, approximate estimation algorithms are required for actual implementation. The two main existing valid implementations [70] of approximate estimation of the multitarget probability density function are

- PHD filter.
- Cardinalized PHD (CPHD) filter.

The main attribute of these current implementations that they estimate the target density distribution (or PHD) $D_{\Psi}(\mathbf{y})$ in order to approximate $f_{\Psi_x}(X_t|Z^t)$. For the case of the PHD filter, the target density distribution is the only quantity estimated. Consequently the PHD filter is only a first moment estimator. Although this is adequate for this work, better accuracy can be achieved by estimating higher moments as well as the first moment. This is exactly what the CPHD filter attempts to accomplish. In addition to estimating the first moment (or target density distribution), the CPHD filter also estimates the cardinality probability distribution, Eq. 2.76, of the multitarget probability distribution.

Although PHD and CPHD estimation methods are only approximations to the multitarget probability density function, note that, for the time-evolving partition classification and path planning that will be presented later in this work, the quantities estimated in these implementations are all that is required. See the literature for the exact implementations of these methods [70].

2.4 Summary

This chapter has presented estimation for target tracking. It was highlighted that the appropriate choice of estimation depends on the assumptions that can be made regarding the possible number of targets as well as how the possible target are assumed to generally move and how observations of targets are made. These assumptions collectively establish the specific target tracking problem attributes that must be accounted for in the estimation method. Recall that the problem attributes for the problem presented in this work are

1. A single observation field of view is small relative to the entire surveillance area.
2. The placement of observations is dynamically constrained to the previous placement.
3. Multiple searchers are utilized in order to increase the observation coverage.
4. Searchers are autonomous agents who determine an appropriate search or tracking strategy.
5. The number of targets to search and track is unknown.

In order to understand the effect on estimation of attributes 1 and 2, consider an autonomous aircraft with a camera for sensing. In order to produce an image with sufficient clarity to analyze, the camera must be zoomed in enough to focus on its point of observation, causing the field of view to be small compared to the entire surveillance area which can be on the scale of several miles. If the field of view were not an issue, the estimation problem could be simplified by only updating the estimate with target detection observations. In this case, the observation likelihood functions could be modeled as Gaussian probability distributions and the target estimates could be Gaussians as well, with closed form solutions

such as various forms of Kalman filtering [3]. The main aspect of estimation would then be associating target detections with target estimates [9, 17]. However, in the case of small observation fields of view, most observations will be that no target has been detected. In order to still construct a target estimate, these no-detection observations must be used to update the target estimate through general recursive Bayesian estimation methods [91]. No-detection observation likelihood functions are necessarily non-Gaussian and not probability distributions. Consequently these methods lose any hope of closed form solution.

These methods have been extended to the case of multiple targets. Classic methods follow variations on multiple hypothesis tracking and are as a consequence not purely Bayesian estimation because they maintain track tables [9, 17, 92]. These methods place the bulk of the algorithmic development on data association. Alternatively, pure Bayesian estimation methods exist [92]. These methods shift the bulk of work back from data association to the design of multitarget likelihood functions. These multiple target tracking methods rely on random processes constructed from possible random vector time sequences. Consequently they cannot technically handle the case of the number of targets being unknown. However, implementations do actually handle such cases by simply concatenating excessive numbers of empty states in the state vector and allocating empty spots with new targets as they are detected. However, to do this requires heuristics.

Theoretically, the aspect of an unknown number of targets further complicates target estimation. The estimation space is $N_t \times N_s$ where N_t is the number of targets and N_s is the number of states to estimate for each target. N_t is variable and unknown. So the estimation space size is unknown and random. However, for time-evolving partition classification, presented later in this work, all that is required is the target density distribution as defined in Eq. 2.72. The estimation space of the target density distribution is equal to the state space of one target. As such, to get around the unknown estimation space size, instead of estimating each target individually as is done in classical multiple target tracking estimation methods, the target density distribution can be estimated directly [70]. The two main methods for directly estimating the target density distribution are the Probability Hypothesis Density (PHD) filter and the Cardinalized PHD (CPHD) filter, as discussed above.

However, if there are no complexities with the number of possible targets, or if methods have already been developed for multiple target tracking, then the estimation output from these methods can be used to construct the target density distribution. The target density distribution is required for the time-evolving partition classification presented in this work.

Chapter 3

Time-Evolving Partition Classification

3.1 Purpose of Partitioning

Recall that this work focuses on the target search and tracking problem in which the number of targets is unknown and target estimation is non-parametric to account for general nonlinear observations. When there are multiple targets, the estimation space is large. Furthermore, when the number of targets is unknown, the estimation space is even further complicated. Path planning directly over this complicated space is difficult. The approach taken in this work is to combine all target estimates into a target density distribution which is defined over the space of the surveillance area, which is small. Then, in this work, regions of characteristic information within the surveillance area are learned by analyzing the target density distribution. Then path planning can be performed over the set of regions. By doing this, the path planning is then performed over a known number of elements (the regions), instead of an unknown number of elements (the targets). In this chapter, the method for accomplishing region learning is presented.

3.2 Region-based Search and Tracking

The work presented here shifts the focus of area surveillance for target tracking from being target-based to being region-based. To motivate this shift of focus, consider a couple different types of surveillance scenarios.

First, consider the scenario in which an agent is given a large geographic area to survey. This geographic area is called the surveillance area. In this scenario there is initially no information known about the possible locations of targets. As such, all points in the surveillance area initially have the same value. All finite length observation paths then have the same value too. For these types of regions, it may be worthwhile to consider path planning methods other than information maximization. For example, perhaps the shape of the region could be utilized to start a search.

Second, consider the scenario in which there is significant prior information available about the surveillance area. This information can be utilized to aid in optimizing a search plan that yields paths that return high expected gains in information. Information gain optimal paths would consist of paths that attempt to improve the amount of information that is known about the surveillance area.

Third, consider the scenario in which an agent is given a very specific region where target location is known with high certainty. In this scenario, it is likely that very little work can be done to actually improve the knowledge of the target location. But rather, the information known about this target must be maintained. And because the target's position is well known, paths should be planned to keep the most likely target location in view. These paths would consequently be based on the available information. But the type of path is not chosen to improve the knowledge, but to exploit it.

These scenarios suggest that the strategy chosen for determining a search plan over a surveillance area should account for the type of region that is required to survey. In these scenarios, the amount of prior information was the main mechanism for choosing an appropriate strategy for search planning. In these scenarios it was assumed that the entire surveillance area had a characteristic type of prior information. Consequently, an appropriate choice of search strategy could be made. However, in general the information available very complex, especially after the search has been in progress. For example, observe the sample target density distribution presented in Fig. 3.1. Observe that in this figure, there may not necessarily be a single type of path planning that works best for all regions of the surveillance area. It is then the purpose of this chapter to establish methods for taking a complex information set (represented by a target density distribution) and partitioning the surveillance area into a set of time evolving regions that consist of simple information sets (partition level target density distributions). Over each of these region partitions, an appropriate choice of search strategy can then be made, as was done in the scenarios presented above.

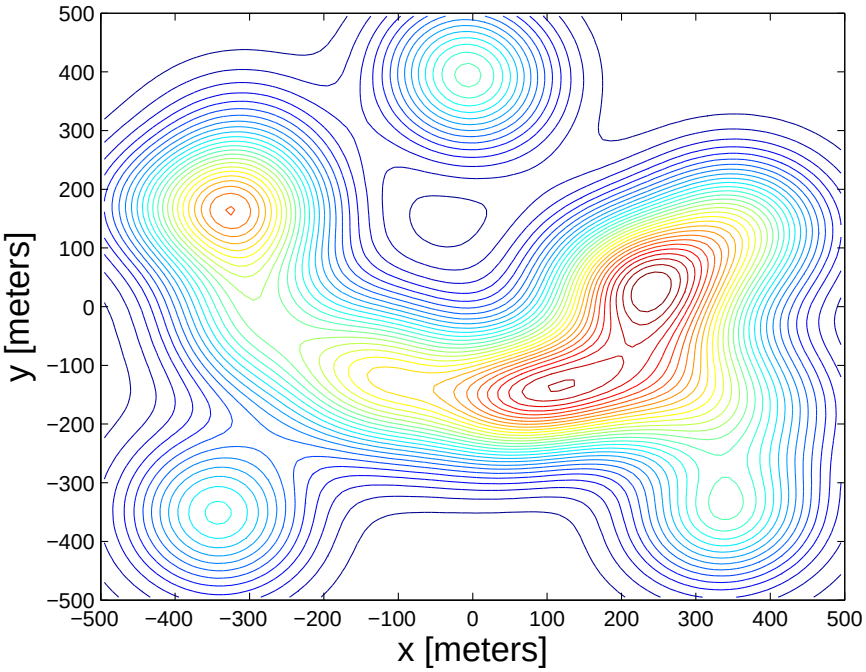


Figure 3.1: A sample target density distribution defined over a rectangular surveillance area. Observe the complexity of the information provided by this target density distribution. The partition classificatier should accordingly break this surveillance area into regions over which simple information is defined.

3.3 Types of Regions in the Surveillance Area

As discussed above, a surveillance area with a complex information set can be partitioned into regions consisting of simple information sets. Note that these regions must be time evolving, as the information set is time evolving. Additionally, in this work it is assumed that the information set defined over the surveillance area is a target density distribution, as defined in Eq. 2.72. Then, based on the type of simple information set defined over each region, the regions can be classified into various types. The type of region then specifies the type of search strategy that would be best for the search plan. As depicted in Fig. 3.2, the different types of regions, based on the type of search strategy that should be performed, are

1. An exploration partition
2. An ordered set of search and tracking partitions
3. A null target partition

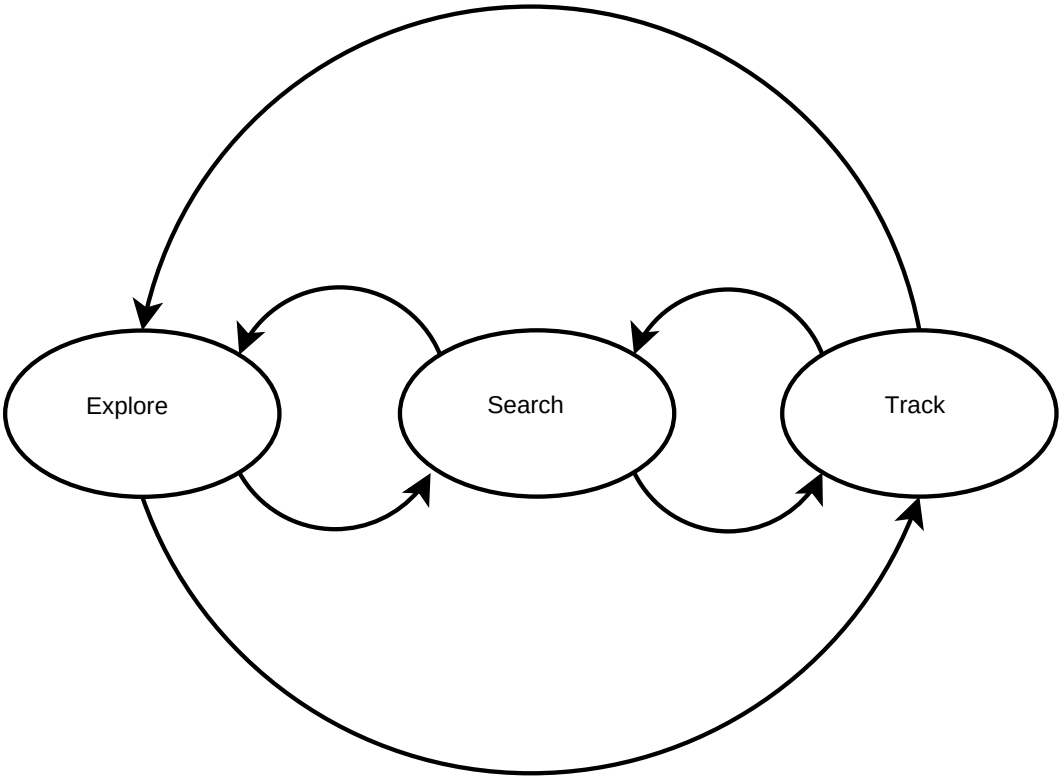


Figure 3.2: The path planning modes that should exist when surveillance is performed by region-based planning.

The exploration partition consists of areas within the surveillance area that have low to no information content. For example, consider a region over which there is defined a uniform probability distribution. Consequently, in these regions there is no bias that would guide a search plan. So the information content cannot be adequately utilized to optimize a search plan. As an alternative to information based planning, in these types of regions maximization of region coverage is more suitable [71, 72].

In terms of information content, the opposite type of region to the exploration partition would then be tracking partitions. Tracking partitions are small spatially and are partitions in which there is strong bias of target density and high certainty. These are partitions in which targets have been detected consistently. Consequently, tracking partitions define locations of known targets. These partitions must continue to be tracked according to the mobility of the tracked targets. The search strategy then becomes that of keeping observance of the known position of the targets. For example, the points of maximum density within tracking partitions are kept in observance. The search strategy for tracking partitions is then the most constraining on agent motion. For example, if an agent is a fixed-wing aircraft, it will have to fly orbit-like paths encircling the known position of the target [26, 84, 66, 90].

Slightly similar to tracking partitions are search partitions. The similarity search partitions have with tracking partitions is that both consist of an information set that can be utilized to aid in optimizing search plans. However, search partitions are different from tracking partitions in that the information content is not very certain. Consequently, not too much can be said about exactly where a target may be located. However, there is strong bias over which parts of the regions have high possibility of target existence. It then becomes the duty of a search plan to optimize paths, based on the information content, in order to actually improve the information content, with the aim to yield a new distribution with much higher certainty. Consequently, the search strategy becomes to maximize some type of information gain, and a searcher paths are then guided to improve the information content in order to ultimately observe a target [87, 21, 88, 43, 60, 81].

Over time some regions will be repeatedly observed. Much of the observed regions will never have targets detected. Based on anticipated possible target mobility, it can be concluded that no targets exist in these areas. Of course, search and tracking performance will limit which regions can be classified as not having targets. However, it is necessary to maintain a time-evolving partition that classifies regions in which no targets exist. These regions form the null target partition. The importance of this partition is seen by considering two scenarios. The first is when there is no target in the surveillance area. At some point in time, the conclusion should be reached that there is no target. This is accomplished by maintaining the null target partition. The second scenario is the one in which there is a vast exploration partition. As regions become fully observed but no targets have been detected, these observed regions should cease to be explored (according to possible target mobility and search performance).

3.4 Characterizing Regions into Partitions

In order to fully characterize regions in the surveillance area into the time-evolving partitions described above, the following steps are performed.

1. Determination of a set of region features.
2. Development of a feature-based partition classifier.
3. Association in time of partitions.
4. Learning for motion prediction of tracking partitions.

Before discussing these steps, note that all characterization of the surveillance area is accomplished solely from analysis of the target density distribution defined in Eq. 2.72, which provides an estimate of target density over the surveillance area. As such, all features must be defined to operate over a target density distribution, rather than a probability distribution.

In order to characterize regions, region features must be determined which are computed from the target density distribution. For example, consider a probability distribution. One feature of a probability distribution is the entropy. Entropy as a feature provides a measure of the amount of information that can be taken from the probability distribution. Entropy measures this information inversely, by providing the amount of uncertainty present in the probability distribution. Other standard features include the probability distribution mean and variance, as well as maxima and minima locations and values. Region features must capture the characteristics inherent in the target density distribution shape in order to extract regions characteristic of search, tracking, exploration, and target nullity. To understand what possible features could highlight the differences between each of these types of partitions, first consider an exploration partition. An exploration partition has no, or at least very little, information content. If a probability distribution were to describe an exploration partition, it would be a uniform distribution. In other words, such distributions have maximum entropy or uncertain. These partitions are separated from the other types of partitions because, in terms of information, all observing any one point in an exploration partition is equal to observing any other point in the same exploration partition. Next, consider the opposite of exploration partitions. This is a tracking partition. In a tracking partition there is so much information available that it can essentially be said that it is known where a target is located. As such, entropy-like features would aid in labeling regions in the surveillance area as tracking partitions. For the case of entropy, this would correspond to very low entropy values. Now consider search partitions. Search partitions are in between exploration and tracking partitions. Consequently most of the surveillance area will likely consist of search partitions. In fact, most of the partition classification work will be focused on search partitions because they are more difficult to characterize and distinguish from each other. For search partitions, more types of features are required for characterization, and this will be the bulk of the presentation later in this chapter.

In conjunction with determining a set of features that characterize regions within the surveillance area, a feature-based partition classificatier must be developed. A feature-based partition classificatier takes the pre-determined feature definitions and uses computed features to classify the surveillance area into partitions. This classifier operates directly on features computed from a single time instance of the target density distribution. This is done in consideration of distribution of intelligence in the network of autonomous agents. In this way, the estimation of the target density distribution can be done distributed over the network, as would be necessary. Then, given the target density distribution, each agent can then perform the required partition classification. The goal of the classifier is then to generate a set of partitions consisting of regions for search, tracking, exploration, and target nullity. And these partitions must be classified in a manner that enables routing of the autonomous agents to over the surveillance area.

It is not necessary to provide time association of the partitions generated from the partition classification. However, the partitions generated by the classifier will be more useful in the vehicle routing step of agent path planning if the labels of the partitions do not change

drastically in time. To understand this, view each partition as a search, tracking, or exploration task that must be completed. If the task identification of each partition were to change at each instant in time, then the agent routing problem would be made chaotic and difficult to manage. It is consequently best to incorporate time association of partitions. In fact, the partitions at time t should be seeded with the partitions at time $t-1$ in order to speed up classification.

As a final optional step, learning to predict the general motion of tracking partitions can be accomplished. Incorporation of this step requires assumptions on the targets such as that they will follow a specific path, or generally that they will be propelled by some structured mechanism. However, in this work, targets were assumed to move in any direction randomly (e.g., according to a diffusion model). Consequently, very little could be done to accurately predict the motion of targets other than to apply diffusion-like uncertainty. As such, this step may often not yield any useful prediction of motion. Yet, if there is some structure that can be assumed about target motion, then this step is very useful. Scenarios in which motion learning are very useful include when targets are on road systems and it is difficult for them to rapidly change direction. Learning tracking partition general motion is enabled through the step of time association because it provides a history of each tracking partition's trajectory.

Full characterization of the surveillance area by time-evolving partitions is then accomplished by developing methods for each of the steps listed above. The development presented in this work consists of the first two steps. The last two steps are accomplished through methods as found in the literature [34, 93]. As such, only the first two steps will be discussed in depth here.

3.5 Region Features

Consider seeking for regions within a target density distribution that contain points of similar characteristics. Repeatedly observed regions will either have very low expected number of targets or have high certainty while unsearched regions will have low certainty and apriori defined expected number of targets. Regions in which targets have been repeatedly detected will have a high expected number of targets while repeatedly observed regions in which no targets have been detected will have a low expected number of targets. Some regions will have high certainty and a high expected number of targets. Others will have low certainty and an unaltered expected number of targets. Yet others will have a low expected number of targets. The surveillance area can accordingly be partitioned into regions based on characteristic certainty and expected number of targets, as well as location within the surveillance area.

The target distribution can be analyzed spatially by considering features based on variations of underlying features corresponding to local uncertainty, local expected number of targets, and normalized position. Each of these underlying features are defined locally. To provide this local definition, each of these features is based on the local area $S_r(x_0) \subseteq S$ of

some point $x_0 \in S$ in the surveillance area (where S is the space of the surveillance area) defined as

$$S_r(x_0) := \{x \in S : d(x, x_0) < r\}, \tag{3.1}$$

where $d(x, x_0)$ is some measure of distance.

Local uncertainty is defined by first selecting a measure for uncertainty. In this work, local uncertainty is based on entropy. As such, local uncertainty is computed by evaluating the local entropy defined as

$$\mathrm{H}_r(x_0) := \int_{x \in S_r(x_0)} f_r(x, x_0) \log\left(\frac{1}{f_r(x, x_0)}\right) d\mu(x), \tag{3.2}$$

where $f_r(x, x_0)$ is the locally normalized target density function defined by

$$f_r(x, x_0) := \frac{f(x)}{\int_{\gamma \in S_r(x_0)} f(\gamma) d\mu(\gamma)}. \tag{3.3}$$

Note that if local uncertainty were not defined on a locally normalized target density distribution, then there would not be a well established maximum value for local uncertainty. A locally normalized density is then required so that the maximum value for local uncertainty can be referenced during classification. This maximum value allows the classifier to determine if a particular region contains significantly biased information of target density. To aid in understanding of local uncertainty, as defined in Eq. 3.2, Fig. 3.3 presents the local entropy of a simple Gaussian probability distribution.

Local expected number of targets comes readily from the target density distribution. To understand this, recall the definition of the target density distribution in Eq. 2.72. From this definition it is apparent that the expectation number of targets within some region A, is just the integration of the target density distribution over that region. Considering the definition of locality in Eq. 3.1, local expected number of targets is then computed by integrating the target density distribution over the local area as

$$\begin{aligned} \mathrm{I}_r(x_0) &:= \mathrm{E}N_{S_r(x_0)} \\ &= \int_{x \in S_r(x_0)} f(x) d\mu(x). \end{aligned} \tag{3.4}$$

To aid understanding of local expected number of targets, Fig. 3.4 presents a sample computation over a surveillance area for when the target density distribution is a simple Gaussian probability distribution.

The intuition behind each of these features is to imagine an area defined locally around some point in the surveillance space. Then imagine a target density distribution defined over the area local to this point. Each feature is then a description of that locally defined distribution. Notice that these features are designed to capture spatial characteristics. Features over time have not been considered in this work, but could possibly serve to help further characterize regions in the surveillance area.

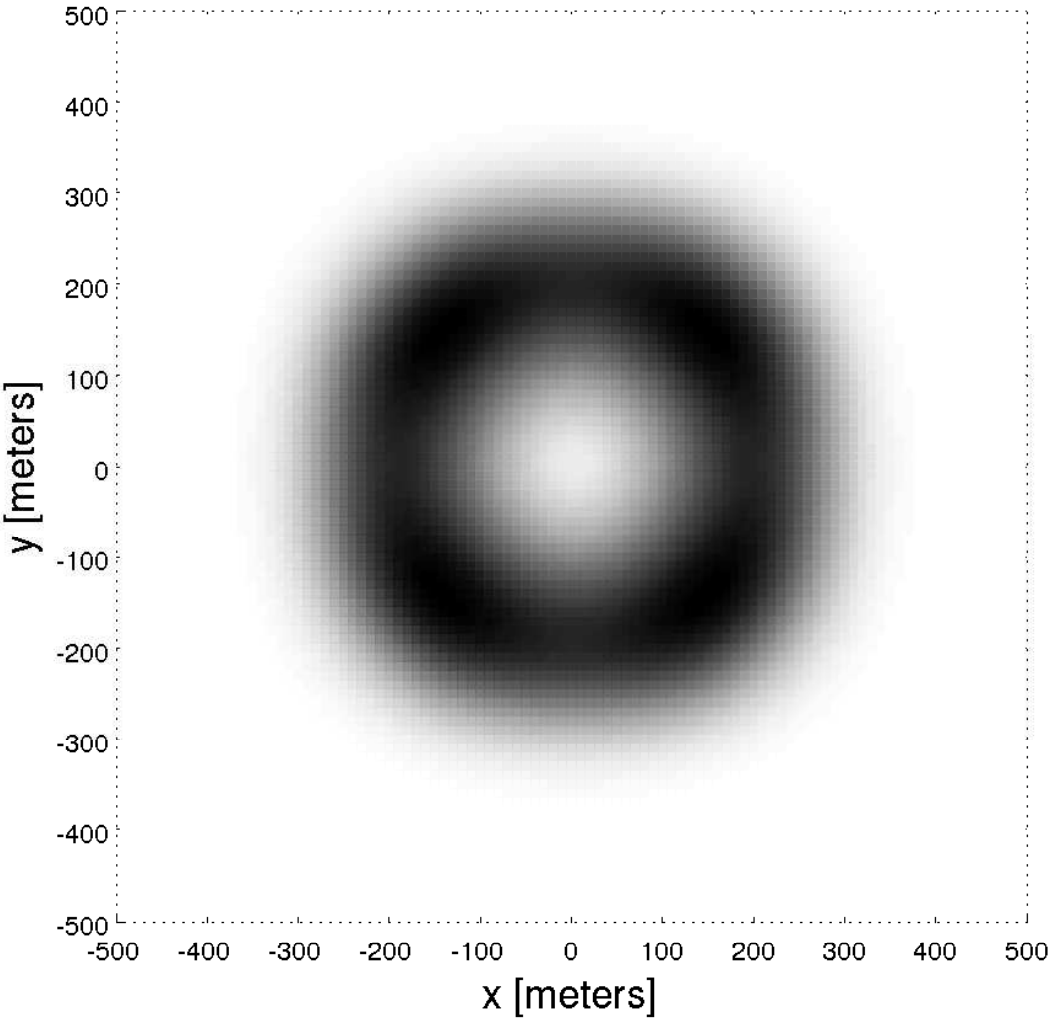


Figure 3.3: A sample computation of local uncertainty for the case when the target density distribution is a simple Gaussian probability distribution along with a uniform distribution.

3.6 Partition Classification

The partition classificatier analyzes the target density distribution and classifies regions within the surveillance area at each instance in time. The regions classified are of the types described above, which are search, tracking, exploration, and target nullity. A general picture of the process of the classifier can be summarized as

1. Construct a subset of the surveillance area by removing regions that correspond to

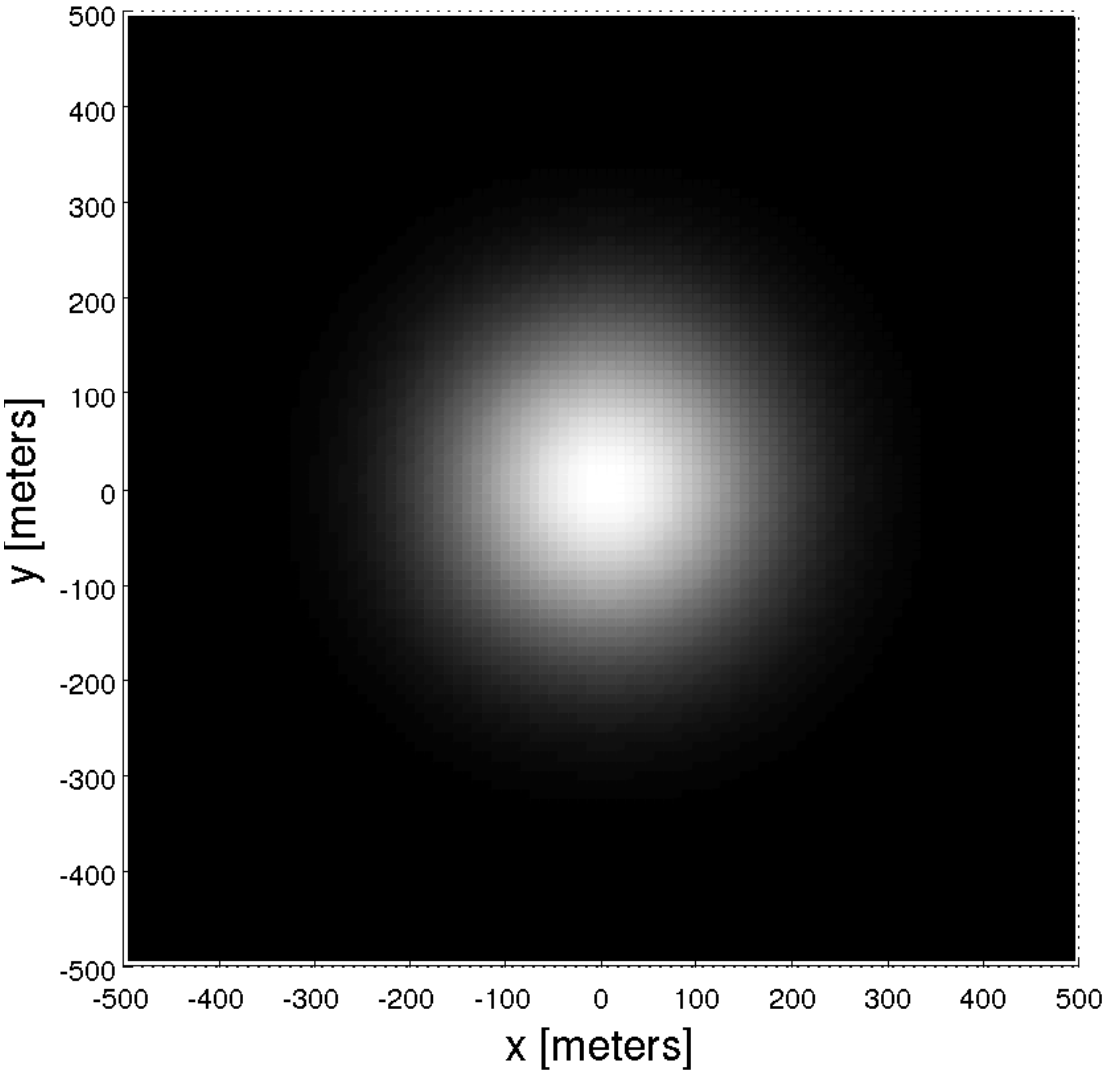


Figure 3.4: A sample computation of local expected number of targets I_r for the case when the target density distribution is a simple Gaussian probability distribution along with a uniform distribution.

exploration regions or target nullity regions.

2. Taking this subset of the surveillance area, perform classification within the target density distribution feature space. This classification defines the search and tracking partitions.

3. The result is then a partition for exploration, a partition for target nullity, and a set of partitions for search and tracking.

Before presenting the complete classifier to accomplish this, a classifier will be presented that partitions the surveillance area into every partition described above except for the null target partition. This classifier is called the foundational classifier. The reason why this foundational classifier is presented first is to highlight the main operations of the complete classifier and the role of the classifier in distinguishing between the necessary types of path planning for target search and tracking. The foundational search and tracking classifier will then be extended to account for the null target partition.

3.6.1 Foundational Search and Tracking Classifier

Fig. 3.5 presents an overview of the structure of the foundational search and tracking time-evolving partition classificatier which operates over the feature space of the target density distribution at each instance in time. The classification is accomplished by a cascade of classifiers, as can be seen in Fig. 3.5. The levels of the cascade classify points in the surveillance area by

1. Determining the search map, which divides the surveillance area into two regions. One region is labeled the exploration partition and the other region is designated for further partitioning (Fig. 3.5 block (1)).

2. Taking the region designated for further partitioning and classifying the points in this region into a set of search and tracking partitions (Fig. 3.5 block (2)).

3. Ordering this set of search and tracking partitions according to the expected number of targets within each partition (Fig. 3.5 block (3)).

4. Sub-partitioning the set of search and tracking targets to account for high values of expected number of targets that may exist in some partitions (Fig. 3.5 block (4)).

Each of these steps of the classifier will now be discussed in detail below. Sample pseudo-code is also presented. Pseudo-code for the computation performed in Fig. 3.5 block (1) is presented in Alg. 3.1. Pseudo-code for the computation performed in Fig. 3.5 block (2) is presented in Alg. 3.4. Pseudo-code for the computation performed in Fig. 3.5 block (3) is presented in Alg. 3.5. Pseudo-code for the computation performed in Fig. 3.5 block (4) is presented in Alg. 3.6.

Search Map Classification

The search map is the output of Fig. 3.5 block (1). A general picture of what occurs at this step of the classifier can be summarized as

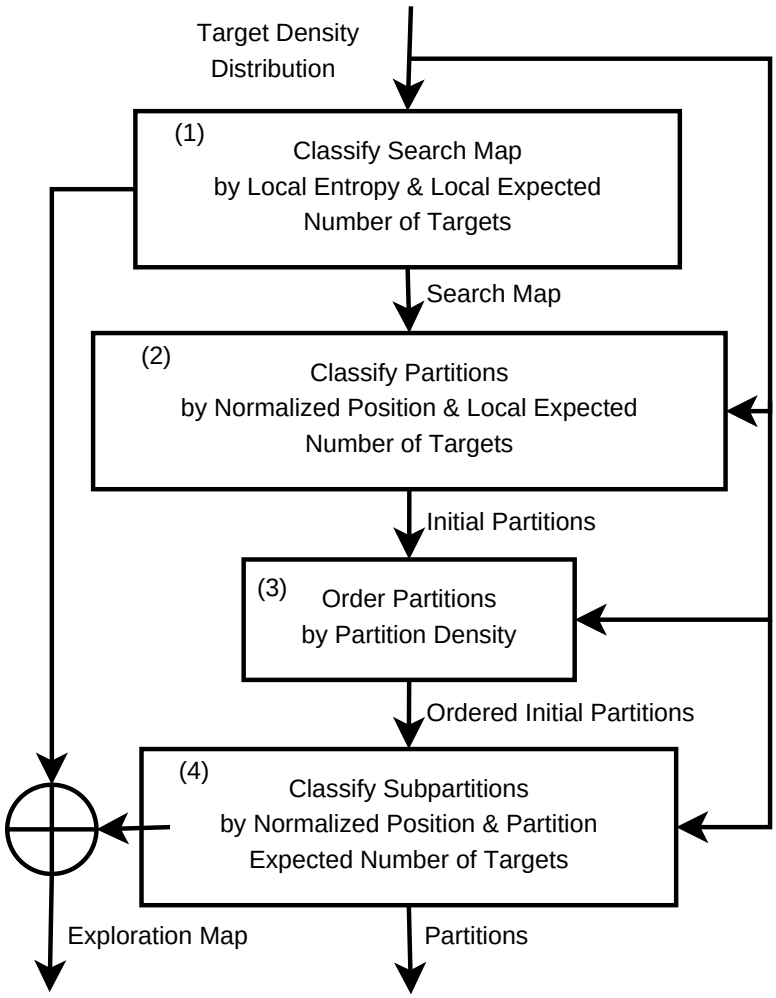


Figure 3.5: Structure of the cascade of classifiers for partitioning the surveillance area into an exploration partition and an ordered set of search and tracking partitions.

1. Determine the set of points in the surveillance area that belong to the exploration partition.
2. Construct a subset of the surveillance area by removing the exploration partition.
3. This subset is called the search map.
4. The search map is then passed on to subsequent steps of the classifier to be classified into search and tracking partitions.

Recall that the exploration partition is defined as regions that have little available information. This roughly corresponds to regions that have approximately locally uniform target

density. There are some subtleties, and these will be addressed next. Calling the output of Fig. 3.5 block (1) the search map then intuitively makes sense because it divides the surveillance area according to the amount of information available throughout the surveillance area. Regions with little information are labeled as the exploration partition and regions with sufficient information to guide a search are labeled as the search map.

Some subtleties involved in computing the search map include points in the surveillance area whose

- Local area has approximately uniform target density.
- Local information is not uniform but the target density has low value.

As described above, the features used in the classification are computed over local areas around each point in the surveillance area. The size of these local areas is the locality over which features are computed. So local uniformity depends on the size of the feature's locality used to measure local uniformity. If the feature's locality is very small, then almost any region in the surveillance area will appear locally uniform. So the size of a feature's locality necessarily becomes part of the design process. As a consequence of the size of a feature's locality, some regions within the surveillance area will be approximately locally uniform that should not be included in the exploration partition. For example, consider a Gaussian distribution with large variance. The region around the peak of the Gaussian distribution will likely be approximately locally uniform. However, this is the peak of the distribution. As such, it is the most likely region of target existence. So it should not be included in the exploration partition.

Another type of region consists of points in the surveillance area that have been observed and in which no targets have been detected. These regions should not be included in the region designated for search and tracking partitioning. In fact, these regions should be avoided until their target density increases. They should form the null target partition. But the null target partition will be discussed later. Often these regions will be surrounded by regions that do not have approximately locally uniform density. For example, consider a uniform target density distribution. When a no-detection observation is fused with target density distribution, a chunk of the target density is removed. In the center of this chunk the information is locally uniform. However, on the fringes of the chunk, there is a drastic change of information. This is caused by the transition from low target density to the prior target density. These fringes of the chunk have no special meaning and should still be regarded as the exploration partition.

Taking these subtleties into account, the computation of the search map will now be developed. Recall that this step of the classifier essentially divides the surveillance area into one region that has enough information to guide a search and another region that has little information. To determine these two regions, three steps are taken. These steps are

1. Computation of the high entropy map.

2. Computation of the low density map.

3. Combination of these maps to form the search map.

These steps will now be presented.

To determine which regions have little information (or are approximately locally uniform), a map is constructed that encodes regions of high local entropy. This map is called the high entropy map, which is defined as

$$S_{HE} := \{x \in S :\mid \mathrm{H}_r(x) - \mathrm{H}_{max} \mid < \epsilon\}, \tag{3.5}$$

where $\mathrm{H}_r(x)$ is the local entropy feature as defined in Eq. 3.2 and H_{max} is the maximum local entropy possible defined as

$$\mathrm{H}_{max} := \log \mid S_r \mid, \tag{3.6}$$

where $\mid S_r \mid := \int_{x \in S_r} d\mu(x)$. Take, for example, the case when the target density distribution is discretized on a fixed grid defined over the surveillance area. Recall the definition of S_r in Eq. 3.1. In Eq. 3.1, note that in order to define S_r, it is required to define a measure of distance $d(x, x_0)$ between two points x and x_0 in the surveillance area. For a fixed grid, this distance is defined as

$$d(x, x_0) := \max\left(\{\mid x(1) - x_0(1) \mid, \mid x(2) - x_0(2) \mid\}\right), \tag{3.7}$$

where the numbers 1 and 2 specify the indices of the points. Then, according to the definition of S_r, the maximum local entropy is $\mathrm{H}_{max} = \log N^2$, where N is the number of rows/columns in the square local area S_r.

With the high entropy map defined, the low density map will now be defined. In order to develop the definition of the low density map, the construction of a typical prior target density distribution will be discussed. At the start of a search and tracking mission, the target density distribution is biased by prior knowledge of where targets may exist. This prior knowledge consists of

- A prior distribution $f_{\text{prior}}(x)$ of previously known target positions.
- An estimated number of additional targets $\mathrm{E}N_{\text{additional}}$ that may exist in the surveillance area.

The prior target density distribution $f_{\text{prior}}(x)$ provides a target density bias based on where targets have most recently been observed and where they might be now. For example, f_{prior} may consist of a summation of Gaussian distributions, with each Gaussian representing the possible location of a particular target whose position was once known or whose position is

simply guessed. The estimated number of additional targets $\mathrm{E}\, N_{\text{additional}}$ affects the initial target density distribution by defining a uniform target density distribution $U(x)$ such that

$$\int_{x\in S} U(x)d\mu(x) = \frac{\mathrm{E}\, N_{additional}}{\int_{x\in S} d\mu(x)}. \tag{3.8}$$

The resultant prior target density distribution is then

$$f(x) = f_{\text{prior}}(x) + U(x). \tag{3.9}$$

From the discussion of a typical prior target density distribution above, the definition of the low density map can now be presented. From Eq. 3.9, note that some regions of the prior target density distribution will have a characteristic uniform value U. This characteristic low information value can then be used to define the low density map. As observations are made over the surveillance area, the target density distribution in observed regions with no detected targets will drop below the characteristic uniform distribution value. These areas can be defined as

$$S_{NI} := \{x \in S : f(x) < U + \epsilon\}. \tag{3.10}$$

This is the low density map. Now, combining the low density map with the high entropy map, the search map S_{search} is then defined as

$$S_{\text{search}} := S \setminus (S_{HE} \cap S_{NI}). \tag{3.11}$$

Defining the search map in this manner effectively addresses the subtleties mentioned above. Alg. 3.1 presents a sample pseudo-code that computes the search map as well as the exploration map, given a set of points in the surveillance area. Fig. 3.6 shows a sample search map along with its corresponding target density distribution. In the figure, black areas represent the search map and white areas represent areas left for exploration. Note that this search map was based on a prior distribution for a surveillance mission in which some knowledge was given of eight existing targets and it was estimated that there were two additional targets with completely unknown position.

Search and Tracking Partition Classification

The search map, which specifies the region of the surveillance area designated for search and tracking partitioning, is then passed on for further classification as shown in Fig. 3.5. At this level of the classifier (Fig. 3.5 block (2)), classification is performed convexly within the feature space consisting of local expected number of targets $\mathrm{I}_r(x)$ as defined in Eq. 3.4 and normalized position. There are many convex classification methods [34] that would work for this level of the classifier. The approach taken in this work is to perform classification by utilizing both K-means and Gaussian Mixture Model EM [34]. K-means is used to initialize search and tracking partitioning at the beginning of the search and tracking mission. Sample

Algorithm 3.1 Function computeSearchMap

Input: pointSet, features, subwindowSize, uniformValue
Output: searchMap, explorationMap

```
{pointSet: set of points in surveillance area to compute search map.}
{features: computed features for the points in the surveillance area consisting of normalizedPosition, expectedNumber (expected number of targets), and localEntropy.}
{subwindowSize: (convolution mask size - 1)/2.}
{uniformValue: characteristic value of low information density value.}

{Find the high entropy map.}
localEntropyMax ← max(features.localEntropy)
uniformEntropy ← log((subwindowSize * 2 + 1)^2)
highEntropyMaxIntegral ← −1
for x = pointSet.begin → pointSet.end do
  if |features.localEntropy(x) − localEntropyMax| < 0.01 then
    highEntropyMap.addToSet(x)
    if features.expectedNumberTargets(x) > highEntropyMaxIntegral then
      highEntropyMaxIntegral ← features.expectedNumber(x)
    end if
  end if
end for

{Classify points as exploration map or search map.}
for x = pointSet.begin → pointSet.end do
  if x ∈ highEntropyMap ∩ features.expectedNumber(x) < uniformValue + 0.01 then
    explorationMap.addToSet(x)
  else
    searchMap.addToSet(x)
  end if
end for

return searchMap, explorationMap
```

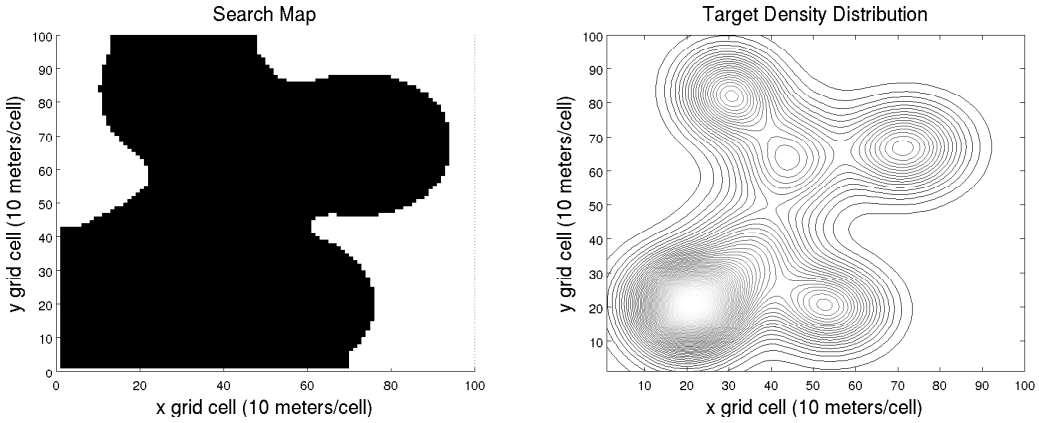


Figure 3.6: The search map (left) computed from a sample target density distribution (right). In the search map plot, white areas correspond to exploration space and black areas correspond to space that will be partitioned further.

K-Means pseudo-code is presented in Alg. 3.2. Gaussian Mixture Model EM is then used at subsequent steps in time where the Gaussian Mixture Model is seeded with previous partition means. Sample Gaussian Mixture Model EM pseudo-code is presented in Alg. 3.3. The number of partitions used in this level of the classifier is initialized by the total expected number of targets in the surveillance area:

$$\begin{aligned} N_{\text{initial}} &:= \text{ceil}\left(\mathrm{E}\, N_S\right) \\ &= \text{ceil}\left(\int_{x \in S} f\left(x\right) d\mu(x)\right) \end{aligned} \tag{3.12}$$

Alg. 3.4 presents a sample pseudo-code of the computation involved in this step of the classifier. Notice that Alg. 3.4 calls either a K-Means classifier or a Gaussian Mixture Model EM classifier to generate the partitions. Fig. 3.7 presents a sample partitioning based on the search map in Fig. 3.6. In Fig. 3.7, variations in color are used to represent the partitions of the surveillance area.

Algorithm 3.2 Function classifyKMeans

Input: pointSet,features,N_{part}
Output: labels, parameters
{Randomize initial means and labels.}
for $i = 1 \rightarrow N_{part}$ **do**
parameters.means[i].x = rand$*(x_{max} - x_{min}) + x_{min}$
parameters.means[i].y = rand$*(y_{max} - y_{min}) + y_{min}$
parameters.means[i].expectedNumber = rand$*(EN_{max} - EN_{min}) + EN_{min}$
end for
zero(numberElements)
for x = pointSet.begin $\rightarrow$ pointSet.end **do**
labels[x] $\leftarrow$ ceil(rand$*N_{part}$)
numberElements[labels[x]] $\leftarrow$ numberElements[labels[x]] + 1
end for
{Run K-Means.}
for $k = 1 \rightarrow N_{kmeans}$ **do**
zero(meanSums)
for x = pointSet.begin $\rightarrow$ pointSet.end **do**
meanSums[labels[x]] $\leftarrow$ meanSums[labels[x]] + features[x]
end for
for $i = 1 \rightarrow N_{part}$ **do**
parameters.means[i] $\leftarrow$ parameters.means[i]/$numberElements[i]$
end for
for x = pointSet.begin $\rightarrow$ pointSet.end **do**
distToMean $\leftarrow$ 10000000
for $i = 1 \rightarrow N_{part}$ **do**
dC $\leftarrow$ $\|$features(x) $-$ parameters.means[i]$\|$
if dC $<$ distToMean **then**
assignedCluster $\leftarrow$ i
distToCluster $\leftarrow$ dC
end if
end for
labels[x] $\leftarrow$ assignedCluster
delta $\leftarrow$ features[x] - parameters.means[assignedCluster]
squaredErrors[assignedCluster] $\leftarrow$ squaredErrors[assignedCluster] + delta*delta'
numElements[assignedCluster] $\leftarrow$ numElements[assignedCluster] + 1
end for
for $i = 1 \rightarrow N_{part}$ **do**
parameters.covariances[i] $\leftarrow$ 1/numberElements[i]$*$squaredErrors[i]
end for
end for
return labels, parameters

Algorithm 3.3 Function classifyGMM

Input: pointSet, features, N_{part}, parameters
Output: labels, parameters

```
initialize(weights,1/N_part)
for k = 1 → N_gmm do
  {Expectation step.}
  for x = pointSet.begin → pointSet.end do
    sumw ← 0
    for i = 1 → N_part do
      w[x,i] ← weights(i)*Gaussian(features[x],parameters.means[i],parameters.covariances[i])
      sumw ← sumw + w[x,i]
    end for
    for i = 1 → N_part do
      w[x,i] ← w[x,i]/sumw
    end for
  end for
  {Maximization step.}
  for i = 1 → N_part do
    sumwi ← 0; zero(sumwx); zero(sumwdxdx)
    for x = pointSet.begin → pointSet.end do
      sumwi ← sumwi + w[x,i]
      sumwx ← sumwx + w[x,i]*features[x]
      sumwdxdx ← sumwdxdx + w[x,i]*(features[x]-parameters.means[i]) * (features[x]-
      parameters.means[i])^T
    end for
    weights[i] ← sumwi/pointSet.size
    parameters.means[i] ← sumwx/sumwi
    parameters.covariances[i] ← sumwdxdx/sumwi
  end for
end for
for x = pointSet.begin → pointSet.end do
  distToMean ← 10000000
  for i = 1 → N_part do
    dC ← ||features[x] − parameters.means[i]||
    if dC < distToMean then
      assignedCluster ← i
      distToCluster ← dC
    end if
  end for
  labels[x] ← assignedCluster
end for
return labels, parameters
```

Algorithm 3.4 Function classifyPartitions

Input: searchMap, features, N_{part}, parameters
Output: partitions, newParameters

{searchMap: set of points in surveillance area to classify partitions.}
{features: computed features for the points in the surveillance area consisting of normalizedPosition, expectedNumber (expected number of targets), and localEntropy.}
{N_{part}: desired number of partitions to classify.}
{parameters: means and covariances to seed the partition classification. These may be empty (at program start) or the parameters of the previous partitions.}

{Check seed partition parameters.}
if isEmpty(parameters) **then**
  {Use K-Means algorithm.}
  [labels,newParameters] ← classifyKMeans(searchMap,features,N_{part})
else
  {Seed Gaussian Mixture Model EM algorithm.}
  [labels,parameters] ← classifyGMM(searchMap,features,N_{part},parameters)
end if

{Construct the search and tracking partitions.}
for $i = 1 \rightarrow$ labels.size **do**
  $x \leftarrow$ searchMap[i]
  $l \leftarrow$ labels[i]
  partitions[l].addToSet(x)
end for

return partitions, parameters

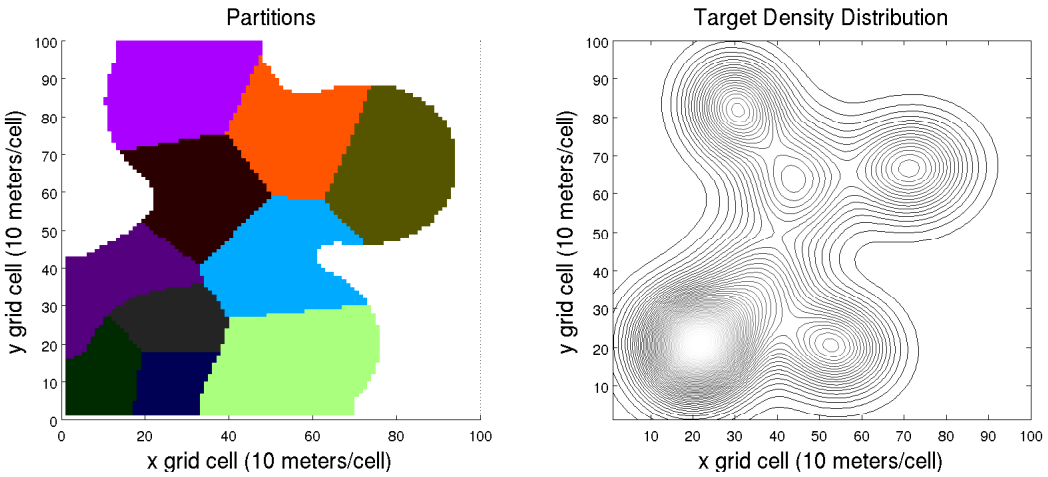


Figure 3.7: The search and tracking partitions (left) computed from a sample target density distribution (right). Partitions are represented by variations in color over the surveillance area

Ordering of Search and Tracking Partitions

After the search map is partitioned into search and tracking partitions, these partitions are then ordered (Fig. 3.5 block (3)). This ordering is done to account for priority in the partitions as well as to prepare for the final level of the classifier in which the partitions are sub-partitioned. This ordering is simply a rearrangement of the partitions in descending order of partition density ρ_{P_i} defined as

$$\rho_{P_i} := \frac{\int_{x \in P_i} f(x) d\mu(x)}{\int_{x \in P_i} d\mu(x)}, \tag{3.13}$$

where P_i is the ith partition. Alg. 3.5 presents a sample pseudo-code for performing the ordering of the search and tracking partitions.

Sub-partitioning of Search and Tracking Partitions

The final level of the classifier is that of taking the ordered search and tracking partitions and then possibly sub-partitioning them (Fig. 3.5 block (4)). This step of sub-partitioning is not necessary. However, it serves to further break up important partitions, thus allocating more resource to them. One case in which sub-partitioning occurs is when some targets are very close to each other. To detect such a case, first define the expected number of targets of partition P_i to be

$$\mathrm{E}\, N_{P_i} := \int_{x \in P_i} f(x) d\mu(x). \tag{3.14}$$

Algorithm 3.5 Function orderPartitions

Input: partitions, features
Output: orderedPartitions
{partitions: search and tracking partition sets consisting of points in search map.}
{features: computed features for the points in the surveillance area consisting of normalizedPosition, expectedNumber (expected number of targets), and localEntropy.}

{Get the partition densities.}
for $i = 1 \rightarrow$ partitions.size **do**
$N \leftarrow 0$
expectedNumber $\leftarrow 0$
for $x =$ partitions[i].begin $\rightarrow$ partitions[i].end **do**
$N \leftarrow N + 1$
expectedNumber $\leftarrow$ expectedNumber + features.expectedNumber(x)
end for
partitionDensities.addToSet(expectedNumber/N)
end for

{Get the new order of the partitions.}
tmpPartitionDensities $\leftarrow$ partitionDensities
for $i = 1 \rightarrow$ partitions.size **do**
[maxVal, maxIndex] $\leftarrow$ max(tmpPartitionDensities)
partitionOrder[i] $\leftarrow$ maxIndex
tmpPartitionDensities[i] $\leftarrow -1$
end for

{Construct the partitions in the new order.}
for N_{part} =partitionOrder **do**
orderedPartitions.addToSet(partitions[Npart])
end for

return orderedPartitions

Then note that there will likely be partitions with expected number of targets greater than 1. The approach taken in this work is to sub-partition any partition P_i that satisfies $\mathrm{E}\, N_{P_i} > 1$ into ceil($\mathrm{E}\, N_{P_i}$) new partitions. In this work, sub-partitioning is accomplished utilizing a Gaussian Mixture Model like above, except the feature space consists only of normalized position.

This step may result in significantly more partitions than there initially were after the search and tracking partitioning level of the classifier. To avoid an excessive number of partitions, a cutoff number of partitions $N_{\max}$ is specified. The final partitions are then constructed in a loop that runs over the ordered partitions. At each step in the loop an ordered partition is checked to see if it should be sub-partitioned. If it requires sub-partitioning, then it is sub-partitioned and each new partition is added to the final set of partitions. If the partition requires no sub-partitioning, it is simply added to the final set of partitions. If the number of final partitions reaches $N_{\max}$, the loop over the ordered partitions breaks. Consequently some of the lower priority ordered partitions may not be included in the final search and tracking partitions. These partitions are then added to the exploration partition, as can be seen in Fig. 3.5. Sample pseudo-code of this level of the classifier is presented in Alg. 3.6.

Now, with the final level of the classifier defined, the foundational partition classificatier has been established. Utilizing this foundational partition classificatier, search and tracking partitioning can be performed. Using this classifier, a sample sequence of the partition classificatier is shown in Fig. 3.9. A description of the objects plotted in Fig. 3.9 is presented in Fig. 3.8.

Algorithm 3.6 Function classifySubPartitions

Input: partitions, density, features, N_{max}, explorationMap
Output: newPartitions, explorationMap

{Inputs: search and tracking partition sets, target density distribution, computed features for the points in the surveillance area, soft maximum number of partitions.}

{Get the partition level expected number of targets.}
for p = partitions.begin $\rightarrow$ partitions.end **do**
  $EN \leftarrow 0$
  for $x = p$.begin $\rightarrow p$.end **do**
    $EN \leftarrow EN$ + density[x]
  end for
  EN_{part}.addToSet(EN)
end for
{Loop through to check for needed sub-partitioning.}
$N_{part} \leftarrow 0$
for $i = 1 \rightarrow$ partitions.size **do**
  if $N_{part} \geq N_{max}$ **then**
    thisSum $\leftarrow EN_{part}[i]$
    numRepartitions $\leftarrow 0$
    if thisSum > 1.25 **then**
      numRepartitions $\leftarrow$ ceil(thisSum)
      [labels,parameters] $\leftarrow$ classifyKMeans(partitions[i],features,numRepartitions)
      for $j = 1 \rightarrow$ labels.size **do**
        $x \leftarrow$ partitions[i][j]
        $l \leftarrow$ labels[j]
        tmpPartitions[l].addToSet(x)
      end for
      for p = tmpPartitions.begin $\rightarrow$ tmpPartitions.end **do**
        newPartitions.addToSet(p)
        $N_{part} \leftarrow N_{part} + 1$
      end for
    else
      newPartitions.addToSet(partitions[i])
      $N_{part} \leftarrow N_{part} + 1$
    end if
  else
    for x = partitions[i].begin $\rightarrow$ partitions[i].end **do**
      explorationMap.addToSet(x)
    end for
  end if
end for
return newPartitions, explorationMap

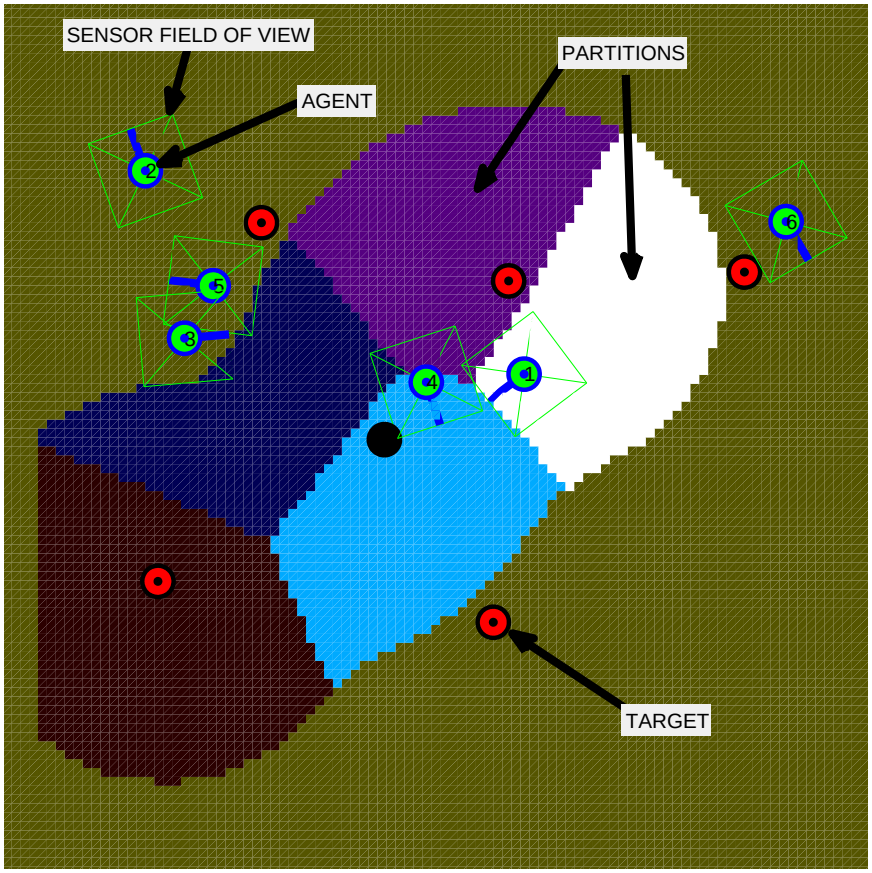


Figure 3.8: Description of the objects presented in the sample sequence of time-evolving partition classification path planning. Agents are represented as circles with protruding lines. Sensor fields of view are represented as additional thin lines around the agents. Targets are represented as circles without protruding lines. Partitions are represented by variations in color across the search area.

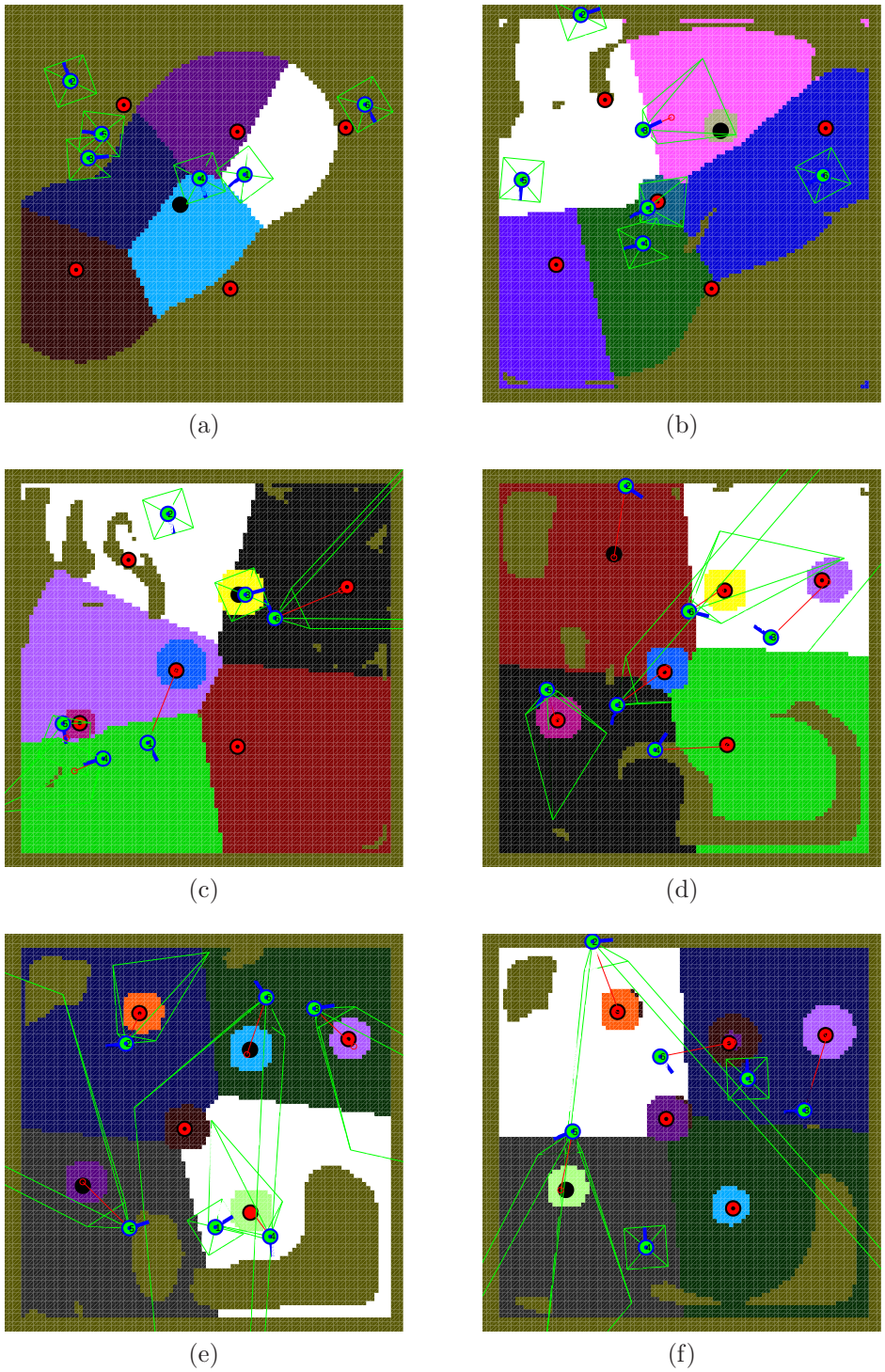


Figure 3.9: Sample sequence of partition classification for a scenario involving six agents and six targets (the number of targets was unknown to the agents). Follow the sequence from left to right and from top to bottom. In (a) initial partitions are formed. In (b) tracking partitions appear. By (f) all targets are within tracking partitions.

3.6.2 Complete Classifier

The complete classifier is an easy extension of the foundational classifier. All that is required is a step in the classification algorithm that classifies regions in the surveillance area into the null target partition. The null target partition is meant to contain regions in the surveillance area that have been searched and, while considering possible target motion, contain essentially no targets. To determine these regions it is then required to specify a target nullity threshold ϵ_{null}. With this target nullity threshold given, all points in the surveillance area that correspond to regions of essentially no targets can then be defined by

$$S_{\text{null}} := \{x \in S : f(x) < \epsilon_{\text{null}}\}. \tag{3.15}$$

Including this in the classifier is an easy extension because it is best incorporated into the classifier algorithm at the very beginning as another level in the cascade. The null target block then output the set of points in the surveillance area that are not in the null target partition. This set, instead of the complete surveillance area, is then passed to the search map classifier, and the classification continues in the manner as detailed above. The corresponding new classifier structure is presented in Fig. 3.10.

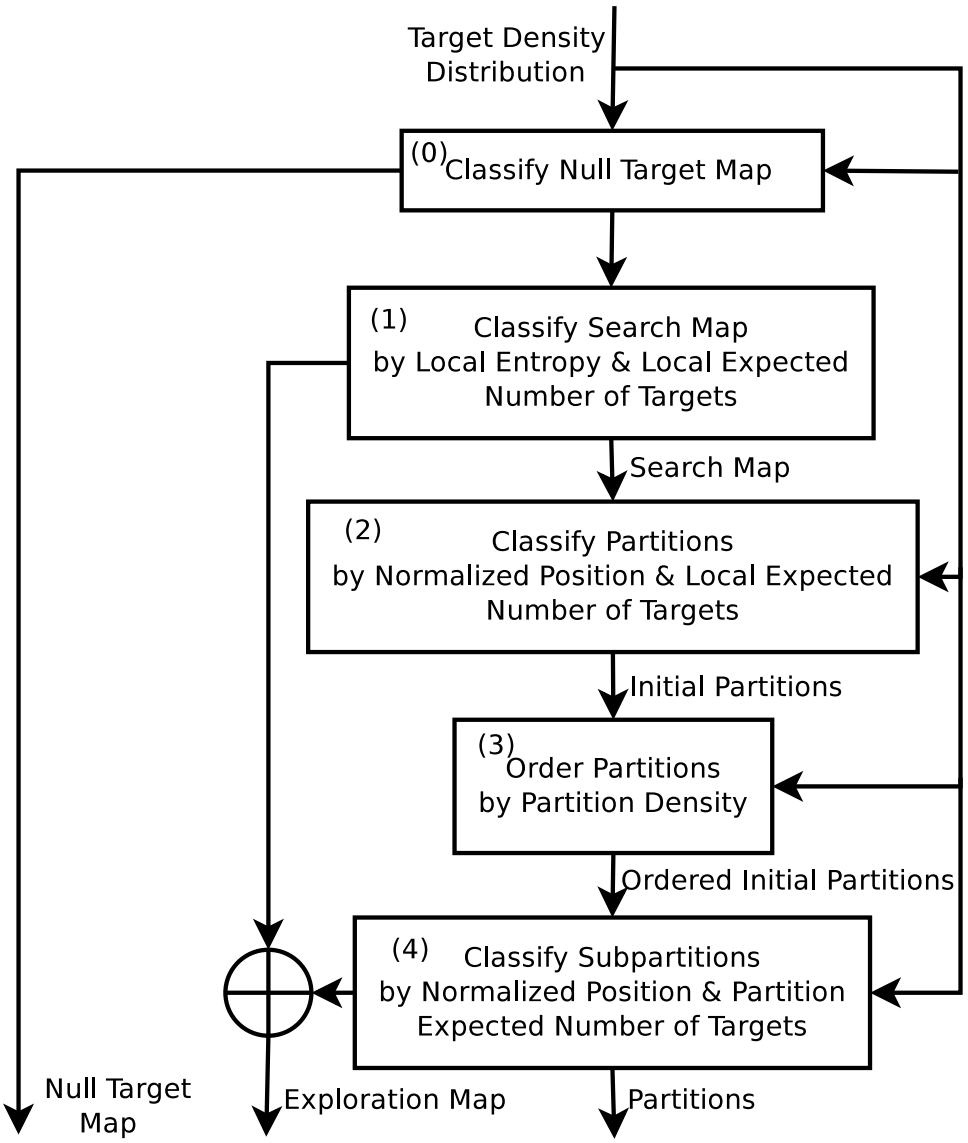


Figure 3.10: The complete partition classificatier, including determination of the null target partition. Observe that this classifier consists of a cascade of simple feature-based classifiers. At the top, regions of target nullity are determined. Then the bulk of the exploration partition is extracted. Next, search and tracking partitions are classified.

3.7 Summary

The focus in target search and tracking can be shifted from being target-based to being region-based. This is done to account for the complexities involved in area surveillance when

there is an unknown number of targets. The type of area surveillance assumed in this work is that of a very large area over which observations can be made, but each observation has a field of view much smaller than the size of the surveillance area.

The information set defined over the surveillance area will in general be very complex. A sample target density distribution illustrating this complexity of information set was presented in Fig. 3.1. By shifting the focus to region-based target search and tracking, the goal is to break the surveillance area up into characteristic partitions over which simple information sets can be defined. Then the autonomous agents can search over the partitions utilizing path planning strategies designed for specific simple information sets.

The algorithm designed to accomplish partition classification is presented in Fig. 3.10. Notice from this figure that the classifier is actually composed of a cascade of simple classifiers. In order of algorithmic execution, these levels of the cascade are

1. Null target map classification.

2. Search map classification.

3. Search and tracking partition classification.

After running this algorithm, the surveillance area is partitioned into one partition for target nullity, one partition for exploration, and an ordered set of partitions for search and tracking. The information set over these partitions is then simple enough to form tasks and have a team of autonomous agents routed over the tasks.

Chapter 4

Path Planning Over Search and Tracking Partitions

4.1 Path Planning Over Partitions

Recall Fig. 1.1. After estimating a target density distribution (block (1) in Fig. 1.1), the target density distribution is analyzed and the surveillance area is partitioned into a partition for exploration and an ordered set of partitions for search and tracking (block (2) of Fig. 1.1). Once the partitions are computed, agent path planning is then performed (block (3) of Fig. 1.1). In this work, the path planning is accomplished by separating the planning into two layers as depicted in Fig. 4.1. The layers of the planning are

1. Partitions are considered tasks and allocated to the team of agents (block (1) of Fig. 4.1).

2. Partition level target density distribution based path optimization (block (2) of Fig. 4.1).

4.2 Task Allocation Based Route Planning Over Partitions

The partitions generated by the classifier define regions over which subsets of the target density distribution can be extracted. This partitioning of the target density distribution into disjoint regions of exploration, search, and tracking suggests the application of some kind of task allocation algorithm that takes each of the partitioned regions as tasks with varying level of certainty or priority (based on the partition shape and partition level target density distribution). Methods that have been developed [1, 40] rely on the ability to evaluate the cost to accomplish each task. These task costs are utilized to optimize the agent routes through the tasks, based on the position and capabilities of each agent.

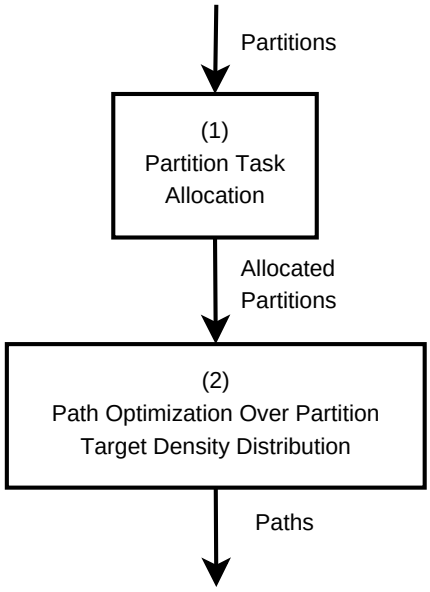


Figure 4.1: Structure of path planning separated into task allocation over search partitions and path planning by optimizing directly over the target density distribution.

4.2.1 Cost Functions

In this work it is assumed that appropriate task allocation or vehicle routing algorithms have been taken directly from the literature or that they have already been designed by some outside source. As such, no work has been done within the field of task allocation or vehicle routing. Consequently, very little will be mentioned regarding how to design or what is optimized in vehicle routing algorithms. However, it is worthwhile to briefly mention the role of cost functions in typical vehicle routing algorithms. Cost functions are what are provided to the vehicle routing algorithm to evaluate cost to visit between each task. So to design a vehicle routing algorithm that takes partitions as tasks, costs to visit each of these partitions must be defined. As an example, the time to visit a task is often utilized as the task cost [40, 41]. Unfortunately, heuristic functions are often required to specify the time required to visit complex tasks such as search and tracking partitions. For example, how would a time of task visitation be assigned for an abnormally shaped search partition? Furthermore, it is also unclear what defines the ending point of a search through a partition. However, the measure of time for task visitation does not have to be exact. It's only needed to provide some relative difference between visiting different tasks.

4.3 Target Density Based Planning Over a Set of Partitions

As agents are allocated to partitions, their paths within the partitions are determined by optimizing directly over the partition level target density distribution. Approaches have been developed to optimize agent paths directly over a target density distribution [69]. These methods seek to maximize the number of expected targets in the target density distribution, which is one type of search strategy. However, considering the diversity of search strategies that have been developed for agent path optimization over probability distributions [18, 21, 43, 71, 81, 94, 105, 83], the approach taken for target density distribution optimization extends these approaches. However, these approaches require generalization for use with target density distributions. Before presenting this generalization, approaches based on probability distributions will be reviewed as well as modifications [105] presented. Then, the generalization will be presented.

4.3.1 Probability Distribution Based Path Optimization

Before plowing ahead and determining ways of optimizing paths over a target density distribution, the case of optimization over a probability distribution will be discussed here. To do this, one must question what it means to optimize over a probability distribution. Let there be some a-priori defined probability distribution $P_0(x)$ over the random variable x. Applied to the topic of interest, x can be the position of a target, which can take on values for position within the surveillance area $x \in S$. $P_0(x)$ is usually called the prior distribution. To optimize over this distribution implies that there is some operation $L(x)$ that transforms $P_0(x)$ to a new distribution $P_1(x)$, and that some structure of $L(x)$ can be chosen so that the expected quality of $P_1(x)$ is greater than the quality of $P_0(x)$.

There are various measures of quality and various approaches toward designing the structure of $L(x)$ to maximize quality. These measures of quality and optimization approaches will be presented by going through the following topics.

1. Information-theoretic optimization.
2. Value function optimization.
3. Receding horizon optimization.
4. Optimal path planning.
5. Accounting for shortcomings in optimization approach.

Information-theoretic Optimization

The purpose of a probability distribution $P(x)$ is to provide information on the probable value of some random variable x. This information is provided by $P(x)$ defining probabilities over the possible values of x. Any number of probability distributions can be used and each one provides information on x. However, the amount of information provided by each distribution is not equal. Consider two probability distributions. The first is a distribution with most probability localized around some value x_0 of x. The second is a distribution with equal probability over every possible value of x. With the second distribution, there is high uncertainty in the probable value of x because any value of x is equally probable. However, in the first distribution it can be concluded with high certainty that the value of x will likely be x_0. Intuitively the first distribution provides more information on the value of x. Information is then opposed to uncertainty. As the uncertainty decreases, the information increases. The most common information measure is provided by entropy, which is actually a measure of uncertainty. Entropy is based on the minimum description length by introducing randomness and taking expectation [85]. Entropy is then defined as

$$\begin{aligned} H(x) &:= \mathrm{E} \log \left(\frac{1}{P(x)} \right) \\ &= \int_{x \in S} P(x) \log \left(\frac{1}{P(x)} \right) d\mu(x) \\ &= - \int_{x \in S} P(x) \log \left(P(x) \right) d\mu(x), \end{aligned} \tag{4.1}$$

where the notation $H(x)$ is shorthand for $H(P(x))$. $H(P(x))$ is used whenever it is needed for clarity when there may be multiple distributions over x. Using entropy, information $J(x)$ can then be defined as $J(x) := -H(x)$.

Utilizing some measure $J(x)$ of information provided by a distribution over x, it is immediate that if for $P_1(x)$ to have more information than $P_0(x)$, it is necessary that

$$J_1(x) > J_0(x), \tag{4.2}$$

where $J_i(x)$ represents the information provided by $P_i(x)$. In order to to maximize the increase in information between $P_1(x)$ and $P_0(x)$, an information-theoretic approach is then defined as [68]

$$\begin{aligned} &\max \Delta J(x) \\ &= \max J_1(x) - J_0(x), \end{aligned} \tag{4.3}$$

which is maximization of the change in information ΔJ. This fully specifies the general objective of information-theoretic optimization. However, there are further generalizations of this objective to provide more versatility in optimization. To develop these generalizations,

first recall the structure of the process in which $P_0(x)$ transforms to $P_1(x)$. Considering that $P_0(x)$ and $P_1(x)$ are probability distributions with the same measure $\mu(x)$, in general this operation is just a weighting of each point in the space of x. This process can then be fully represented by point-wise weighting, followed by any necessary normalization in order to preserve $P_1(x)$ as a probability distribution. This process is mathematically described as

$$P_1(x) = \frac{1}{N} L(x) P_0(x), \tag{4.4}$$

where

$$N = \int_{x \in S} L(x) P_0(x) d\mu(x). \tag{4.5}$$

However, further structure can be given to $L(x)$. $L(x)$ weights each value of x, so it is a function defined over the space of x. Yet it also has randomness. This randomness is incorporated by inclusion of a random variable z. A particular instantiation of z is called an observation because it is dependent on the *true* value of x. In this light, this weighting function is referred to as a likelihood function and can be viewed as some function which specifies the likelihood of the true value of x given a particular observation $z = \bar{z}$. A likelihood function then has the form $L(x; z)$, which suggests the interpretation of it being a weighting function as parametrized by the observation. It can also be viewed as being similar to a conditional probability distribution $P(z|x)$ but with x being the variable. The process of $P_0(x)$ transforming to $P_1(x)$ is then just the conditional probability of x given z. $P_0(x)$ can then be simply referred to as $P(x)$ (the distribution before any observation is made). So $P_1(x)$ is then

$$\begin{aligned} P_1(x) &= P(x|z) \\ &= \frac{1}{N} L(x; z) P_0(x) \\ &= \frac{1}{N} L(x; z) P(x), \end{aligned} \tag{4.6}$$

where

$$\begin{aligned} N &= P(z) \\ &= \int_{x \in S} L(x; z) P(x) d\mu(x). \end{aligned} \tag{4.7}$$

Recall that in the transformation from $P_0(x)$ to $P_1(x)$ it was assumed that some structure of $L(x; z)$ could be deterministically chosen. This structure is incorporated into the likelihood function by further parametrizing deterministically. For example, in the application of interest in this work, the position of an agent as well as the type of sensor, including its properties and orientation, serve as deterministic parameters that specify some structure of the likelihood function generated by observations. These deterministic parameters are

incorporated by inclusion of the parameter vector η for the likelihood function $L(x; z, \eta)$. All of the transformation's structure is now fully specified.

Utilizing this additional structure, the information-theoretic optimization problem can be further developed considering $P_1(x) = P(x|z)$ and $P_0(x) = P(x)$. The increase in information can now be written as

$$\begin{aligned} &\max \Delta J(x) \\ &= \max J(P(x|z)) - J(P(x)). \end{aligned} \tag{4.8}$$

Recalling the most common measure of information, $J(P(x)) = -H(x)$, maximization of the change in information becomes

$$\begin{aligned} &\max \Delta J(x) \\ &= \max -H(P(x|z)) + H(P(x)) \\ &= \max -\mathrm{E} \log \left(\frac{1}{P(x|z)} \right) + \mathrm{E} \log \left(\frac{1}{P(x)} \right) \\ &= \max \mathrm{E} \log \left(P(x|z) \right) - \mathrm{E} \log \left(P(x) \right). \end{aligned} \tag{4.9}$$

This fully defines the optimization problem posed as maximization of the change in information for information defined as $J(x) = -H(x)$. However, note that ΔJ is actually a divergence between $P(x|z)$ and $P(x)$. Consequently this optimization is actually that of maximizing the divergence between two probability distributions. This optimization can then be more elegantly stated by first determining the form of the divergence that is being maximized. Combining terms, the optimization becomes

$$\begin{aligned} &\max \Delta J(x) \\ &= \max \mathrm{E}_{P(x,z)} \log \left(P(x|z) \right) - \mathrm{E}_{P(x)} \log \left(P(x) \right) \\ &= \max \mathrm{E}_{P(x,z)} \log \left(\frac{P(x|z)}{P(x)} \right) \\ &= \max \mathrm{E}_{P(\bar{z})} \int_{x \in S} P(x|z = \bar{z}) \log \left(\frac{P(x|z = \bar{z})}{P(x)} \right) \\ &= \max \mathrm{E}_{P(\bar{z})} D_{KL}(P(x|z = \bar{z}) \| P(x)), \end{aligned} \tag{4.10}$$

where D_{KL} represents this divergence. This divergence is known as Relative Entropy or the Kullback-Leibler divergence [61]. From this perspective maximizing the increase in information can be viewed as maximizing the expected relative entropy given an observation. Note that this particular form is referred to as Mutual Information because it can be further

expanded as

$$\begin{aligned}
&\max \mathrm{E}_{P(\bar{z})} \int_{x \in S} P(x|z=\bar{z}) \log \left(\frac{P(x|z=\bar{z})}{P(x)} \right) \\
&= \max \mathrm{E} \log \left(\frac{P(x,z)}{P(x)P(z)} \right) \\
&= \max D_{KL}(P(x,z) \| P(x)P(z)) \\
&= \max \mathrm{I}(P(x,z); P(x)P(z)).
\end{aligned} \tag{4.11}$$

Divergence is a measure of the difference in information provided by two different distributions. It is then immediate to generalize information-theoretic optimization as the maximization of some general information divergence. Using the f-divergence generalization [28], the optimization can be stated as

$$\max \mathrm{E}_{P(z)} D_f(P(x|z) \| P(x)). \tag{4.12}$$

The f-divergence is a generalization of divergence given appropriate discriminating functions [77]. Given two distributions P and Q with the same measure, the f-divergence is defined by

$$D_f(P\|Q) := \int f\left(\frac{dP}{dQ}\right) dQ. \tag{4.13}$$

Relative entropy is an f-divergence for the case when the discriminating function is $f(u) = u\log(u)$. The power in generalization to f-divergence is that the behavior of the optimization can be designed by varied by using different types of discriminating functions. Besides relative entropy, another useful divergence is Hellinger distance, obtained using the discriminating function $f(u) = \frac{1}{2}\left(\sqrt{u}-1\right)^2$. With this function the divergence becomes

$$D_{hellinger} := \frac{1}{2} \int (\sqrt{P(x)} - \sqrt{Q(x)})^2 d\mu(x). \tag{4.14}$$

Another divergence generalization is the α-divergence [78] defined as

$$\begin{aligned}
D_\alpha(P\|Q) : &= \frac{1}{\alpha - 1} \log \int \left(\frac{dP}{dQ}\right)^\alpha dQ \\
&= \frac{1}{\alpha - 1} \log \int dP^\alpha dQ^{1-\alpha}.
\end{aligned} \tag{4.15}$$

Utilizing the α-divergence, modifying the type and behavior of the divergence is then chosen by tuning the value of α. When $\alpha \to 1$, the α-divergence becomes relative entropy. When $\alpha = 0.5$, the α-divergence becomes the Hellinger affinity.

Relative entropy has become somewhat of a default information divergence and is probably the most used. However, these other divergences have been presented because there are

situations in which other types of divergence are better. For example, the Hellinger affinity provides greater discrimination between two distributions that are very similar [59]. One situation in which this arises is when tracking a target that is moving slow relative to the detection frequency. Consequently the probability distribution changes very little from one time to another. In the path planning phase, the Hellinger affinity then provides the greatest discrimination between the optional agent paths and points of observation.

Now recall that the likelihood function $L(x; z, \eta)$ has some designable structure consisting of the deterministic parameters η. Information-theoretic optimization approaches directly optimize based on this structure modeled in the likelihood function. The optimization problem can then be stated as choosing the value of η from

$$\begin{aligned} \eta^\star &:= \arg\max_\eta \mathrm{E}_{P(z;\eta)} D_f \left(P(x|z;\eta) \| P(x)\right) \\ &= \arg\max_\eta \mathrm{E}_{P(z;\eta)} \int_{x\in S} P(x) f\left(\frac{P(x|z;\eta)}{P(x)}\right) d\mu(x), \end{aligned} \tag{4.16}$$

where $P(x|z;\eta)$ accounts for the deterministic parameters which are chosen by the optimization. Recall that

$$P(x|z;\eta) = \frac{1}{N} L(x; z, \eta) P(x). \tag{4.17}$$

And note that it is required to take the expectation over the possible values of z that could be observed.

Value Function Optimization

Recall that information-theoretic optimization approaches directly maximize the expected information. This means that these methods optimally reduce the expected uncertainty. Uncertainty reduction is not always the goal required for a particular search or tracking scheme. For example, consider the case in which a target's position is localized quite well (very low uncertainty). Instead of observing the point that results in minimum expected uncertainty, it may be desired to keep constant surveillance on the target. It is then best to choose the observation point as the point of maximum likelihood. Next, consider the case in which the choice of observation point is constrained to be within some radius of the previous observation point. This constraint restricts the selection of an observation point to be a point that is possibly sub-optimal globally. However, an information-theoretic optimal observation point can be selected by optimizing over an infinite sequence of observation points. Ideally such an optimization could be performed. However, in many scenarios this optimization cannot be performed. The standard approach is then to reduce the length of the sequence from being infinite to being finite. The length of the sequence is problem specific. But, if the sequence is too short, optimization may lead to selecting a sequence that is unacceptable. For example, consider the case of a fixed wing aircraft. Fixed wing aircraft have a minimum turning radius, so at this minimum turning radius, a fixed wing aircraft will orbit around a

central point. Assume that this aircraft only observes points directly below it on the ground. Also assume that there is a globally optimal point that should eventually be observed. If the sequence over which the aircraft performs optimization is too short, the aircraft may get caught in an orbit around the globally optimal point. Thus, if this occurs, the aircraft will never observe the globally optimal point. Computation of expected information over long sequences is infeasible in many scenarios because of the expectation over possible observations. So, with the purpose of extending the length of the sequence, it then becomes necessary to consider alternative metrics for performing optimization. This leads to optimization over value functions. Value function optimization is similar to information-theoretic optimization in that optimization is performed over the probability distribution. However, the objective of value function optimization is not restricted to maximizing expected information and is more tunable to specific search or tracking cases. Value function optimization is then more general than information-theoretic optimization. The generality of value function optimization will be seen below showing that information-theoretic optimization can be derived from a value function perspective.

Value function optimization approaches consist of determining

- The type of search or tracking desired.
- An objective function based on the type of search or tracking desired.

The type of search or tracking is mainly based on the amount of uncertainty present in the estimate of a target's position. If nothing is known, some form of deterministic guaranteed search [74] may be the best option. However, if there is some amount of information, a search based on the target estimate can be performed. An objective function is what is used to perform optimization. In machine learning, an objective function is what determines a reward for particular agent observation states [93]. The optimization then uses rewards computed from objective functions to determine optimal paths. Recalling the possible target positions x existing in the surveillance area S, the autonomous agent's designable structure η (e.g., agent position and sensor orientation), and possible observations $z \in \zeta$, an objective function generally has the form $U(x, z, \eta)$. The expected value of some η is then

$$\begin{aligned} V(\eta) &:= \mathrm{E}\, U(x, z, \eta) \\ &= \int_{z\in\zeta} \int_{x\in S} P(x, z; \eta) U(x, z, \eta) d\mu(x) d\mu(z) \\ &= \int_{z\in\zeta} P(z; \eta) \int_{x\in S} P(x|z; \eta) U(x, z, \eta) d\mu(x) d\mu(z) \\ &= \mathrm{E}_{P(z;\eta)} \int_{x\in S} P(x|z; \eta) U(x, z, \eta) d\mu(x). \end{aligned} \tag{4.18}$$

Notice the similarity between Eq. 4.18 and Eq. 4.16. First, let

$$u = \frac{P(x|z; \eta)}{P(x)}. \tag{4.19}$$

Then, comparing Eq. 4.18 and Eq. 4.16, if $U(x, z, \eta)$ is chosen such that

$$P(x|z;\eta)U(x,z,\eta) = P(x)f(u), \tag{4.20}$$

then it follows that

$$\begin{aligned} V(\eta) &= \mathrm{E}_{P(z;\eta)} \int_{x\in S} P(x|z;\eta)U(x,z,\eta)d\mu(x) \\ &= \mathrm{E}_{P(z;\eta)}\, D_f(P(x|z;\eta)\|P(x)). \end{aligned} \tag{4.21}$$

For example, the relative entropy version of information-theoretic optimization is for $f(u) = u\log u$. It then stands that value function optimization is more general than information-theoretic optimization.

The main advantage of using the more general value function optimization is that there is more design in the optimization procedure. Above, the general form $U(x, z, \eta)$ was used for the objective function. The function is free to design. So consider the alternative form $U(x, \eta)$. This form can essentially be viewed as either holding z fixed or neglecting it altogether. The optimization then becomes

$$V(\eta) = \int_{x\in S} P(x)U(x,\eta)d\mu(x). \tag{4.22}$$

This form uses the current probability distribution over target position rather than averaging over the possible future probability distribution. The usefulness of this form can be seen by setting

$$U(x,\eta) = L(x; z = \neg D, \eta), \tag{4.23}$$

where $z = \neg D$ means the observation is that of not detecting a target. Recalling Bayes Theorem,

$$\begin{aligned} P(x|z=\neg D;\eta) &= \frac{P(x)L(x;z=\neg D,\eta)}{P(z=\neg D)} \\ &= \frac{P(x)L(x;z=\neg D,\eta)}{\int_{x\in S} P(x)L(x;z=\neg D,\eta)d\mu(x)}. \end{aligned} \tag{4.24}$$

It then follows that the value function is

$$\begin{aligned} V(\eta) &= \int_{x\in S} P(x)L(x;z=\neg D,\eta)d\mu(x) \\ &= P(z=\neg D;\eta). \end{aligned} \tag{4.25}$$

This is the probability of *not* detecting a target. Intuitively, an appropriate optimization would be to maximize the probability of detecting a target (or minimizing the probability

of not detecting a target). The probability of detecting a target is $1 - P(z = \neg D; \eta)$. Accordingly, the metric probability of detection (POD) [18, 95, 105] can be defined as

$$\mathrm{POD}(\eta) := 1 - P(z = \neg D; \eta), \tag{4.26}$$

and then the optimization becomes

$$\begin{aligned} V(\eta^\star) &= \min_\eta \int_{x \in S} P(x) L(x; z = \neg D, \eta) d\mu(x) \\ &= \min_\eta P(z = \neg D; \eta) \\ &= \max_\eta \mathrm{POD}(\eta). \end{aligned} \tag{4.27}$$

As defined above, $\mathrm{POD}(\eta)$ provides an intuitive value function for optimization. It avoids the necessity of averaging over the possible observations. And it directly utilizes the observation likelihood function. To understand the benefit this brings, consider the case in which an autonomous aircraft is equipped with a sensor that always points downward no matter what the orientation of the aircraft is. The designable structure is then $\eta = \eta_x$, where η_x is the position of the aircraft. The optimization problem is then an autonomous agent position path planning problem. The likelihood function fully captures the capability of the sensor given the position of the aircraft. The likelihood function then provides an objective function to determine the value of an agent's position within the surveillance area.

Yet, in general the sensor orientation η_ϕ is designable independent of η_x. Consequently $\eta = \{\eta_x, \eta_\phi\}$. In this general case the observation likelihood function no longer captures the capability of the sensor given η_x. Instead it captures only the sensor capability for a given position *and* sensor orientation. If the likelihood function is still used for optimization, the optimization is then significantly complicated and ceases to be a simple agent position path planning problem because sensor orientation optimization is also required.

However, notice that η consists of two independent components. One component is the agent position η_x. The second component is the sensor orientation η_ϕ. Using this decomposition, the optimization can be separated into two parts. The first part is agent position path planning and the second part is sensor orientation optimization given agent position. The key to accomplishing this decomposition is to find a function $f_C(x, \eta_x)$ that captures the sensor capability given the agent position. $f_C(x, \eta_x)$ is essentially the sensor coverage. Sensor coverage specifies the quality of measurement over points in the surveillance area relative to the agent position. This function encodes the capability of the sensor so that its inclusion in the optimization problem allows removal of sensor orientation. Recall that $\mathrm{POD}(\eta)$ was determined by letting the objective function $U(x, \eta)$ be equal to the observation likelihood function $L(x; z = \neg D, \eta)$, which can actually be viewed as opposite to sensor coverage. So a more general value function can be obtained by letting the objective function $U(x, \eta)$ equal the sensor coverage $f_C(x, \eta_x)$. The value function then becomes

$$V_0(\eta_x) := \int_{x \in S} P(x) f_C(x, \eta_x) d\mu(x). \tag{4.28}$$

The optimal η_x is then that $\eta_x^\star$ which maximizes this value.

Following this formalization of value function optimization, all that is needed is a sensor coverage function that accurately describes the capability of the sensor. For the application of interest, there is an autonomous agent equipped with some sensors. These sensors are used to generate observations of target existence within the surveillance area. For a given sensor there is a field of view, as can be readily understood considering the case in which the sensor is a camera. Consider the autonomous agent to be an aircraft and let the camera be perfectly gimballed a full 360 degrees. Considering the resolution and zoom of the camera, there is a restricted set of points on the ground that can be observed. For example, if the camera is calibrated to observe ground points directly below the aircraft, then points extending away from the aircraft will have degraded resolution and worse measurement quality. Extending farther from the aircraft, eventually no useful measurement can be utilized. This variation of measurement quality is a description of a sensor's coverage. Fig. 4.2 shows an example sensor coverage for a camera calibrated to observe points on the ground directly below the aircraft.

Alternatively the camera could be calibrated to observe points at some horizontal radius away from the aircraft. This scenario is useful when the autonomous agent is a fixed-wing aircraft. Optimal tracking is then achieved by orbiting around a detected target and keeping observation of the target. The best measurements will then be at the specified radius, which in the most naive case is the aircraft's orbit. Deviations radially will degrade the measurement quality. An example of this sensor coverage is shown in Fig. 4.3.

Given the position of the agent η_x, the sensor orientation can then be determined by using the observation likelihood function as

$$V_0(\eta_\phi^\star) = \max_{\eta_\phi} \mathrm{POD}(\eta_x, \eta_\phi). \tag{4.29}$$

Choosing the sensor's orientation in this manner works well in conjunction with the sensor coverage definition of value function optimization because both POD and sensor coverage methods optimize probability of detection. However, it is not necessary for Eq. 4.29 to be POD. One alternative would be the maximum likelihood local to the aircraft position.

Extension to Receding Horizon

Recall that there is some designable structure in the likelihood function $L(x; z, \eta)$. This structure is η, which consists of the configuration of an agent and point of observation (e.g., agent position accompanied by sensor orientation). η belongs to some space H consisting of all possible configurations. Until now it has been assumed that η can be arbitrarily chosen within H. Yet, typically the point of observation cannot be arbitrarily chosen because of kinematic or dynamic constraints on η. For example, if η includes the position of a UAV, then the UAV's position is constrained by dynamic physical laws. At each instant in time, the choice of η is then confined to a suboptimal value within some subset $\bar{H} \subset H$ of the

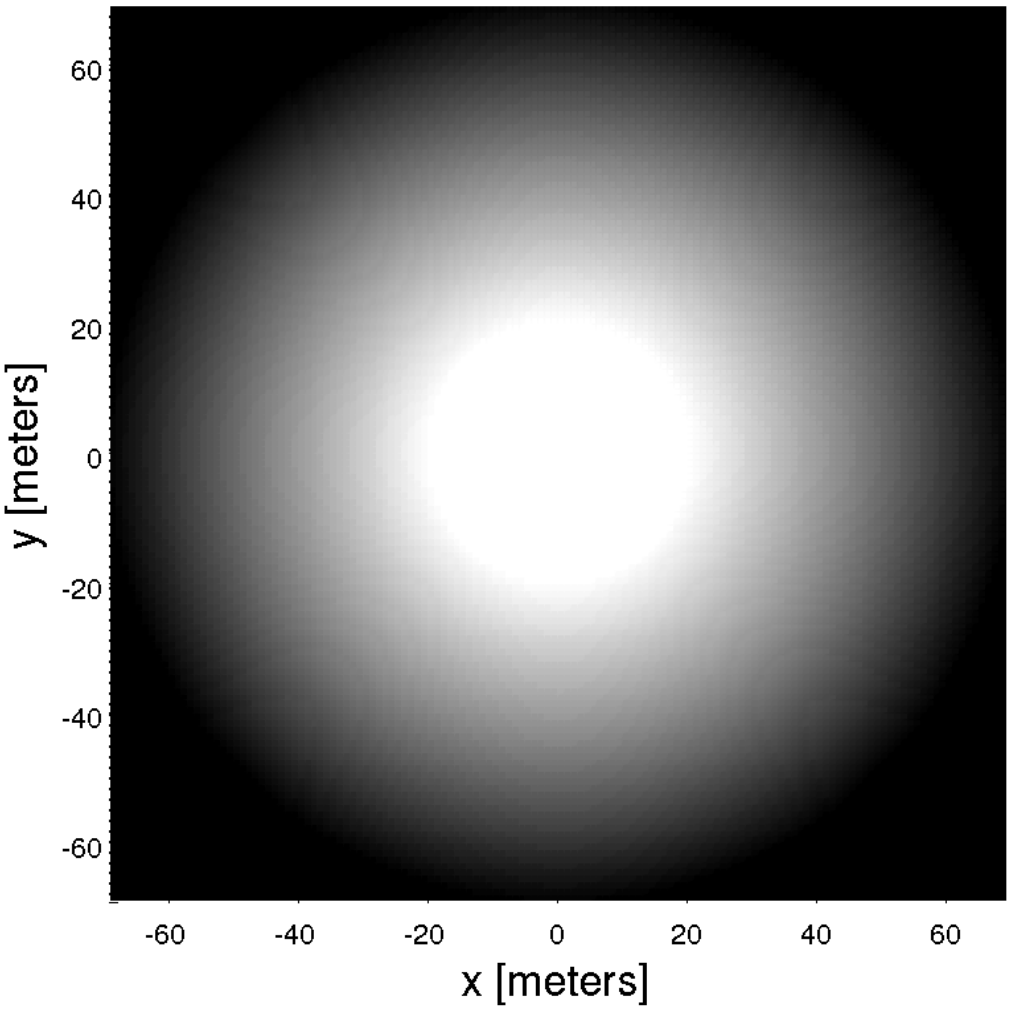


Figure 4.2: Example sensor coverage function for a sensor with optimal viewing straight down from the agent.

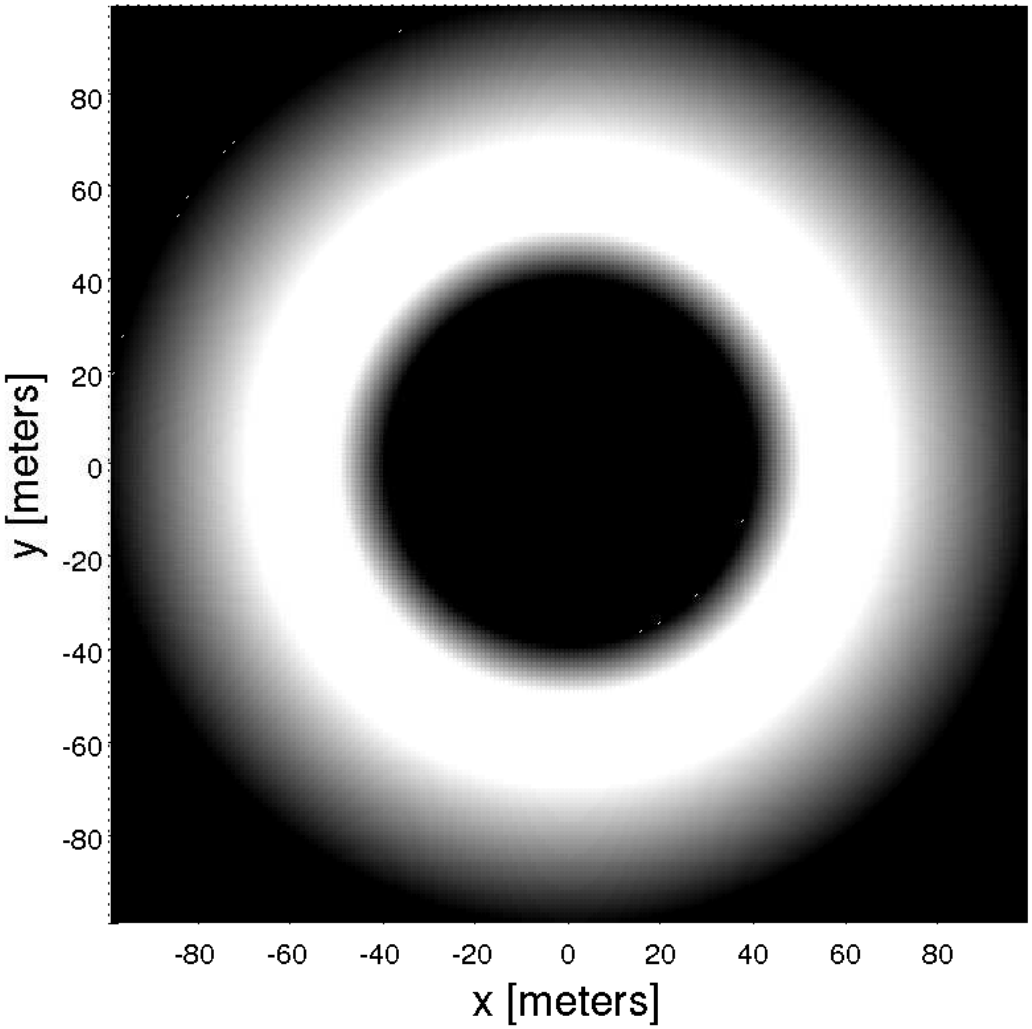


Figure 4.3: Example sensor coverage function for a sensor with optimal viewing at a specified radius away from the agent.

possible configuration local to the agent's current position. Naively using the approaches described above to select some $\eta \in \bar{H}$, the path an agent follows can consequently end up being extremely suboptimal. For example, consider the scenario in which there are two peaks (A and B) in a grid discretized probability distribution, as shown in Fig. 4.4. Peak A is north of the agent by 5 cells. Peak B is south of the agent by 5 cells. Leading up to peak A there is no probability mass. Leading down to peak B there is very little probability mass. Yet at peak A there is .9 probability mass and at peak B there is only .05 probability mass. Let the agent be simple in that at each time step it can translate to one of its adjacent cells and it only observes what is in the cell in which it is positioned. In this case $\eta = \eta_x$. If the purpose is to detect the target, clearly the optimal path is to head toward peak A even though intermediate points in the path will bring no reward. However, if an agent simply follows the path constructed from choosing locally optimal points, it will head toward peak B. From this example it is immediate that selecting a single η is not sufficient when it must be chosen from a local subset $\bar{H}$, due to kinematic or dynamic constraints on η. For this example, one solution to this sub-optimality problem is, instead of optimizing for one local point η, perform optimization to select a sequence of η over five steps. For this particular problem this would result in the agent heading toward peak A and not peak B, as desired.

Following along the lines of sequential experiment design [68, 24, 99] and the information-theoretic optimization strategies posed previously, the five step sequence optimization conceptually presented above can technically be performed by comparing the difference between the starting probability distribution $P(x)$ and the final probability distribution $P(x|z_{1:5}; \eta_{1:5})$. An appropriate value function for some configuration η_0 could then be constructed as

$$V(\eta_0) = \max_{\substack{\eta_{1:5} \\ \eta_i \in R(\eta_{i-1})}} \mathrm{E}_{P(z_{1:5};\eta_{1:5})} D_f \left(P(x|z_{1:5}; \eta_{1:5}) \| P(x) \right). \tag{4.30}$$

Note that the optimization problem posed in Eq. 4.30 encompasses five time steps into the future. At each time step is an unknown, random, possible observation Z_t. Going into the future along this horizon, observe that there can be represented five sequential transitions of the probability distribution according to this sequence of five possible observations $(Z_1, Z_2, Z_3, Z_4, Z_5)$. The possible probability distributions at each of these time steps into the predicted future then define a sequence of probability distributions (or a random process of the target position). Note that the optimization in Eq. 4.30 can be broken into the summation of five sequential designs between each prediction time step. To do this, let the position of the target over the prediction sequence be denoted as $(x_1, x_2, x_3, x_4, x_5)$. The

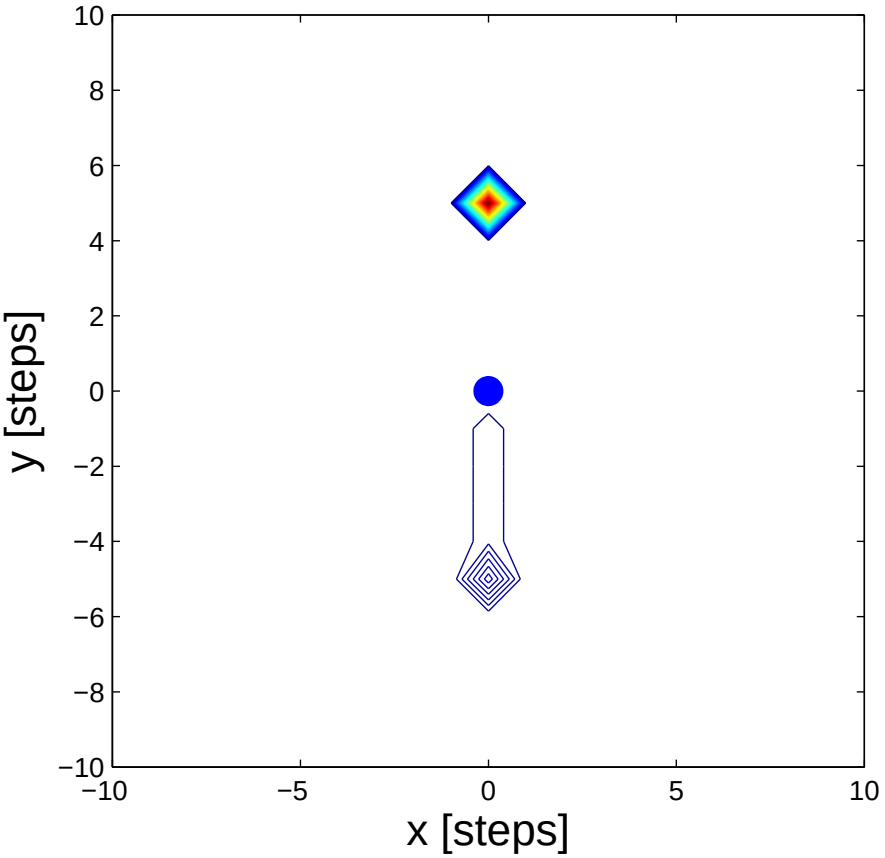


Figure 4.4: Two peak example of suboptimal optimization due to configuration constraints. In this figure, the filled circle represents the agent and the probability distribution is depicted by a contour plot.

optimization in Eq. 4.30 is then equivalent to

$$
\begin{aligned}
V(\eta_0) = & \max_{\substack{\eta_{1:5} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:5} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} D_f \left(P(x_t|z_{1:t};\eta_{1:t}) \| P(x_0) \right) \\
= & \max_{\substack{\eta_{1:5} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:5} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} D_f \left(P(x_t|z_{1:t};\eta_{1:t}) \| P(x_{t-1}|z_{1:t-1};\eta_{1:t-1}) \right) \\
= & \max_{\substack{\eta_{1:5} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:5} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} \int_{x \in S} P(x_{t-1} = x|z_{1:t-1};\eta_{1:t-1}) \\
& \qquad f \left(P(x_t = x|z_{1:t};\eta_{1:t}) \| P(x_{t-1} = x|z_{1:t-1};\eta_{1:t-1}) \right) d\mu(x).
\end{aligned}
\tag{4.31}
$$

In order to see this, look at the summation for the first two possible future observations (Z_1, Z_2). Utilizing relative entropy ($f(u) = u \log u$ where $u = \frac{P(x_t|z_{1:t};\eta_{1:t})}{P(x_{t-1}|z_{1:t-1};\eta_{1:t-1})}$), the summation can be expanded and intermediate terms canceled as

$$
\begin{aligned}
& \sum_{t=1:2} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} D_{KL} \left(P(x_t|z_{1:t};\eta_{1:t}) \| P(x_{t-1}|z_{1:t-1}) \right) \\
= & \sum_{t=1:2} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} H(x_{t-1}|z_{1:t-1};\eta_{1:t-1}) - H(x_t|z_{1:t}) \\
= & \, \mathrm{E}\, H(x_0) - H(x_1|z_1;\eta_1) + H(x_1|z_1;\eta_1) - H(x_2|z_{1:2};\eta_{1:2}) \\
= & \, \mathrm{E}\, H(x_0) - H(x_2|z_{1:2};\eta_{1:2}) \\
= & \, \mathrm{E}\, D_{KL} \left(P(x|z_{1:2};\eta_{1:2}) \| P(x) \right).
\end{aligned}
\tag{4.32}
$$

The cancellation occurring in the summation of sequential experiment designs, as shown in Eq. 4.32, shows that the design over a finite horizon is equivalent to the summation of multiple sequential designs performed between prediction time steps. This proves that sequential design is a valid approach. However, observe that this cancellation means that all intermediate values have no weight in the optimization of Eq. 4.30. Therefore, Eq. 4.30 computes value based only on the *final* probability distribution in the sequence. Based on the conclusions of the unconstrained single point global optimization methods presented above, one might claim that this is perfectly fine and exactly what is sought in the optimization. However, note that Eq. 4.30 is far from being a global optimization problem. As such, there is no guarantee that the fifth (or last) configuration η_5 will be globally optimal. Furthermore, note the difference in planning time between single point unconstrained optimization and finite horizon optimization. With unconstrained single point optimization the planning is done based on the most current information and predicted ahead into the future by *one* time step. With finite horizon constrained optimization, planning is done over multiple time step predictions into the future. As such, it is actually a sequential *expected* experiment design. The usefulness of a prediction based on current information decreases as the prediction's horizon into the future increases. Consequently, optimization performed according to Eq. 4.30 over a finite length horizon may result in lack of robustness due to planning based

on outdated information at the final prediction step. With these issues in mind concerning the robustness of predicted value over a planning horizon, the optimization problem should be modified to not only give value to each step in the planning horizon, but to value time steps closer in the prediction horizon more highly. To do this, the optimization problem can then be written as

$$V(\eta_0) = \max_{\substack{\eta_{1:5} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:5} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} D_f \left(P(x_t|z_{1:t};\eta_{1:t}) \| P(x_0) \right), \tag{4.33}$$

where, instead of summing the sequential designs between each prediction time step, sequential designs between the prior information and each time step are summed. By doing this, each time step in the planning horizon has value.

Optimization over a sequence of points is termed receding horizon optimization. The length of the sequence is called the planning horizon. The horizon must be chosen considering the constraints on η. Receding horizon value function optimization can then be defined by extending the approaches presented above to the sequence $\eta_{1:T} = (\eta_1, \eta_2, ..., \eta_T)$. Here each η_t satisfies the constraint $\eta_t \in R(\eta_{t-1})$ where $R(\bar{\eta}) = \bar{H} \subset H$ defines the configurations of η reachable from $\bar{\eta}$. Recalling that the usefulness of information decreases with prediction horizon into the future, information-theoretic finite receding horizon optimization can then be defined using the f-divergence as

$$\begin{aligned} V(\eta_0) &= \max_{\substack{\eta_{1:T} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:T} \gamma^{t-1} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} D_f \left(P(x_t|z_{1:t};\eta_{1:t}) \| P(x_0) \right) \\ &= \max_{\substack{\eta_{1:T} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:T} \gamma^{t-1} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} \int_{x \in S} P(x_0 = x) f \left(\frac{P(x_t = x|z_{1:t};\eta_{1:t})}{P(x_0 = x)} \right) d\mu(x), \end{aligned} \tag{4.34}$$

where $\gamma \in (0, 1]$ is an optional discount factor. Note that a discount factor is not typically implemented in finite receding horizon optimization. Using the α-divergence, information-theoretic finite receding horizon optimization can be defined as

$$\begin{aligned} V(\eta_0) &= \max_{\substack{\eta_{1:T} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:T} \gamma^{t-1} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} D_\alpha \left(P(x_t|z_{1:t};\eta_{1:t}) \| P(x_0) \right) \\ &= \max_{\substack{\eta_{1:T} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=1:T} \frac{\gamma^{t-1}}{\alpha - 1} \mathrm{E}_{P(z_{1:t};\eta_{1:t})} \log \int_{x \in S} P(x_t = x|z_{1:t};\eta_{1:t})^\alpha P(x_0 = x)^{1-\alpha} d\mu(x), \end{aligned} \tag{4.35}$$

But in general, value function based finite receding horizon optimization can be defined as

$$V_T(\eta_0) := \max_{\substack{\eta_{1:T} \\ \eta_i \in R(\eta_{i-1})}} \sum_{t=0:T} \mathrm{E}_{P(x_{1:t}, z_{1:t};\eta_{1:t})} U(x_{1:t}, z_{1:t}, \eta_{1:t}). \tag{4.36}$$

Observe that receding horizon optimization methods that utilize the random, possible sequence of observations $Z_{1:T}$ will not be able to be computed realistically quickly [81].

This computational complexity was considered above in the single point global optimization methods. However, this fact is drastically worse for receding horizon optimization. So receding horizon optimization is the realm in which objective functions of the form $U(x, \eta)$ show their strength.

Optimal Path Planning

Recall the two peak example given above. Extending the planning to a five step horizon caused the agent to head to peak A, which was the desired action. However, after reaching peak A, optimality is no longer guaranteed. For example, consider the case in which no target is detected at peak A or in the path toward peak A. The probability mass at peak B will then shoot very high. Assuming the agent makes perfect observations defined by the likelihood function

$$L(x; z, x_0, \phi_s) := \begin{cases} 1 & \text{if } z = D \text{ and } x = x_0 \text{ or } z = \neg D \text{ and } x \neq x_0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.37}$$

The probability distribution will then be as shown in Fig. 4.5. The new optimal path would be to head straight down toward peak B. However, the number of steps from peak A to peak B is ten and no probability mass exists within a five step horizon from where where peak A was located. Consequently, in this scenario, the five step receding horizon optimization results in extremely suboptimal paths. Because no rewards exist from the agent's new position, the agent will just wonder off aimlessly. This example demonstrates the fact that finite length receding horizon path optimization still fails to guarantee optimal path planning.

Optimal path planning can be guaranteed by letting the length of the horizon approach infinity. To do this, denote the reward r_t at time t in the horizon

$$r_t := \mathrm{E}_{P(x;\eta_{1:t})}\, U(x, \eta_{1:t}). \tag{4.38}$$

As was done above, the value function was specified to be the sum of these rewards and optimization was performed over a finite horizon of sensor configurations η_t. However, in order to obtain an optimal infinite horizon form, this approach will here be modified to align with the development in reinforcement learning [93]. This is done by first noting that the transition from one configuration η_0 to the next η_1 is done by some control action $a \in A$. This transition can be modeled as a Markov transition probability distribution $P(\eta\prime|\eta, a)$. Implementing this transition probability then allows the optimization to be over the control action instead of the sensor configuration.

For some control action a, the infinite horizon value function can now be defined as

$$V^a_\infty(\eta) := \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \,|\eta_0 = \eta\right]. \tag{4.39}$$

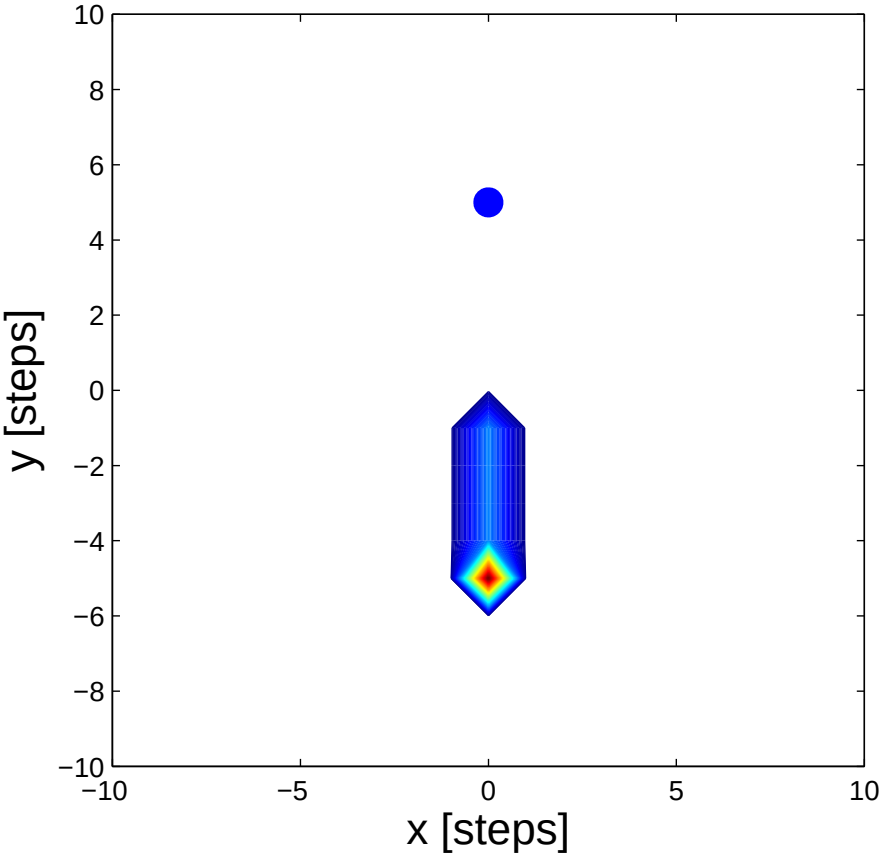


Figure 4.5: Two peak probability distribution example after not detecting a target at peak A. In this figure, the filled circle represents the agent and the probability distribution is depicted by a contour plot.

But this form appears to require optimization over the entire horizon. A more useful form is obtained as

$$\begin{aligned} V_{\infty}^{a}(\eta) &:= \mathrm{E}\left[\sum_{t=0}^{\infty}\gamma^{t} r_t | \eta_0 = \eta\right] \\ &= \mathrm{E}\left[r_0 + \gamma \sum_{t=0}^{\infty}\gamma^{t} r_{t+1} | \eta_0 = \eta\right] \\ &= r(\eta) + \gamma \sum_{\eta'} P(\eta' | \eta, a) V^{a}(\eta'). \end{aligned} \tag{4.40}$$

Then the optimal value is obtained by maximizing over the possible actions as

$$\begin{aligned} V_\infty(\eta) &= \max_{a \in A} V_\infty^a(\eta) \\ &= r(\eta) + \gamma \max_{a \in A} \sum_{\eta'} P(\eta' | \eta, a) V^a(\eta'). \end{aligned} \tag{4.41}$$

Note that in this development summations were utilized. This assumed that the sensor position configuration space was discretized, which is perfectly reasonable. The form of the optimization as stated in Eq. 4.41 can be solved utilizing value iteration or other routines common in reinforcement learning [93, 80]. However, the probability distribution changes at every time step, requiring a value iteration algorithm to run at each instance in time. As such, this optimal value function approach may not be implementable. Yet its consideration should always be made since it is optimal.

Accounting for Shortcomings in the Optimization Approach

Ideally the path planning optimization problem could be posed optimally as an infinite horizon value function optimization problem as stated in Eq. 4.41. However, in the application of interest, the rewards vary at each time step according to the time-evolving probability distribution. Consequently reinforcement learning methods such as value iteration would have to be performed at each time step. So the infinite horizon value function cannot realistically be computed and approximations are required for path planning. Clearly, finite receding horizon optimization is a suboptimal approximation of the optimal infinite horizon optimization. However, as discussed above, the degree of sub-optimality can possibly be too much to be acceptable. Severe sub-optimality encountered while using finite receding horizon optimization is caused by

- Equal-value paths within planning horizon.
- Desired terminal states being in unreachable sets.
- Existence of restrictions on allowable agent state.

If a path planning algorithm naively utilizes finite receding horizon optimization, the algorithm may suffer from severe sub-optimality due to these causes. In many situations this severe sub-optimality may even result in complete failure of the path planning algorithm. Yet finite receding horizon path planning will often provide decent paths that are based on the available information and has the advantage of being mathematically formal. So it follows to question whether finite receding horizon path planning can be heuristically modified to more closely approximate the optimal infinite horizon optimization. If it can, it will do so by addressing these three causes of severe sub-optimality.

One approach to heuristically modify finite receding horizon path planning was developed to address the problem of equal-value paths within the planning horizon [96]. This development was motivated by the scenario in which the probability distribution over the surveillance area consists of small regions with characteristic shape with uniformly distributed probability along with large regions with no probability mass. Fig. 4.6 shows an example of this scenario. Notice in this figure that most of the surveillance area has no probability mass and what regions have probability mass are fairly slender and well defined. In this scenario the path planning problem becomes one in which an agent may be far from any probability mass. Consequently, within an agent's planning horizon there may often be no paths with any value...all paths then have the same zero value. Sticking strictly to a predefined horizon for path planning, it then may frequently be impossible to select a path based on finite receding horizon optimization because all paths are equally bad. The approach developed to address this was to adaptively increase the planning horizon until valuable paths are obtained [96]. The main work was to develop an algorithm enabling cooperation of multiple agents to increase their planning horizons together. Because of the well designed shape of the regions with probability mass, simply extending the horizon of the planner works well, assuming the value function utilized can be quickly computed for long horizons. But in the general case this approach doesn't quite solve the problem of equal-value paths. For example, consider the case in which the probability distribution is one large uniform distribution over a generic shape such as a large square. In this case, restrictions on the maximum horizon will still result in equal-value paths. Furthermore, as the horizon increases the computation increases. Additionally, if the maximum planning horizon is not long enough, it is possible for there to still be unreachable sets in the surveillance area. Similarly, restrictions on agent state may still be a problem.

Another approach is to add a terminal value onto the receding horizon value, as is some times done in optimal control [2]. The value function then becomes

$$V_T(\eta_0) = \max_{\eta_{1:T}} f_{term}(\eta_T) + \sum_{t=1:T} \mathrm{E}\, U(x_t, z_{1:t}, \eta_{1:t}). \tag{4.42}$$

In optimal control the terminal value $f_{term}(\eta_T)$ serves to simply weight the final state in the horizon differently from the rest of the intermediate states. However, in general $f_{term}(\eta_T)$ serves as a heuristic to give a rough check on the value of being in a particular state T steps into the future. Let η_x^{opt} be the globally optimal point within the surveillance area to observe. As an example, imagine that it is undesirable for the agent to travel far from η_x^{opt}. $f_{term}(\eta_T)$ could then be defined as

$$f_{term}(\eta) = -\alpha \|\eta - \eta_x^{opt}\|, \tag{4.43}$$

where α is some scalar to adjust how drastic the penalty is. However, when implementing a terminal value in the value function, care must be taken to balance its effects to not overly dominate the receding horizon value. If this care is not taken, the receding horizon value will do very little in the optimization and once again the value will be based solely on the

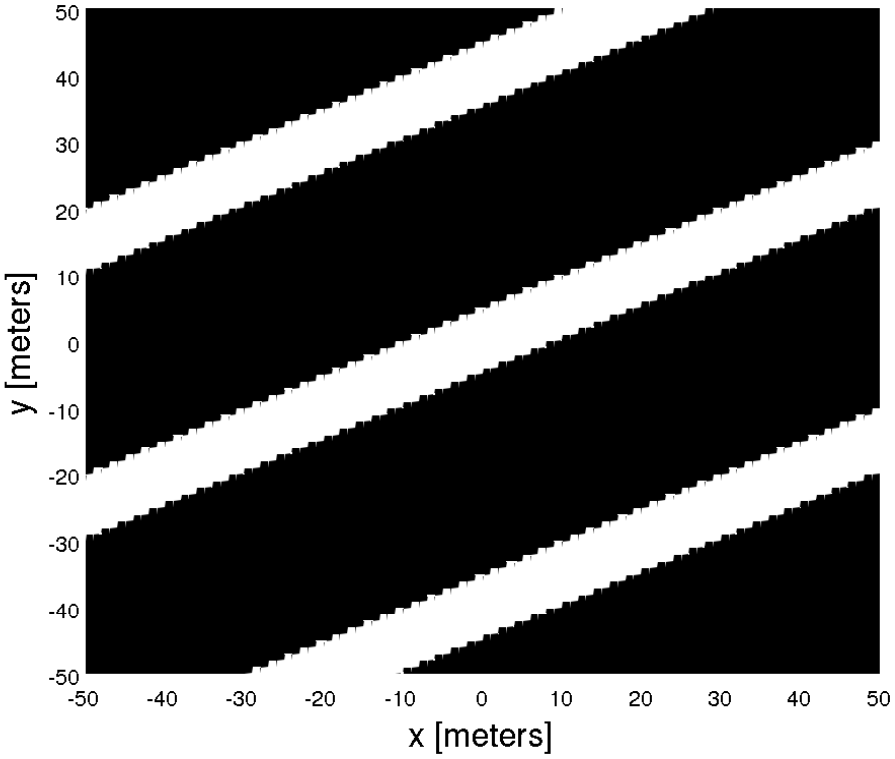


Figure 4.6: Example probability distribution that works well with adaptively extending the horizon of finite horizon receding horizon path planning. This example would be similar to the scenario in which there are three parallel roads next to each other within which surveillance must be conducted.

final state, as was the case in Eq. 4.30. And path planning will end up relying solely on the rough function f_{term}. For example, a distance penalty such as was defined in Eq. 4.43 will tend to dominate the value function and paths will tend to form simply from the distance of the final state in the receding horizon to the globally optimal state. Consequently paths will be formed without much information at all.

An ideal terminal value would provide a gradient of value based on position while being normalized to the scale of the receding horizon value. Determining an ideal terminal value function is a very difficult design process suffering from consequential robustness failures because all possible states must be accounted, but often aren't. Yet, the purpose of implementing a terminal value was to provide a rough check on the deviation from the globally optimal state. Along these lines, instead of the terminal value providing a gradient of value, the terminal value can simply provide a rough check on the general direction toward the globally optimal state. For example, the terminal value could be zero if the agent is headed

roughly toward the globally optimal state and a large negative value if it is not. This type of rough check on general direction is essentially adding a guidance check based on the final state in the planning horizon. Assuming the agent state is perfectly predicted forward to the end of the planning horizon, placing this check solely on this final state is adequate. However, if there is any error in prediction, the final state of the planning horizon will not be realized. For increased robustness, consider placing this guidance at more points along the planning horizon. This is the topic of the rest of this section.

As a third approach, consider guided receding horizon path planning [105], which is a combination of unconstrained globally optimal path optimization and suboptimal finite receding horizon path optimization [105]. This approached is called guided receding horizon optimization because the unconstrained globally optimal point is used to guide the suboptimal receding horizon path. This approach assumes the optimization problem has been decoupled into 1) agent path planning to compute η_x and 2) local observation point optimization to compute η_ϕ. The decoupling in the optimization is accomplished by establishing an appropriate sensor coverage $f_C(x, \eta_x)$ as described above. Let the globally optimal point to observe be labeled η_x^{opt}. A couple reasonable choices for η_x^{opt} are either the point that satisfies $\max P(x)$ or $\max \mathrm{POD}(\eta_x)$. Guided receding horizon path planning then uses η_x^{opt} to guide an agent's finite receding horizon paths to eventually reach η_x^{opt}.

This guidance toward η_x^{opt} is accomplished by defining a cone around the heading ψ of the agent as

$$S_{cone} = \{x \in S : |\angle(\eta_x opt - x) - \psi| < \delta_{cone}\}. \tag{4.44}$$

If at some point while building the tree of possible paths into the receding horizon the cone of a step in the path does not contain η_x^{opt}, then the next step is required to be a point defined by a hard turn toward the η_x^{opt}. This adds a simple constraint to the finite horizon optimization that also reduces the number of possible paths in the optimization.

Assuming the existence of η_x^{opt}, guided receding horizon path planning effectively solves the problem of not reaching η_x^{opt} due to equal-value paths by simply restricting the allowable paths to follow. However, implemented as is, guided receding horizon path planning still doesn't address two of the causes of possible severe sub-optimality in finite receding horizon optimization. These two causes are when η_x^{opt} is in an unreachable set and there are restrictions on the state of the agent. Both of these causes of severe sub-optimality have the potential for making path planning by guided receding horizon optimization to ultimately reach η_x^{opt} infeasible. Consequently, guided receding horizon path planning is accompanied by a couple other modes of control that are activated when events are triggered by these causes [105]. Although these modes and events that activate them are application specific, they can in general be determined by defining

- The locally defined unreachable set S_{UR}.
- Maneuvers to default to when $\eta_x^{opt} \in S_{UR}$.
- Boundaries defining restrictions on agent position.

- Maneuvers to default to when an agent approaches these restricted boundaries.

There are then two events and three modes. The events detect when severe sub-optimality is possible due to η_x^{opt} being in S_{UR} or the agent possibly violating position restrictions. Based on if/which events are triggered, the three modes are then

1. Guided receding horizon optimization.
2. Maneuvers to bring η_x^{opt} out of X_{UR}.
3. Maneuvers to prevent violating restrictions on agent position.

After defining maneuvers for items 2) and 3), optimization by guided receding horizon path planning can then be outlined in the flow chart presented in Fig. 4.7.

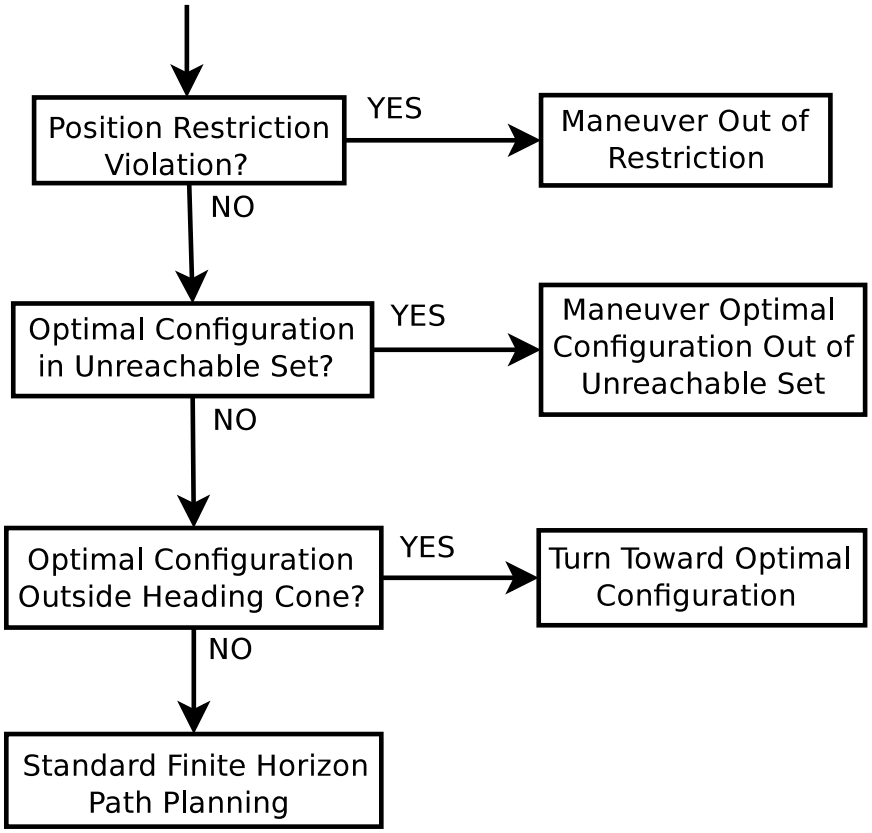


Figure 4.7: Diagram outlining the logical flow of guided receding horizon path planning. This diagram is kept very generic. Implementation of guided receding horizon path planning requires specification of the maneuvers mentioned in this figure as well as methods form determining whether the optimal sensor configuration is unreachable.

In order to demonstrate how to incorporate these items into guided receding horizon path planning, consider an application specific problem. In this problem the agent is an autonomous fixed-wing aircraft equipped with a gimballed camera designed to always point straight down toward the ground independent of the aircraft orientation.

Given an aircraft position and orientation, there is a set of points that the aircraft cannot reach with a constant, maximum bank angle. This is because fixed-wing aircraft tend to have constraints on maximum bank angle. The shaded area of Fig. 4.8 depicts this set of unreachable points. In order to define the set, given the position η_x and heading ψ of the aircraft, and minimum orbit radius r, define six points

$$\begin{bmatrix} c_L \\ c_R \\ l_1 \\ l_2 \\ r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} \eta_x + rRot(\pi/2)v \\ \eta_x + rRot(-\pi/2)v \\ \eta_x + \delta_s Rot(\pi/2)v \\ l_1 + rv \\ \eta_x + \delta_s Rot(-\pi/2)v \\ l_2 + rv \end{bmatrix}, \tag{4.45}$$

where $Rot(.)$ is a rotation matrix and $v = [cos(\psi), sin(\psi)]^{\mathrm{T}}$ is the direction of heading. Neglecting the points reachable by the sensor in the immediate future, the unreachable set is defined as

$$S_{UR}^{neg} = S_{UR}^{L} \cup S_{UR}^{R}, \tag{4.46}$$

where

$$\begin{aligned} S_{UR}^{L} &= \left\{x \in S : \|x - c_L\|^2 \leq r^2\right\}, \\ S_{UR}^{R} &= \left\{x \in S : \|x - c_R\|^2 \leq r^2\right\}. \end{aligned} \tag{4.47}$$

The set of points locally reachable by the sensor is

$$S_{reach}^{s_{local}} = \{x \in S : \|x - \eta_x\| < \delta_s\}. \tag{4.48}$$

Points in the immediate future can also easily be reached by the sensor. Including these, the reachable set becomes

$$S_{reach}^{s} = S_{reach}^{s_{local}} \cup \text{hull}(l_1, l_2, r_1, r_2). \tag{4.49}$$

The unreachable set is then

$$S_{UR} = S_{UR}^{neg} \setminus S_{reach}^{s}. \tag{4.50}$$

If η_x^{opt} ends up in the aircraft's unreachable set, the path planning algorithm must detect it and specify a path that removes η_x^{opt} from the unreachable set. The most aggressive path would detect whether η_x^{opt} is in S_{UR}^{L} or S_{UR}^{R}. If it is in S_{UR}^{L}, then the aircraft would orbit around the boundary of S_{UR}^{R} until η_x^{opt} is no longer in S_{UR}^{L} (and vice-versa). This aggressive path can lead to constant turning around with little receding horizon search. An alternative

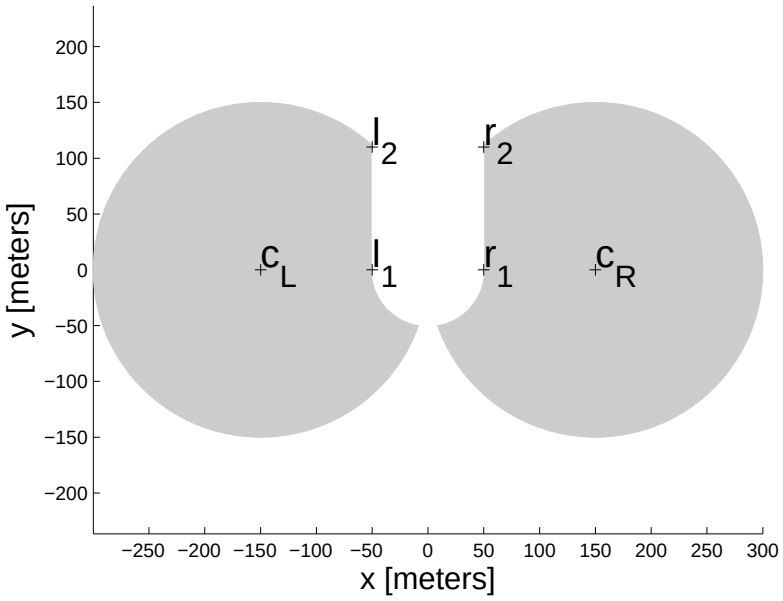


Figure 4.8: Unreachable set (shaded area) defined for a fixed-wing aircraft with an always straight down looking sensor.

is to find an optimal orbit similar to the aggressive path. Since many orbits of varying radius will be compared, it's equivalent to comparing paths of differing horizons. After choosing N_{orbits} to optimize, the set of points assigned to orbit O^i is

$$S_{O^i} = \{x \in S : |\|x - c_{O^i}\| - r| < \delta\}. \tag{4.51}$$

Let $\|S_{O^i}\|$ represent the size of S_{O^i}. Utilizing POD as a value function, define the probability of detection *density* of orbit O^i as

$$\text{pod}(O^i) := \frac{1}{\|S_{O^i}\|}\text{POD}(S_{O^i}). \tag{4.52}$$

$\text{pod}(O^i)$ then quantifies the value of an orbit that is independent to orbit size. The optimal orbit to follow in order to remove η_x^{opt} from the unreachable set is then the solution to

$$O^\star = \arg \max_{O^{1:N_{orbits}}} \text{pod}(O^i). \tag{4.53}$$

Now consider the second factor that influenced the feasibility of guided receding horizon path planning. This factor consists of restrictions on the allowable positions of the agent. Because restrictions on agent position must absolutely be followed, check violations of these restrictions must be the very first check that the path planning algorithm performs. For

the case that this boundary is a rectangle outside which the aircraft cannot go, if at the aircraft's predicted position either S^L_{UR} or S^R_{UR} contains points outside of the boundary, then the aircraft should orbit around the boundary of the opposite side. For example, if S^L_{UR} contains points outside the allowable boundary, then the aircraft should orbit around S^R_{UR} until no boundary conflict exists. Fig. 4.9 depicts when this event is triggered. Notice that this is generally a conservative approach and guarantees the aircraft will remain within the boundary.

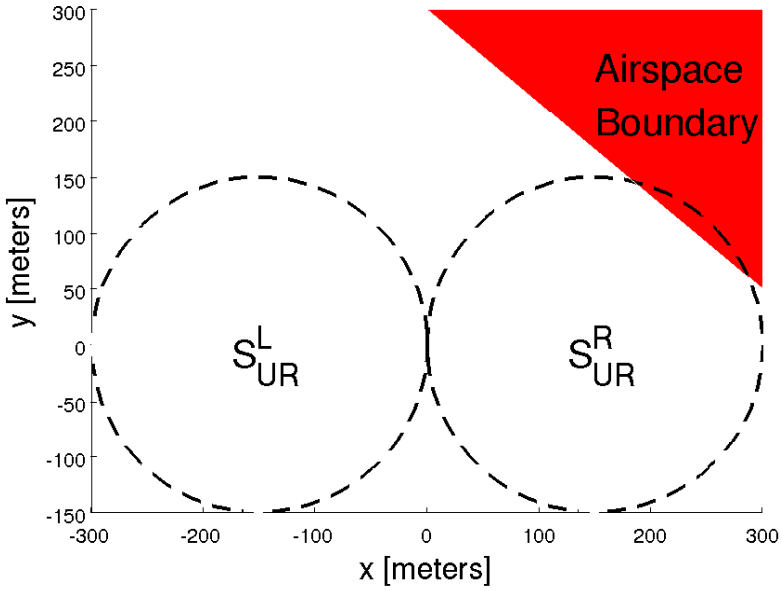


Figure 4.9: Conservative depiction of allowable agent position boundary violation event.

The implementation of these solutions to the two factors influencing feasibility of guided finite receding horizon is for a fixed-wing aircraft with a sensor that points straight down toward the ground independent of the orientation of the aircraft. Using these solutions results in a path planning algorithm that ultimately leads an agent to the optimal state η_x^{opt} while still planning paths over regions with high finite receding horizon value. The flow of this algorithm is presented in Fig. 4.7. Much of these solutions can be easily extended to similar scenarios. For example, agent position restrictions for any scenario in which the agent is a fixed-wing aircraft can be solved exactly as presented above. However, the guidance is dependent on the type of sensor observation, so it is sensor dependent.

4.3.2 Generalization to Target Density Based Path Optimization

Extending the concepts presented for probability distribution based path optimization, the utility of points in the surveillance area must be defined. To do this, consider an agent's

observation coverage $f_C(x, x_0)$ about a point x_0 in the surveillance area. An agent's observation coverage can be determined by accounting for the properties of the agent's sensor. For example, if an agent can make observations perfectly within a radius r, then the observation coverage is

$$f_C(x, x_0) = \begin{cases} 1, & \text{if } \|x - x_0\| < r, \\ 0, & \text{otherwise.} \end{cases} \tag{4.54}$$

However, in general the observation coverage is determined by the sensor's capable field of view as well as the resolution of observable points within the field of view, and the probability of missed detection [58].

To motivate derivation of utility, consider a zero horizon path. The observation coverage can be used to define the utility of a point x_0 as

$$V_0(x_0) := \int_{x \in S} f(x) f_C(x, x_0) d\mu(x), \tag{4.55}$$

where $f(x)$ is the target density distribution. Extending this to finite horizon planning, define the H-step horizon observation coverage over the path $x_{0:H} = (x_0, ..., x_H)$ as

$$f_C(x, x_{0:H}) := 1 - \prod_{t=0:H} \left(1 - f_C(x, x_t)\right). \tag{4.56}$$

The utility of a point $x_0 \in S$ can then be defined as

$$V_H(x_0) := \max_{\substack{x_i \in \mathrm{R}(x_{i-1}) \\ i=1,...,H}} \int_{x \in S} f(x) f_C(x, x_{0:H}) d\mu(x), \tag{4.57}$$

where $\mathrm{R}(x)$ is the set of all points within the reach set of x [105]. Intuitively, Eq. 4.57 represents the expected number of targets within the sensor coverage over a H-step path originating from the point x_0. Maximizing Eq. 4.57 then corresponds to choosing the point $x_0^\star$ that yields the maximum expected number of targets within the observation coverage of a path originating from $x_0^\star$.

To extend utility as defined in Eq. 4.57 to optimization over partition level target density distributions, let the set of partitions defined over the search area be $P = \{P_1, ..., P_n\}$. The partition level target density distribution $f_{P_i}(x)$ for the partition $P_i \in P$ can be defined as

$$f_{P_i}(x) := \begin{cases} f(x) & \text{if } x \in P_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4.58}$$

The partition level utility is then defined as

$$V_H^{P_i}(x_0) := \max_{\substack{x_i \in \mathrm{R}(x_{i-1}) \\ i=1,...,H}} \int_{x \in S} f_{P_i}(x) f_C(x, x_{0:H}) d\mu(x). \tag{4.59}$$

With Eq. 4.59 the paths of the agents are computed. Yet, depending on sensor dynamics, the point in the search area chosen for observation may still have to be chosen. By choosing agent paths that optimize observation coverage, the sensor's point of observation can be chosen by optimizing over the area local to an agent's predicted position at the next observation time t. The optimal observation point depends on the sensor's target detection observation likelihood function $L(x; x_{obs}, x_t)$ [58], where x_{obs} is the point in the search area to observe and x_t is the agent's predicted position at the next observation time. The optimal observation point $x^{\star}_{obs}$ is then the x_{obs} that maximizes

$$V_{obs}(x_{obs}) := \int_{x \in S_r(x_t)} L(x; x_{obs}, x_t) f(x) d\mu(x), \tag{4.60}$$

where $S_r(x_t)$ was defined in Eq. 3.1.

Choosing paths according to this optimization procedure still requires further development, validation, and theoretical guarantees. Yet this procedure is only valid for partitions classified as search partitions. Tracking partitions will likely require a more aggressive procedure which is yet to be developed. Additionally, exploration partitions lack any information which enables the procedure described above. Consequently, appropriate methods for conservative area exploration still need to be determined.

Chapter 5

Results

5.1 Order of Results Presentation

The performance of time-evolving partition classification path planning for target search and tracking depends on the time-evolving partition classification performance as well as the partition level path planning performance. Instead of attempting to analyze both of these at the same time, the partition level path planning is first considered independently of the time-evolving partition classification. Partition level path planning consists of direct distribution path optimization. So, to analyze partition level path planning, methods for direct distribution path optimization are analyzed. Then, after analyzing the performance of direct distribution path optimization, the performance of time-evolving partition classification path planning is analyzed by combining time-evolving partition classification with partition level path planning. In order to analyze the performance of time-evolving partition classification path planning, performance of state-of-the-art path planning will first be presented. Then, the state of the art will be compared to time-evolving partition classification path planning. Afterward, more specific details of time-evolving partitioning will be presented. As such, the performance will be analyzed in the order of

1. Direct distribution path optimization results.
2. state of the art performance results.
3. Comparison between state-of-the-art path planning and time-evolving partition classification path planning.
4. Details of time-evolving partition classification path planning.

5.2 Modeling Considerations

In all of the results presented in this chapter, it is assumed that the team of autonomous agents consists of fixed wing aircraft equipped with visual spectrum gimballed cameras used to observe the environment and detect targets. From experiments, the capabilities of this system were observed [38]. These capabilities were then used to develop the simulation environment so that it closely captures the behaviors of the real system. The camera sensors were designed with a field of view corresponding to a view angle of 0.9273 rad. Effects of resolution were included by limiting the distance of allowable observations to 250 m. The agents were designed to fly at 25 m/s and 100 m altitude with a limited maximum turn rate of 0.2 rad/s.

In the case of moving targets, the targets were allowed to move according to a transition model defined by

$$x_{T,t} = x_{T,t-1} + r \begin{bmatrix} cos(\theta) \\ sin(\theta) \end{bmatrix}, \tag{5.1}$$

where r and θ are distributed as

$$\begin{aligned} r &\sim \text{Gaussian}\left(\mu, \sigma^2\right), \\ \theta &\sim Uniform([0, 2\pi)), \end{aligned} \tag{5.2}$$

with $\mu = 2$ m in one second and $\sigma^2 = 10$ m^2. The time interval of each simulation and inference iteration was 2 seconds. Consequently, each of the targets were biased to move from their position at each simulation iteration by 4 m.

The camera based computer vision target detection was simulated by utilizing the probability distributions developed for the likelihood functions of a camera sensor model, as was presented in the chapter on target density estimation. Recall that in designing the camera sensor likelihood functions, there was developed a set of probability distributions. Of these probability distributions, those that are used to simulate the camera based computer vision target detection are

- $P(V|X, \hat{T}, \hat{R})$: probability of a target at X being within the camera field of view.
- $P(D|V, X, \hat{T}, \hat{R})$: given that a target at X is within the camera field of view, this is the probability of a target at X being detected, based on the resolution of the point in the image that X projects to.
- $P(u|D_{\text{true}}, X, \hat{T}, \hat{R})$: given that there is a true detection, the image location of this detection is sampled from this probability density function, which is Gaussian in the image frame coordinates.
- $P(D|\neg V, X, \hat{T}, \hat{R})$: given that a target at X is *not* within the camera field of view, this is the probability of a false detection.

According to these probability functions, detections are then generated for each true target X_i by first checking if $P(V|X_i, \hat{T}, \hat{R})$ is greater than some threshold P_V. If it is, then it is checked if $P(D|V, X_i, \hat{T}, \hat{R})$ is greater than some threshold P_D. If it is, then it is determined that a true detection for target X_i has been made. The detection is then sampled from $P(u|D_{\text{true}}, X_i, \hat{T}, \hat{R})$. However, if it turns out that none of the true targets pass the check for being within the camera field of view, then a check is made for whether a false alarm should be generated. To do this a sample is drawn from $P(D|\neg V, X, \hat{T}, \hat{R})$, which is a binary probability distribution. If the sample returns true, then a detection is sampled uniformly within the camera image. This computer vision detection simulation is presented in Fig. 5.1.

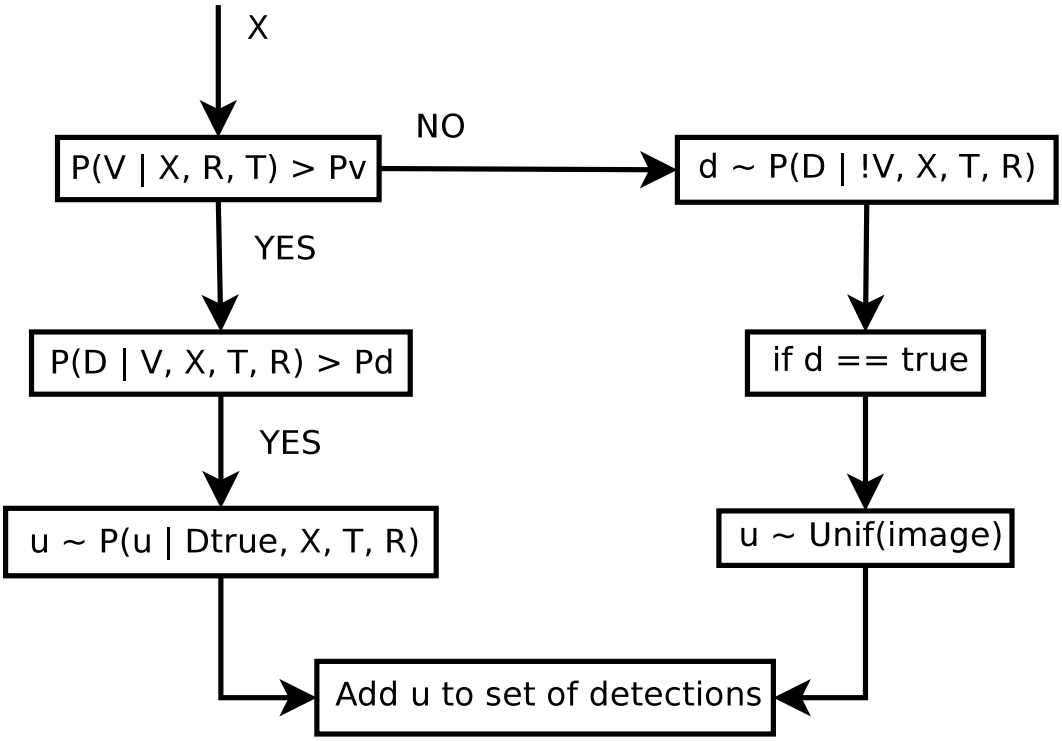


Figure 5.1: Logical flow of the computer vision detection simulation algorithm. Note that all of the probability distributions utilized in this algorithm were previously developed for use in the camera sensor likelihood functions as presented in the chapter on target density estimation. This algorithm accounts for false alarms as well as true target detections, all based upon the model of the camera sensor capabilities and expected computer vision detection algorithm performance.

5.3 Performance Measures

Throughout this chapter, the performance of algorithms was computed utilizing various types of measures. This section presents the types of measures used for each of the algorithms.

5.3.1 Direct Distribution Path Optimization Measures

For the case of a direct distribution path optimization, a target position estimator $\hat{\theta}$ of the true target position θ is readily defined. In this work, the target position estimator was $\hat{\theta} = \eta_x^{opt}$, where η_x^{opt} is the globally optimal point of observation within the surveillance area. And unless stated otherwise, this globally optimal point of observation is

$$\eta_x^{opt} = \arg\max_{x \in S} P(x), \tag{5.3}$$

where $P(x)$ is the most current target estimate position probability distribution.

Utilizing Eq. 5.3 as the target position estimator, for given experiment the estimator absolute error can then be defined as

$$\begin{aligned} \|e\| &:= \|\theta - \hat{\theta}\| \\ &= \|\theta - \eta_x^{opt}\|. \end{aligned} \tag{5.4}$$

An estimator's performance can be measured by computing its mean squared error (MSE). Utilizing the target position estimator error, in Eq. 5.5, the target position estimator MSE is defined as

$$\begin{aligned} \mathrm{MSE} &:= \mathrm{E}\,\|e\|^2 \\ &= \mathrm{E}\,\|\theta - \eta_x^{opt}\|^2. \end{aligned} \tag{5.5}$$

However, because of the lack of parametric structure in this problem, the MSE cannot be computed because the expectation in Eq. 5.6 cannot be performed. Consequently the MSE must be approximated. To approximate the MSE, the sample MSE is computed as

$$\begin{aligned} \bar{\mathrm{MSE}} &:= \frac{1}{N} \sum_{i=1:N} \|e\|^2 \\ \bar{\mathrm{MSE}} &= \frac{1}{N} \sum_{i=1:N} \|\theta - \eta_x^{opt}\|^2. \end{aligned} \tag{5.6}$$

In plots, the square root of this value is typically shown to give a more intuitively presentation.

Sample MSE is a good measure for analyzing probability distribution based path planning algorithms because a plot over time is produced with a curve that should decrease over time.

This decrease of the sample MSE would show improved localization over time. However, in reality sample MSE is not always the only measure required for analysis. In particular, there are other factors that affect path planning performance other than estimator accuracy. One dominant factor is the performance of the computer vision detection algorithm. To check for computer vision detection accuracy, it is best to look only at the data at each detection. This is done best for a static target. Since it is known that the position of a static target is constant, the position of target detections must be approximately constant. This should be reflected in the probability distribution. At each target detection the peak of the probability distribution should get closer to the true target position. Over time the peak should then settle. Yet if there's significant error in the position reported by target detections, this settling of the probability distribution peak will not happen. It will instead flutter around the position of the true target. The amplitude of this flutter for an experiment can be measured by taking the time average of absolute target position estimator error, defined in Eq. 5.5, at each time that there is a target detection. Denote this measure by $\bar{e}_{det}$. In order to define $\bar{e}_{det}$, first define set T_{det} to be the set of all times at which there was a target detection. Now $\bar{e}_{det}$ can be defined as

$$\bar{e}_{det} := \sum_{t \in T_{det}} \|e_t\|, \tag{5.7}$$

where $e_t = \theta - \eta_x^{opt}(t)$ is the estimator error at time t in a particular experiment test run.

In some cases there may be multiple experiment test runs for the same computer vision algorithm. This essentially provides more detection data and can be combined together. To do this, first let $T_{det}(k)$ be the set of detection time indices for the kth experiment test run. Then, given K experiment test runs, $\bar{e}_{det}$ becomes

$$\bar{e}_{det} := \sum_{k=1:K} \sum_{t_k \in T_{det}(k)} \|e_{t_k}\|, \tag{5.8}$$

where t_k is a time instance in the kth experiment test run.

5.3.2 Time-evolving partition classification Path Planning Measures

For the case of there being a probability distribution for each target, path planning performance can readily be determined by the measures presented above. However, in the context of there being an unknown number of targets whose positions in the surveillance area are represented by a distribution of target density, measures for path planning performance are no longer readily determined. In particular, consider the target positions represented by a time-evolving set of partitions. There is no longer any sense of convergence of the distribution to a single peak.

Instead of extending the metrics used for direct distribution path optimization, time-evolving partition classification based target search and tracking performance is analyzed by observing statistics over the simulation time of the

- Number of targets in search and tracking partitions.
- Number of search and tracking partitions.
- Average search and tracking partition size.
- Average size of search and tracking partitions containing targets.
- Number of targets localized.
- Exploration partition size.
- Null target partition size.

The analysis is then more involved. However, these statistics represent the performance of the search and tracking very well. The number of targets in search and tracking partitions specifies how many of the targets are at least in partitions wherein there is sufficient information to detect them fairly soon. The number of search and tracking partitions can then be contrasted with the number of autonomous agents involved in the target search and tracking mission. Next, the average search and tracking partition size gives insight into whether the partitions are growing, staying constant, or shrinking. In an ideal scenario this size would continually shrink until the point of all search and tracking partitions being tracking partitions. The average size of search and tracking partitions containing targets is then an excellent statistic for describing what types of partitions the targets are within. The goal is to have all targets within tracking partitions. The degree to which this is true is specified by the combination of these statistics.

5.4 Direct Distribution Path Optimization

Methods of direct distribution path optimization were tested at three levels of development. These levels are

- Software-in-the-loop simulation.
- Hardware-software-in-the-loop simulation.
- On-board in-flight experiments.

The software development occurred in stages, according to these levels of development. At the software-in-the-loop simulation level, the algorithms were written as well as simulated aircraft and sensing models. These software simulated models were designed to closely match the actual characteristics and expected performance of the aircraft and sensor (the sensor was a gimballed camera). Software-in-the-loop simulation tests were utilized extensively to ensure that the algorithms were ready for hardware based testing. At the Hardware-software-in-the-loop simulation level, the algorithms were actually placed on-board the aircraft and high-fidelity hardware based simulation was used to model the aircraft state. However, at this stage the aircraft hardware was used but fake sensor data was input to the aircraft sensors. Additionally, the only software modeled piece of hardware was the camera. Then, after tests passed successfully, the camera model was removed, the true camera was connected, and the algorithms were tested in-flight on-board the aircraft in the real scenario. Various results will be presented below as collected from data at these three stages.The results will be presented for two types of path planning. These types of path planning are for

- Searching.
- Tracking.

In both of these types of path planning, receding horizon sensor coverage over a distribution is maximized. However, for the types of sensor coverage is different for each case. For searching, the sensor coverage is maximum for points close to the agent. For tracking, the sensor coverage is maximum at the orbit radius of the agent, which was a fixed-wing aircraft. Additionally, searching was performed by guided receding horizon optimization, which was presented earlier.

5.4.1 Searching: Fall 2009 Experiments and Demonstration

In the fall of 2009 development was accomplished to enable autonomous search and rescue by a ScanEagle aircraft [83]. The field experiments were performed at Camp Roberts, California. The mock scenario was to find a kayak that had been laid out on the ground. No human assistance was utilized.

The path planning algorithm was written in C++ and developed through software-in-the-loop (SIL) simulation and in flight on a ScanEagle UAV during a week of experiments. The path planning algorithm was part of a large development project in which system architecture was established for inter-process communication [12]. The processes developed included communication with UAV and gimbal state packets, real-time video processing and target detection (computer vision algorithms), data fusion for estimating a distribution of probabilities for the target position, and path planning to search for and track targets based on the target probability distribution. The results provided in this section are for the performance of this entire system. Each process was improved throughout the flight experiments. The general trend of improvement reflects the improvement of the path planning algorithm.

For example, good target detections are only possible if the UAV and sensor path planning places the sensor in positions and orientations that produce useful observations.

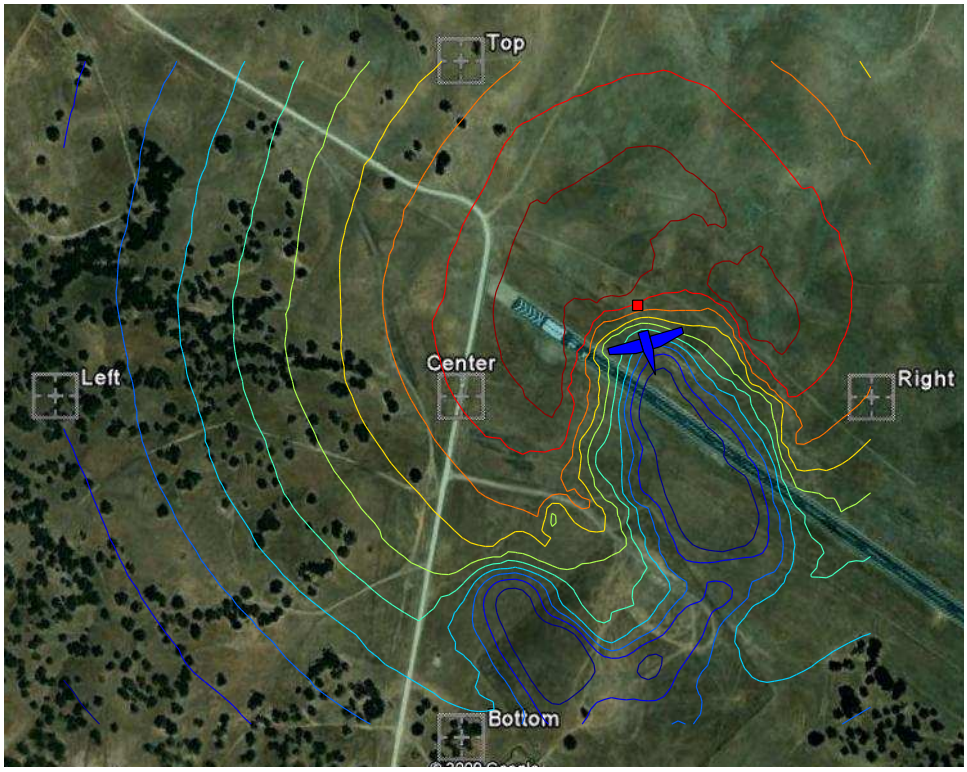


Figure 5.2: Target estimate distribution of probabilities before detections occur. Gradient goes from blue (low probability) to red (high probability).

The data fusion algorithm performed very well and demonstrated robustness in the presence of UAV and gimbal localization and orientation errors. Fig. 5.2 is a representation of an instance of the target estimate probability distribution during one of the flight experiments. In this figure the target has not yet been detected. Only no-detection observation likelihoods have been fused to the target estimate. Fig. 5.3 is a representation of an instance of the target estimate probability distribution during one of the flight experiments after detections have been made. Notice that the distribution has high certainty in its estimate of the target position. In this case, the error in target localization was approximately 40 meters. Throughout the week of experiments, the target detection and localization improved. Recall that η_x^{opt} represents the point of globally optimal observation within the surveillance area. After several flight tests it was determined that the error in target localization was due to flutter in the target positions reported by the detections generated by the computer vision algorithm. Consequently, to measure the improvement of the path planning algorithm, e_{det} defined in Eq. 5.7 was computed for flight tests. Recall that to compute e_{det}, the error between the true target position and the η_x^{opt} estimate is computed for each detection. For each version of the

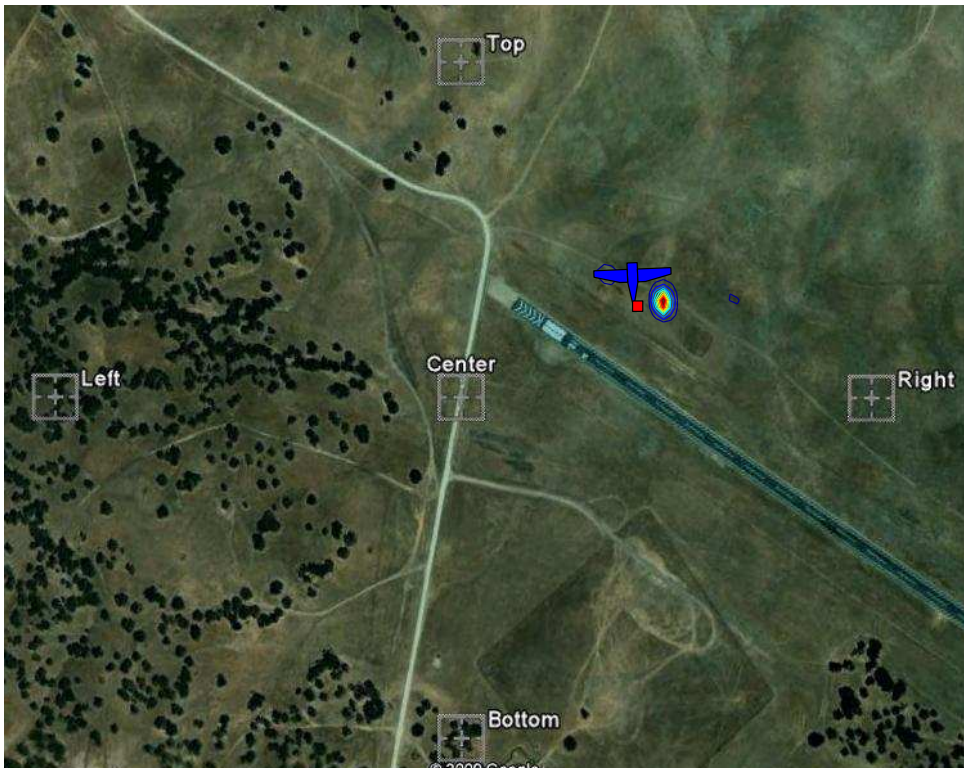


Figure 5.3: Target estimate distribution of probabilities after detections occur. Gradient goes from blue (low probability) to red (high probability).

algorithm several flight tests were performed and the resulting set of errors were used to compute e_{det}. The improvement of the algorithms over algorithm version was then determined by analyzing each version's e_{det}. Fig. 5.4 plots the e_{det} for each algorithm version (consisting of 100 detection events). In Fig. 5.4, the error bars show the minimum and maximum target position estimator absolute errors as computed for each target detection event.

A challenging feature about this project was that all software was on the ground. Our typical flight experiments consisted of on-board computation. So computation on ground posed a new problem. Because all computation was performed on ground, there were problems caused by communication. The heavy communication load resulting in

- Delay in telemetry packets received.
- Delay in gimbal state packets.
- Time synchronization errors in camera image stream.

The accuracy of the target position estimate was ultimately dependent on the delay of camera orientation and aircraft state packets (less than 5Hz) as well as on the synchronization of

video time with ScanEagle autopilot time and calibration of the camera. It was observed that, based only on the frequency of gimbal orientation packets, the error in gimbal orientation could be as high as approximately 15°. This error could not be reduced because of how quickly the gimbal could move within gimbal state packet reception. This corresponds to the localization error in target detections reported from the vision algorithm.

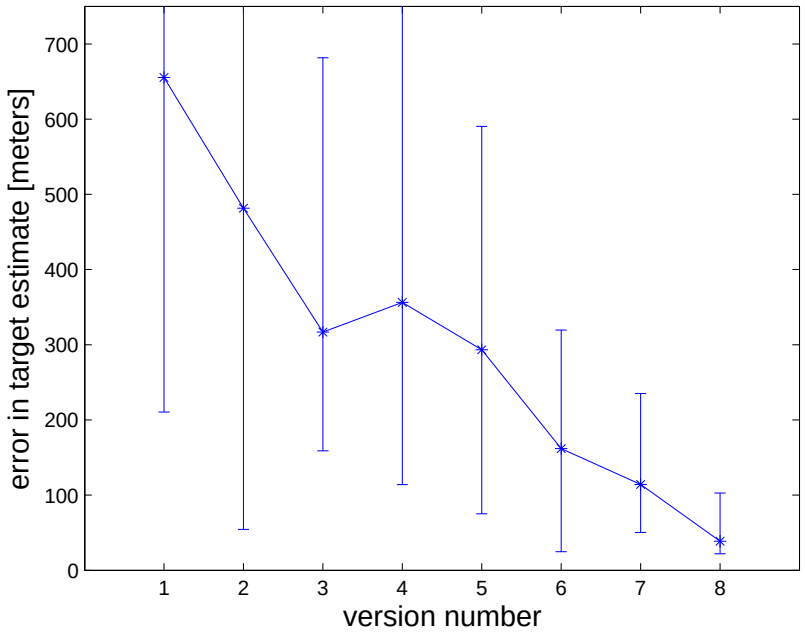


Figure 5.4: Target position probability distribution maximum a-posteriori estimate error improvement over the week of flight experiments as the computer vision, estimation, and path planning algorithms were improved. The experiments consisted of 100 detection events from which the data was derived. Error bars show the minimum and maximum errors that were measured during experiments.

Finally, Fig. 5.5 presents a sample sequence over time of the a ScanEagle aircraft autonomously searching for a static target in an actual flight experiment. The target estimate position probability distribution is represented by a colored contour line plot. The red box is the approximate location of the static target. In Fig. 5.5a can be observed the prior distribution, which was a Gaussian distribution with high variance offset a bit from the actual target location. When this particular experiment trial was initiated, the aircraft was initially flying away from the target. As dictated by the algorithm developed for guided receding horizon path planning and observed in Fig. 5.5b, the aircraft commenced to turn around, sweeping paths leading to the globally optimal point of observation (the location of the maximum value of the probability distribution). Shown in Fig. 5.5, the globally optimal point of observation was ultimately observed and passed without detecting the target since the target

estimate was not quite accurate. Notice that up to this point no target detections have occurred. Consequently only no-detection likelihood functions are updating the probability distribution. At this point the globally optimal point of observation has moved according to observations made by the aircraft camera and the autonomous aircraft returns yet again for this new optimal point in Fig. 5.5d. Then the target is detected, Fig. 5.5e. Note here that there is error in the target localization. As explained above, this error was a consequence of performing all computations on ground, which meant that the rate of state information was low compared to what it would be on-board the aircraft. This then resulted in a random offset in the telemetry packet time and camera video frame grabber time stamps. The camera was controlled by a gimbal. The orientation state of the gimbal was retrieved from the telemetry packets. Because these packets came on the order of half a second, the reported orientation of the gimbal had significant, random offset error. Within the time given for performing the experiments, this error could only be reduced to the offset caused between telemetry packet arrivals. Although this error was an annoyance, the results still show the capability of the autonomous system. Searching performance was good and computer vision based target detections were good. Finally, Fig. 5.5f shows the autonomous aircraft heading back to the new globally optimal point of observation.

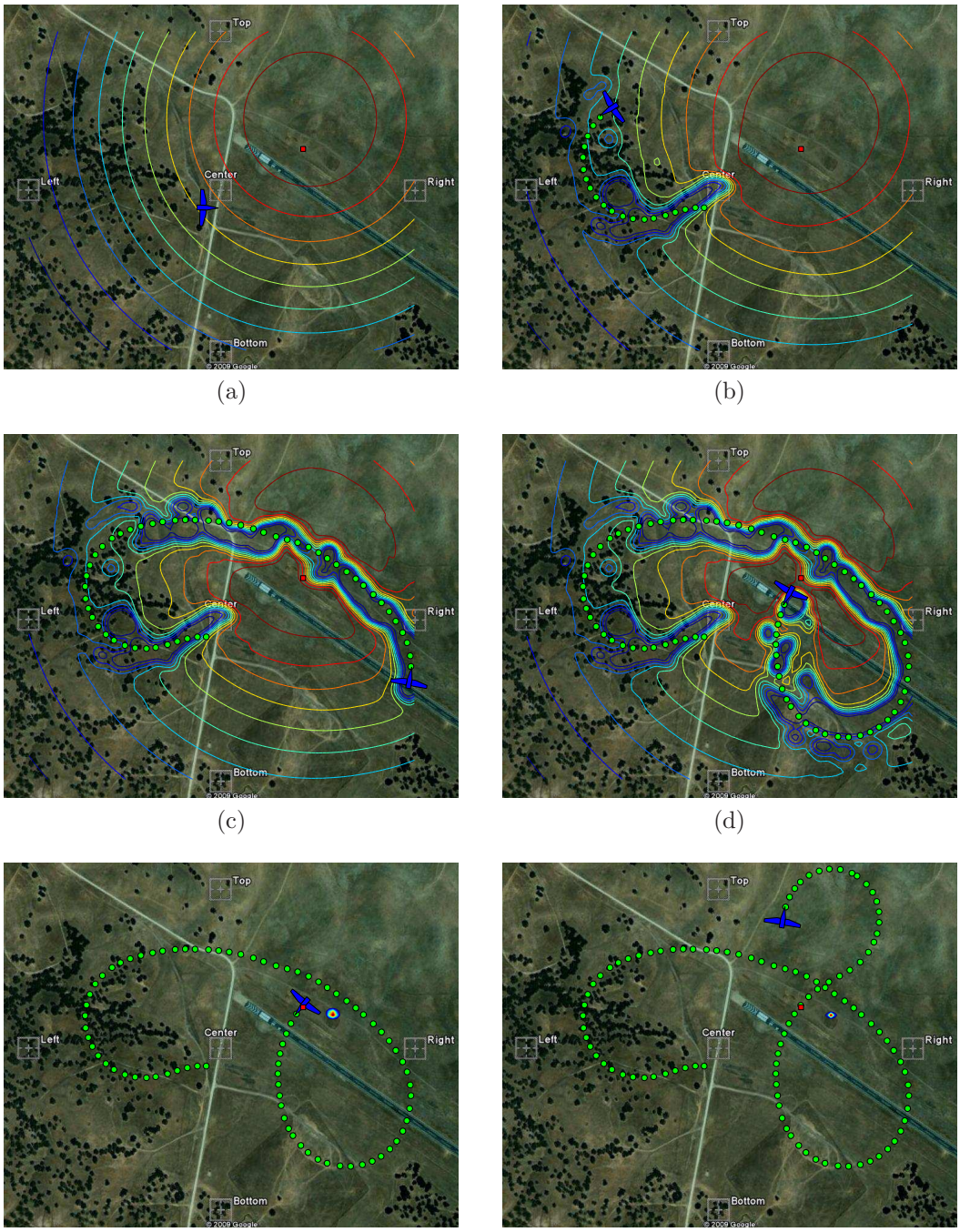


Figure 5.5: Sample sequence of time evolving probability distribution with autonomous aircraft searching for a static target (approximately at location of red box) during actual flight tests performed at Camp Roberts, California in the fall of 2009. Notice the error in target localization. This was due to due camera video and telemetry time offset as explained in this section. The colored lines represent contour lines of the target estimate position probability distribution.

5.4.2 Tracking: Summer 2010 Experiments and Demonstration

The development for a demonstration in the summer of 2010 was significantly more demanding. Yet it built directly off of the development for the ScanEagle project. Whereas the flight experiments during the ScanEagle project were all tested on stationary targets, in the development for summer 2010 moving targets were tested almost exclusively. The development was ultimately to demonstrate moving ground pedestrian tracking by an autonomous fixed-wing aircraft using a visual spectrum camera for sensing. This demonstration was also performed at Camp Roberts, California.

Hardware-in-the-Loop Testing

Similar to the ScanEagle project, all algorithms were extensively tested in complete software-in-the-loop simulation environments. Then they were tested extensively in hardware-in-the-loop environments. Generally what is meant by hardware-in-the-loop is that all hardware is fully utilized but fake sensor data is input to the sensors using a high fidelity simulator. For our system development, the two main algorithmic development works consisted of

- Target estimation and path planning.
- Computer vision based target detection.

Because development within each of these works was easily decoupled, they were tested in separate hardware-in-the-loop environments. However, target estimation and path planning relies on detections generated from the computer vision algorithm. Consequently, in the hardware-in-the-loop environment designed for testing the target estimation and path planning algorithms, a software modeled camera with accompanying fake target detections were utilized. As for the computer vision based target detection algorithm hardware-in-the-loop environment, only simple testing could be performed because fake data from air could not be generated. Consequently, most of the computer vision algorithm tuning had to wait for real flight data.

In this section results from target estimation and path planning hardware-in-the-loop testing will be presented. In these tests the autonomous aircraft was initialized far from the target. However, the prior distribution over the target position was well localized about the target, although the certainty of the distribution was initially high. As such, at the beginning of each test the target estimate was good. In the case of a moving target, it would then transition away from its initial position while the autonomous aircraft drew closer. Consequently, before any detections were possible, the target estimate would deteriorate.

Before testing target tracking performance for moving targets, the performance for static targets was tested. This test really was to check that the performance was still good since much of the software developed in the ScanEagle project had been transitioned to an improved framework and some modifications to the algorithms had been done. Throughout these tests, in order to measure the performance of the autonomous aircraft path planning

algorithm, the target position estimator absolute error was computed after each flight on the experiment data. Fig. 5.6 presents the absolute error of the target estimation for this static target tracking check.

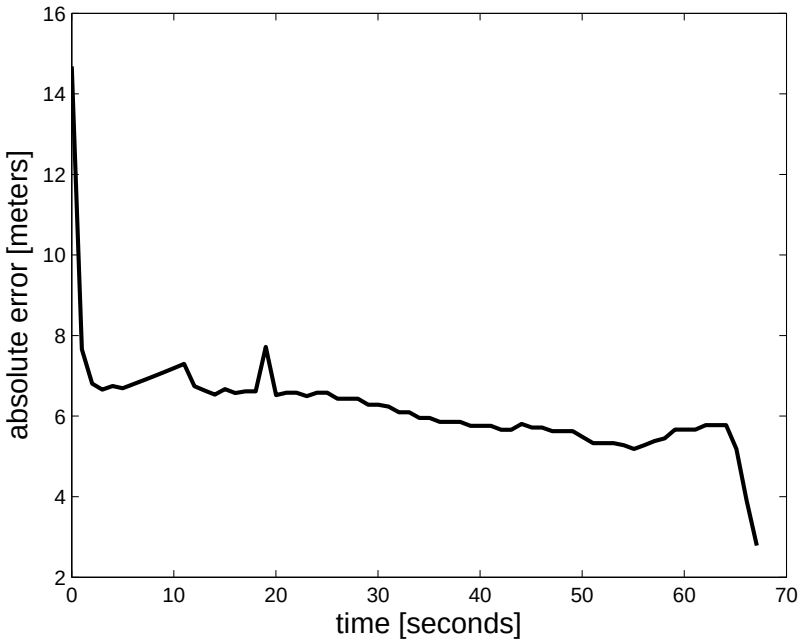


Figure 5.6: Target position estimator absolute error during a hardware-in-the-loop test to check that static target tracking performance was acceptable. From this plot it is observed that the tracking performance for a static target was definitely acceptable.

From Fig. 5.6 it is observed that the algorithm performed quite well for tracking a static target. Due to this quick success, all testing immediately moved forward to testing on moving targets. Before presenting results for moving target tracking, note that the frequency of target observations based on computer vision was at a minimum of once per second. Often the frequency was higher. Depending on the type of aircraft, the aircraft speed was between 25 and 30 meters per second. So between each target observation the aircraft travels a minimum of 25 to 30 meters (82 to 98 feet), sometimes reaching up to 60 meters (over 100 feet) or so. This means that if a target is moving, it can easily be lost before the aircraft settles on a good tracking orbit. Further complicating this is the camera field of view. The height of aircraft was kept above 80 meters (260 feet). In order to detect a ground pedestrian at this height using a camera, the camera had to be zoomed significantly resulting in a small sensor field of view. Consequently, even during good aircraft tracking orbit, the target would not always be in view. Putting these factors together, it is expected that the target position estimator would oscillate according to the frequency of detections. With each detection it is expected that the error would decrease. But until the next detection is made, the error

should increase. This is much different from the case of tracking a static target. For a static target the tracking problem is that of maintaining detections in order to reduce the target position estimator error to zero. However, for a moving target the tracking problem becomes that of preventing the error from getting too bad. The objective then becomes to bound the maximum error of the target position estimator. This is accomplished by the aircraft quickly settling to a good tracking orbit.

Fig. 5.7 shows a sample hardware-in-the-loop moving target tracking absolute target position estimator error over time. True to this plot, the target position estimator absolute error truly did approach zero as time increased. This was mainly due to the aircraft settling into a good tracking orbit. However, as previously mentioned, this error should not decrease to zero. Instead, it should oscillate. It turns out that in this sample run, the target's speed of movement decreased around 400 seconds. So because the tracking was being tracked well, the amplitude of the oscillation decreased around this time. Take, for example, another test run in which the moving target did not slow down its speed of motion. Fig. 5.8 shows the target position estimator absolute error for this sample run. Notice in this figure that the amplitude of the oscillation decreases but stays large according to the speed of motion of the target.

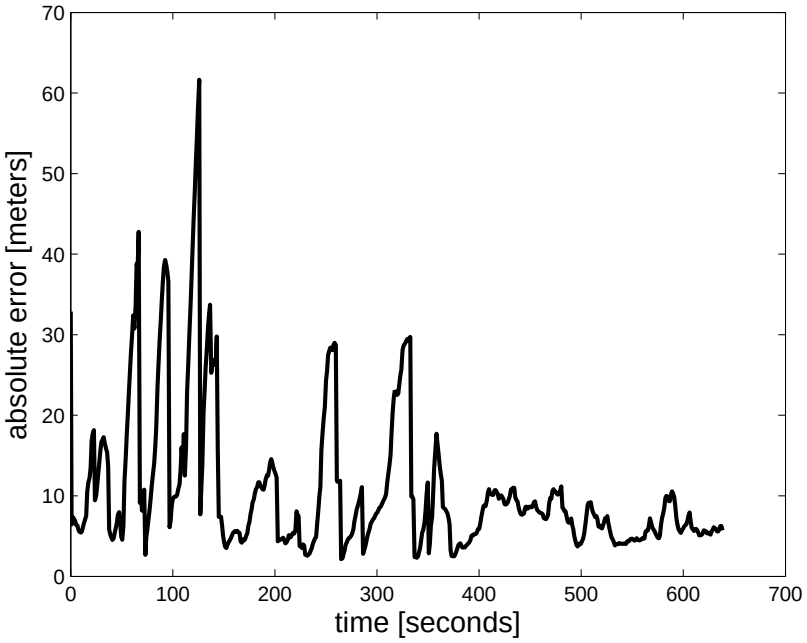


Figure 5.7: Sample hardware-in-the-loop moving target tracking absolute target position estimate error over simulation time. Notice that because the target slowed down over time, the error approached zero as time increased.

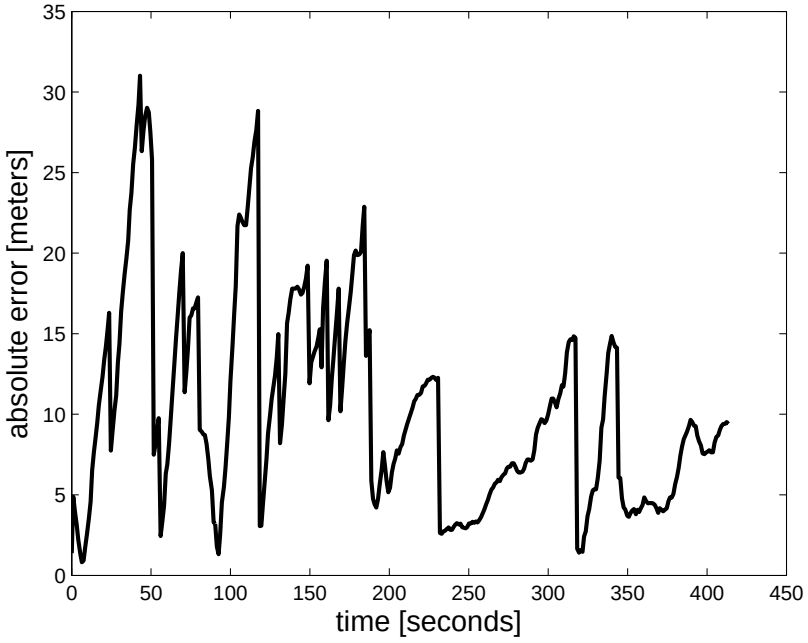


Figure 5.8: Sample hardware-in-the-loop fast moving target tracking absolute target position estimate error over simulation time. Notice that because the target's speed was high, the oscillations are large.

In order to get an idea of the moving target tracking performance of the target position estimator and autonomous aircraft path planning algorithms, several hardware-in-the-loop sample runs were performed and the corresponding sample mean squared error (MSE) was computed. Fig. 5.9 plots the square root of the sample MSE. Notice that the expected oscillations are present in Fig. 5.9. These oscillations could be reduced by increasing the rate of the computer vision based target detection algorithm. However, this algorithm was going as fast as possible.

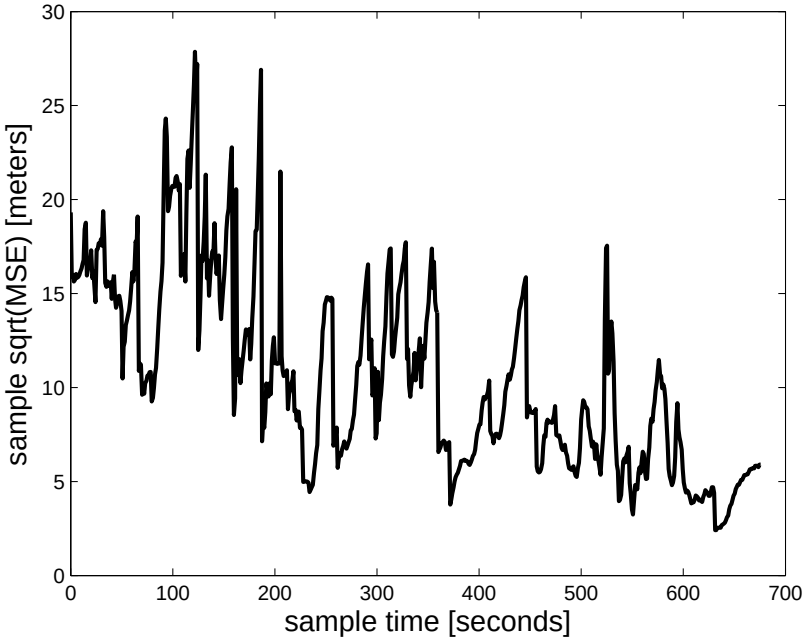


Figure 5.9: Hardware-in-the-loop sample mean squared error of target position tracking estimate error over simulation time. Notice the oscillations caused by target motion between detections.

In-Flight Testing

Similar to the test runs performed in hardware-in-the-loop environments, flight test runs can be grouped into

- Static target tracking performance.
- Moving target tracking performance.

Based on the results from the hardware-in-the-loop test runs above, the path planning algorithm works well given that the computer vision based target detection works sufficiently

well and the in-flight sensor disturbances are not too much worse than those present in the hardware-in-the-loop environment. Consequently, it was expected that at least the static target tracking would perform well. Fig. 5.10 presents the target position estimator absolute error for a sample flight test with a static target. Recall that in all of these experiment test runs the prior probability distribution of the estimate target position was well localized but with high uncertainty. Consequently, with the onset of detections, the measured error increases before it eventually decreases. From Fig. 5.10 this can be seen. But fortunately the error did decrease as expected. This good performance also shows that flutter as was seen in the ScanEagle flight experiments is not present. It was expected that this flutter would not be present because, for these flight tests, all computation was performed on-board the aircraft, whereas for the ScanEagle experiments computation had been on the ground. However, observe Fig. 5.11 that the error drastically increases at the end of the experiment. This was actually due to the sensitivity of the computer vision based target detection algorithm. In this case it had falsely detected a target at a position a little distance from the true target. Combining all flight experiment test runs for static target tracking, the square root of the sample MSE is plotted in Fig. 5.12.

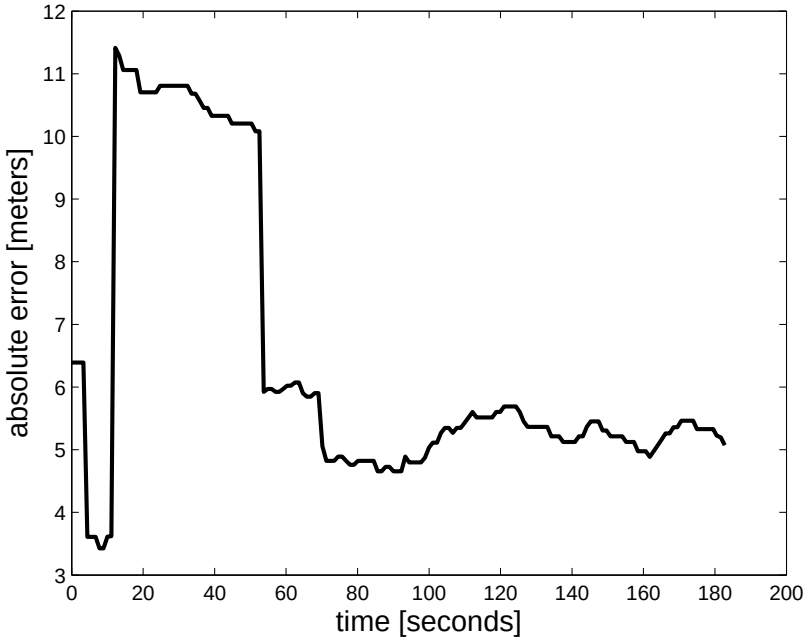


Figure 5.10: Sample target position estimator absolute error for a sample flight test with a static target. Notice that the error drops to approximately 5 meters.

The results presented in Fig. 5.12 suggest that in flight static target tracking performed well. This is what was expected considering that much of the path planning algorithmic development had been done during the ScanEagle project. It was hoped that the computer

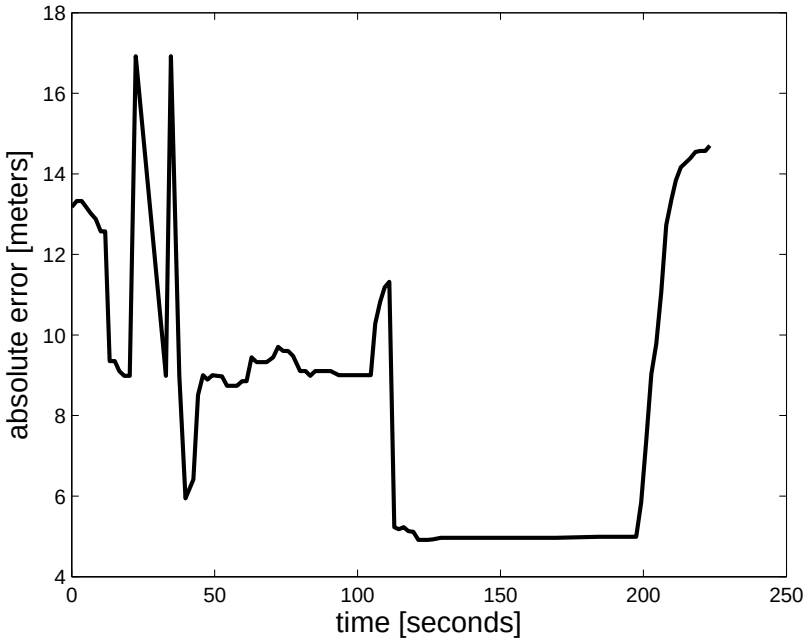


Figure 5.11: Sample target position estimator absolute error for a sample flight test with a static target. In this flight test the sensitivity of the computer vision target detection algorithm was observed. The ending high estimator error was caused by a false alarm. These events provide instances for tuning the sensitivity of the computer vision algorithm before the system is finalized.

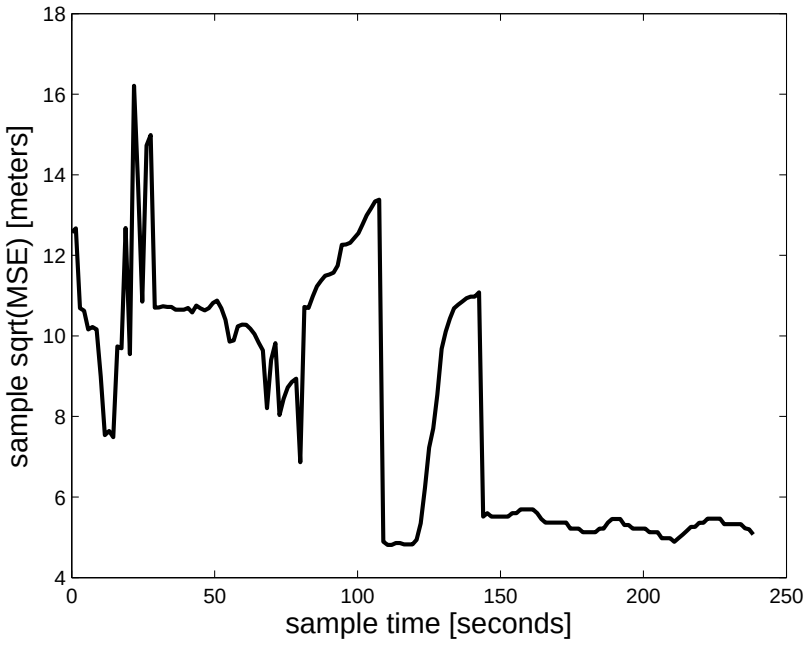


Figure 5.12: Sample square root mean squared error of target position estimator absolute error for the flight tests performed on a static target.

vision based target detection would be more accurate since all computation had been moved to being on-board the aircraft. And fortunately this turned true, even though the computer vision problem was much more difficult during these tests than in the ScanEagle tests. The reason for this extra difficulty was that during these tests it was attempted to detect a ground pedestrian within a field whereas in the ScanEagle it was attempted to detect a much larger sized kayak.that was laid on the airfield pavement runway.

The success of the static target tracking led testing to focus completely on moving target tracking. Although tracking was good for static target tracking, initially moving target performance was not good. It appeared that the computer vision classifier target detection rate was not high enough and that what detections were made were not strong enough to result in sufficiently quick localization in the target position estimation algorithm. It was consequently decided that the sensitivity of the classifier should be increased and that much more image data should be collected. After several flight tests to provide classifier tuning data, a classifier was ultimately reached that provided better target detection rate. However, the target detection rate was still not as high as desired. Fig. 5.13 presents a plot of the target position estimator absolute error for a sample flight test. From this figure the target detection rate is apparent. As expected the absolute error grew in between detections. Observe these error growths between detections in Fig. 5.13. And with each detection the error dropped. So from this sample flight test it can be observed that there were approximately 10 or so target detections within about 215 seconds. This detection rate is not sufficient for high performance target tracking. However, it was enough to at least continue to occasionally observe the target and regain where it was located. After some additional classifier tuning the target detection rate was improved to result in the absolute error as shown in Fig. 5.14. The error in this flight test was kept within a much better bound. Combining all flight tests for moving target tracking testing after decent computer vision based target detection had been achieved, the square root sample MSE was as presented in Fig. 5.15.

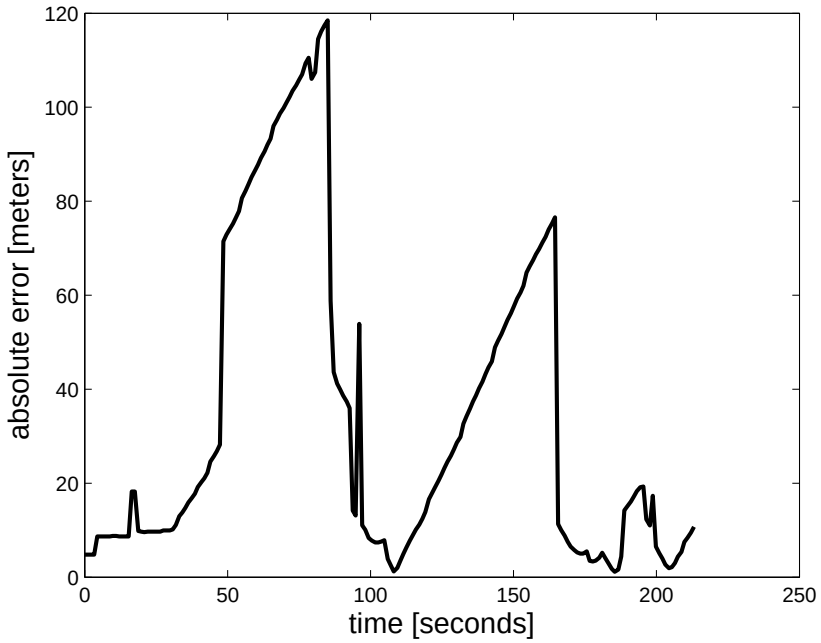


Figure 5.13: Sample target position estimator absolute error for a sample flight test with a moving target. Observe the apparent detection rate in this plot. There were approximately 10 detections. Notice that in between the detections the error grew. It is consequently necessary to maintain a higher detection rate than that apparently achieved in this plot.

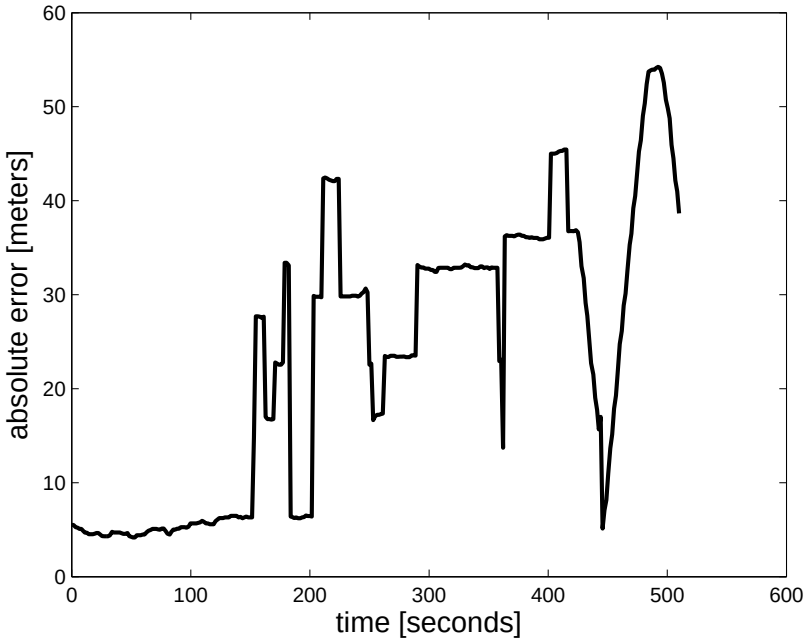


Figure 5.14: Sample target position estimator absolute error for a sample flight test with a moving target. Observe that the apparent detection rate in this plot is much higher than that in Fig. 5.13. Consequently, the error was maintained at a lower level in this plot than in the plot of Fig. 5.13. This was due to significant computer vision detection algorithm tuning after several flight tests.

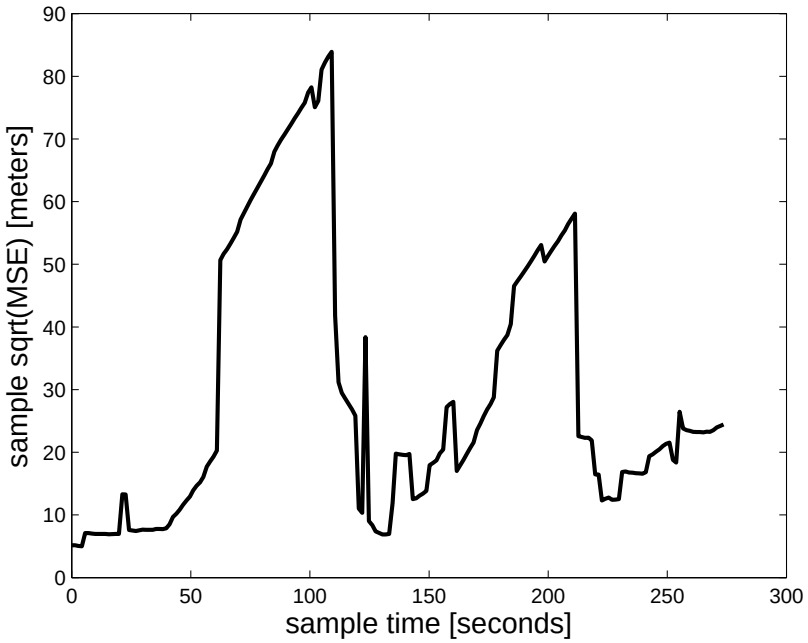


Figure 5.15: Sample square root mean squared error of target position estimator absolute error for the flight tests performed on a moving target.

5.5 State of the Art Performance

The state of the art for general non-parametric distributions is to optimize receding horizon paths directly over distributions. So, the methods developed above then define the state of the art. In this section, the performance of state-of-the-art path planning over a target density distribution will be analyzed. The specific state-of-the-art method chosen for this performance analysis is the method of receding horizon sensor coverage maximization directly over the target density distribution. This state-of-the-art method, and its performance presented here, will be referenced throughout the rest of this chapter.

In order to analyze the performance of state-of-the-art path planning, three scenarios are tested. These scenarios are

- The number of targets is equal to the number of agents.
- The number of targets is less than the number of agents.
- The number of targets is greater than the number of agents.

To test the scenario of the number of targets being equal to the number of agents, the specific case of there being six agents and six targets is considered. To test the scenario of the number of targets being less than the number agents, the specific case of there being six agents and three targets is considered. To test the scenario of the number of targets being greater than the number of agents, the specific case of there being three agents and six targets is considered.

To enable comparison of the performance of state-of-the-art path planning with the performance of time-evolving partition classification path planning, the state of the art performance is analyzed here by utilizing the same metrics that will be used to measure the performance of time-evolving partition classification. To perform this analysis, partitions must be computed. However, the partitions are not used for path planning. The partitions are only used to analyze the performance of path planning. For example, path planning is considered to perform well if the number of partitions remains bounded, all targets eventually are found within search or tracking partitions, the exploration partition decreases, and the number of localized targets approaches the true number of targets. With this in mind, figures for each of these scenarios will now be presented below with these quantities plotted.

5.5.1 State of the Art: Number of Targets Equals the Number of Agents

Fig. 5.16 presents the number of targets that were found within search or tracking partitions over simulation time. Notice that all targets are quickly found with search or tracking partitions. The number of search and tracking partitions over simulation time is presented in Fig. 5.17. Notice that the number of search and tracking partitions remains constant,

so the number of partitions is bounded. Fig. 5.18 presents the average partition size over the simulation time. Note that this also remains roughly constant, instead of decreasing as was desired. Similarly the average partition size in which a target was found also remained roughly constant, as shown in Fig. 5.19. This suggests that the search strategy resulted in tracking partitions rarely forming. Now consider the exploration partition. The exploration partition specifies which regions have low information. As such, it is desired for the exploration partition size to decrease over simulation time. However, according to Fig. 5.20, there is a point in time at which the exploration partition size starts to increase. Finally, the number of targets localized is then presented in Fig. 5.21. Notice that this number never gets close to the true number of targets in the time given during the simulation.

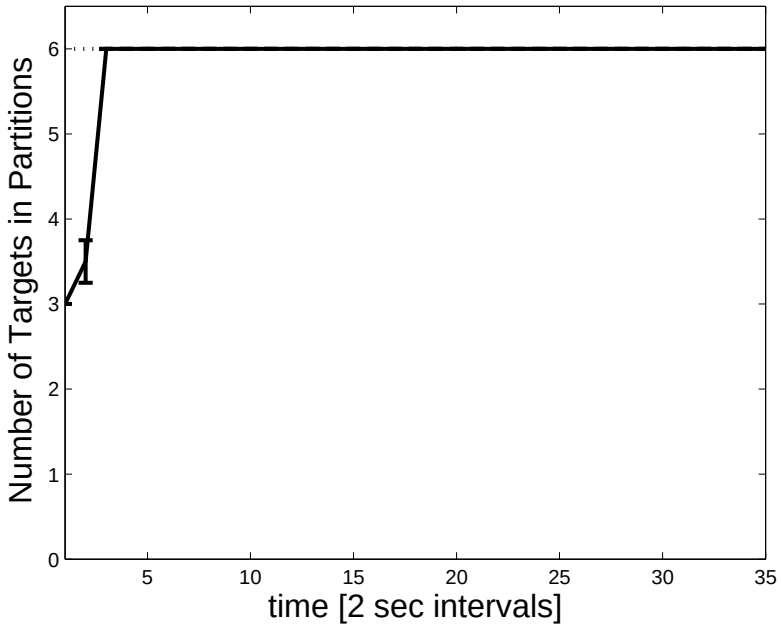


Figure 5.16: Scenario: six targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of targets within search and tracking partitions over simulation time. Error bars represent sample standard deviation.

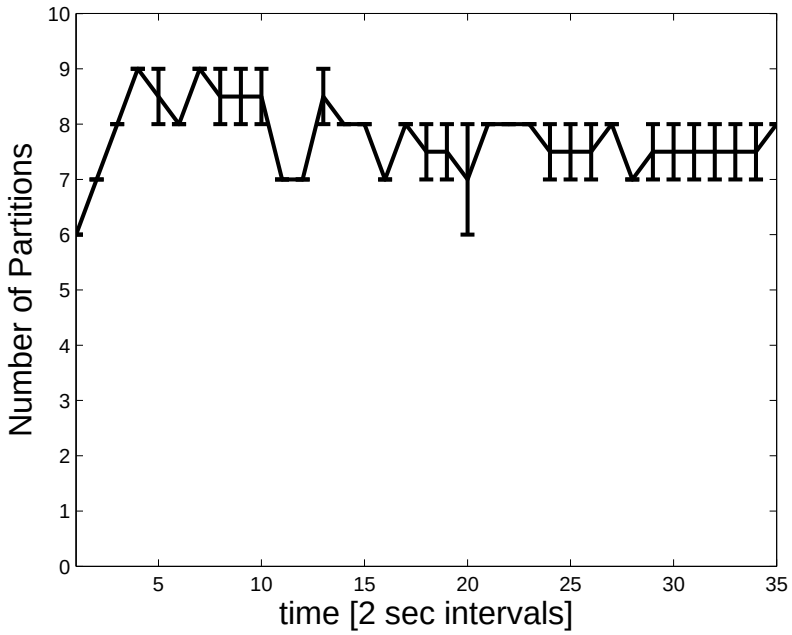


Figure 5.17: Scenario: six targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

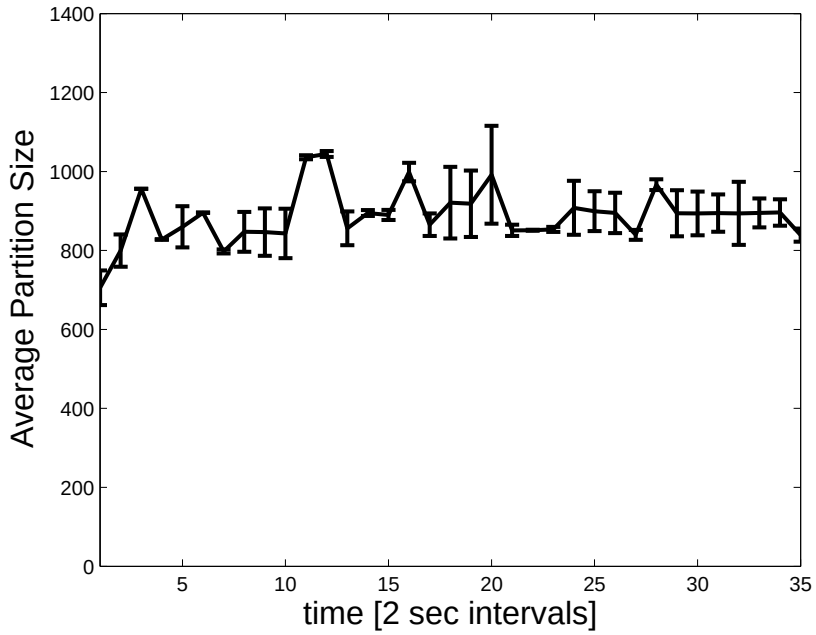


Figure 5.18: Scenario: six targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

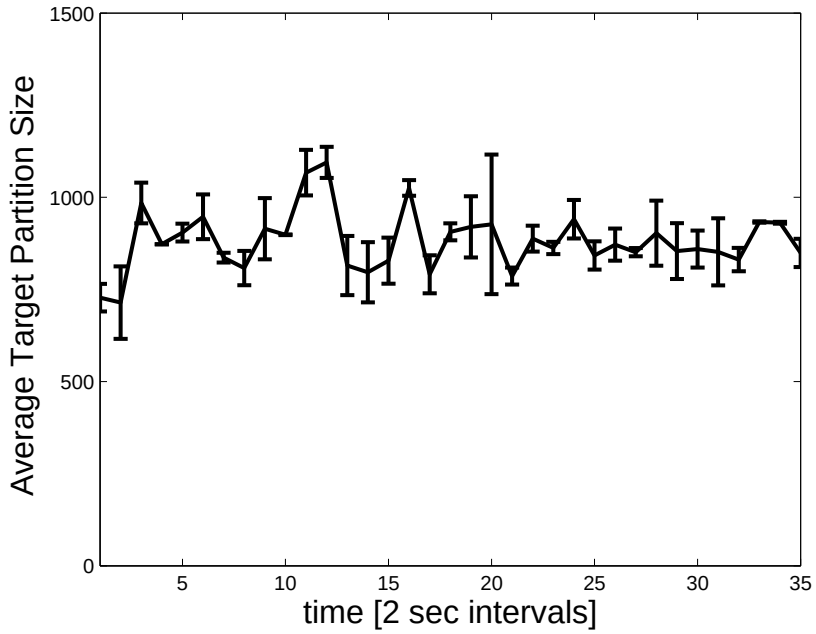


Figure 5.19: Scenario: six targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean average partition size of partitions containing targets over simulation time. Error bars represent sample standard deviation.

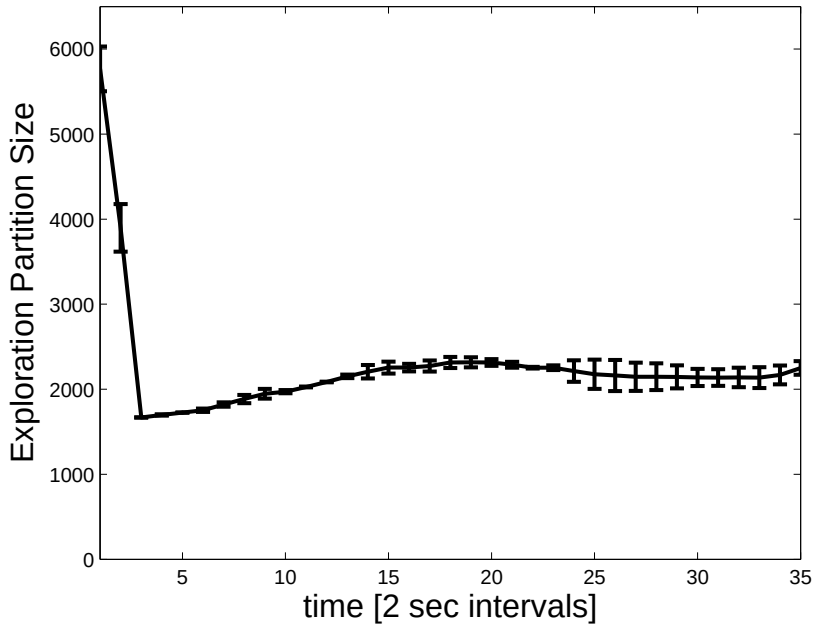


Figure 5.20: Scenario: six targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean exploration partition size over simulation time. Error bars represent sample standard deviation.

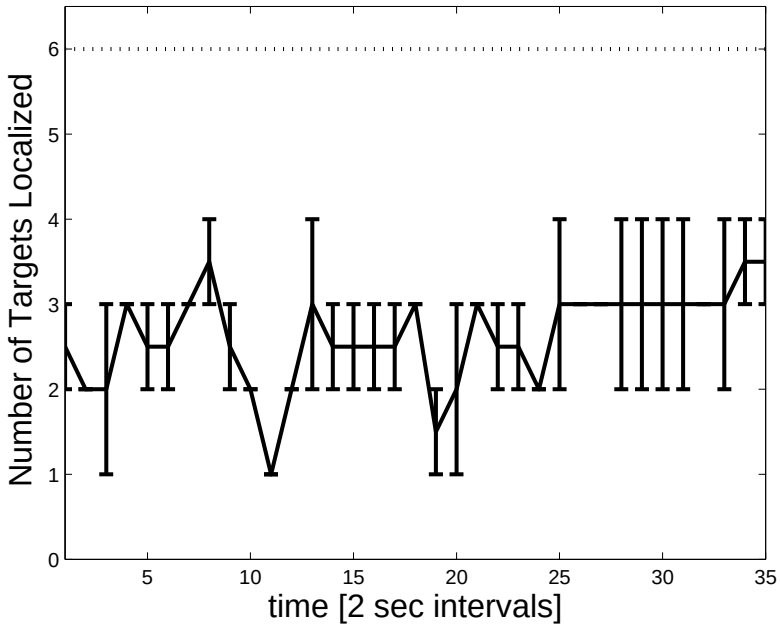


Figure 5.21: Scenario: six targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of targets localized over simulation time. Error bars represent sample standard deviation.

5.5.2 State of the Art: Number of Targets Less than the Number of Agents

Similar to the case of the number of targets and agents being equal, Fig. 5.22 shows that the number of targets that were found within search or tracking partitions over simulation time quickly approached the true number of targets. Additionally, according to Fig. 5.23, the number of partitions is bounded. However, just as was the case above, Fig. 5.24 shows that the average partition size over the simulation time remained roughly constant, instead of decreasing. Also, the average partition size in which a target was found also remained roughly constant, as shown in Fig. 5.25. Presented in Fig. 5.26, the exploration partition size tends to follow a similar trend as for the case of the number of targets and agents being equal. As such, the exploration partition does decrease. However, there is a point in time at which it starts to increase. The final performance measure to consider is the number of localized targets, which is presented in Fig. 5.27. From this figure it is clear that the state of the art did not perform well for this scenario to localize targets. This can be understood by noting that the number of targets for this scenario was only three. So there was more space that had to be searched to find the targets.

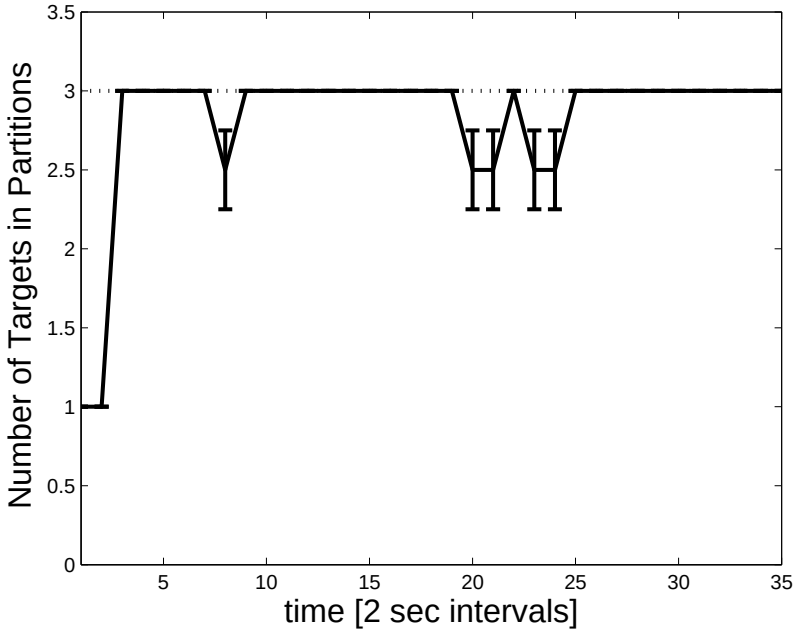


Figure 5.22: Scenario: three targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of targets within search and tracking partitions over simulation time. Error bars represent sample standard deviation.

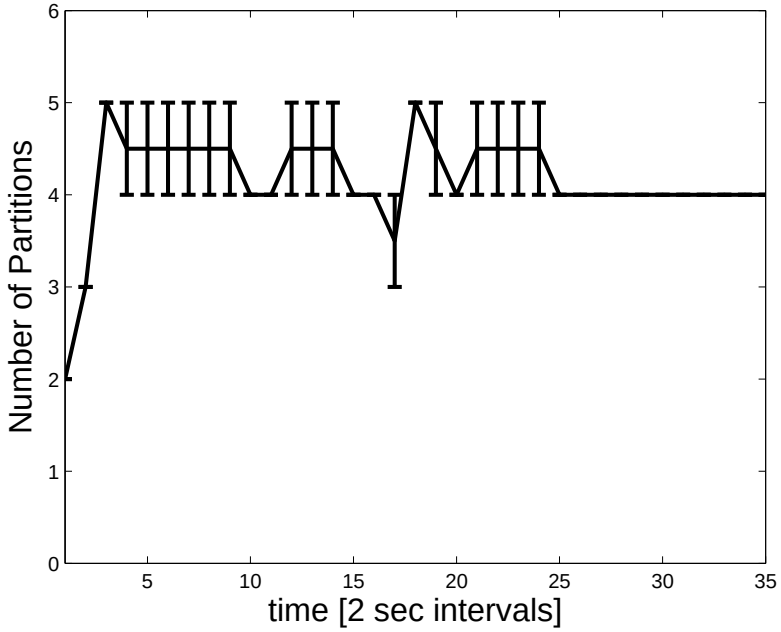


Figure 5.23: Scenario: three targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

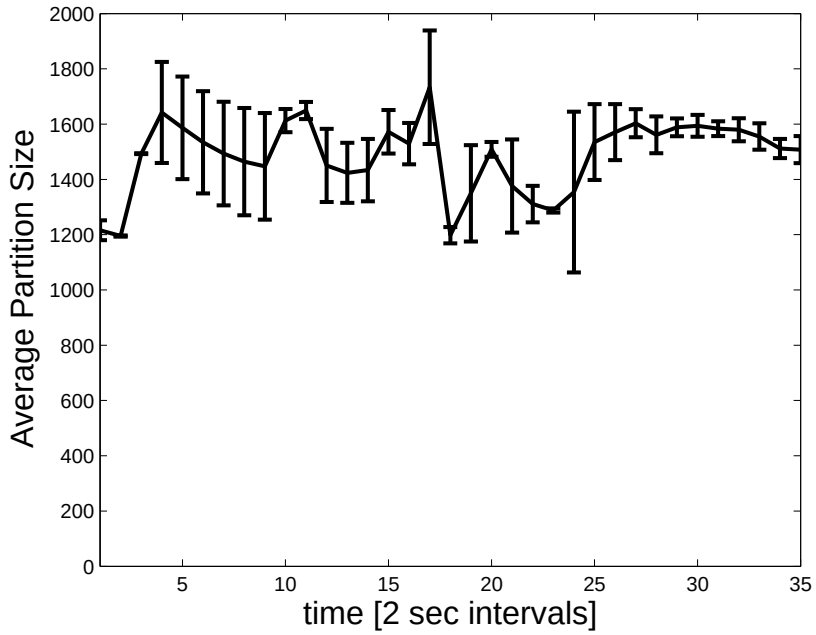


Figure 5.24: Scenario: three targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

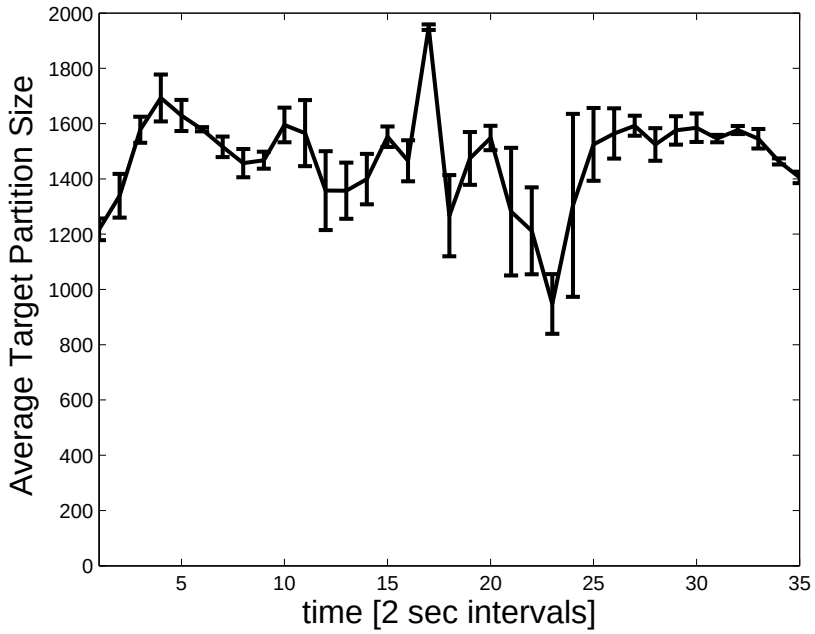


Figure 5.25: Scenario: three targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean average partition size of partitions containing targets over simulation time. Error bars represent sample standard deviation.

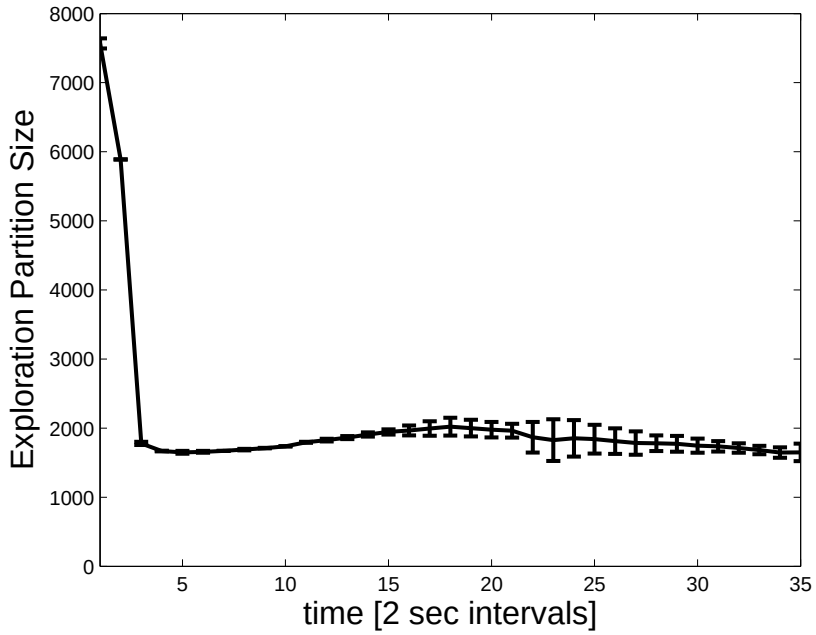


Figure 5.26: Scenario: three targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean exploration partition size over simulation time. Error bars represent sample standard deviation.

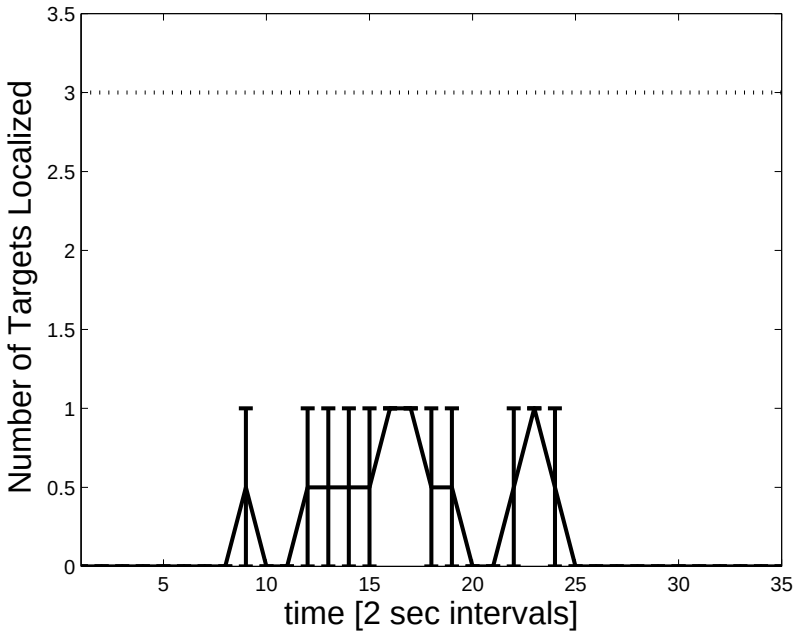


Figure 5.27: Scenario: three targets and six agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of targets localized over simulation time. Error bars represent sample standard deviation.

5.5.3 State of the Art: Number of Targets Greater than the Number of Agents

This scenario is generally a difficult one to address in an effective way because the number of available resource (agents) is less than the number of targets that should be tracked eventually. Consequently, there is a trade-off between which targets to track. However, despite this, according to Fig. 5.28 and Fig. 5.29, partition classification works well because all targets are quickly captured within search or tracking partitions and the number of partitions is bounded. Yet, as far as the state-of-the-art path planning approach performance goes, the average partition size remains roughly constant, as presented in Fig. 5.30. Also, the average partition size in which a target was found also remained roughly constant, as shown in Fig. 5.31. This suggests that most targets remained in searching partitions, instead of tracking partitions. Fortunately the exploration partition size drops significantly and remains low, as presented in Fig. 5.32. And according to Fig. 5.33, the path planning approach actually performs fairly well to keep at least a couple targets localized throughout the course of the simulations.

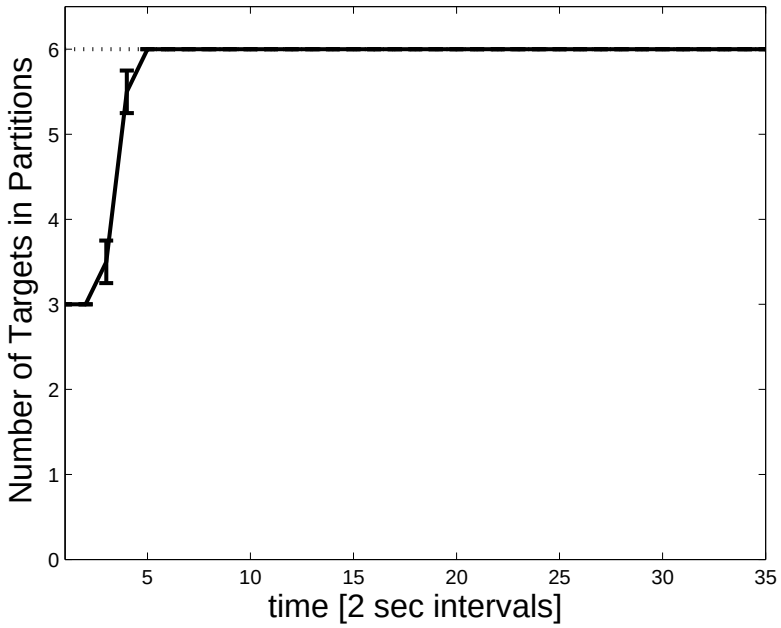


Figure 5.28: Scenario: six targets and three agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of targets within search and tracking partitions over simulation time. Error bars represent sample standard deviation.

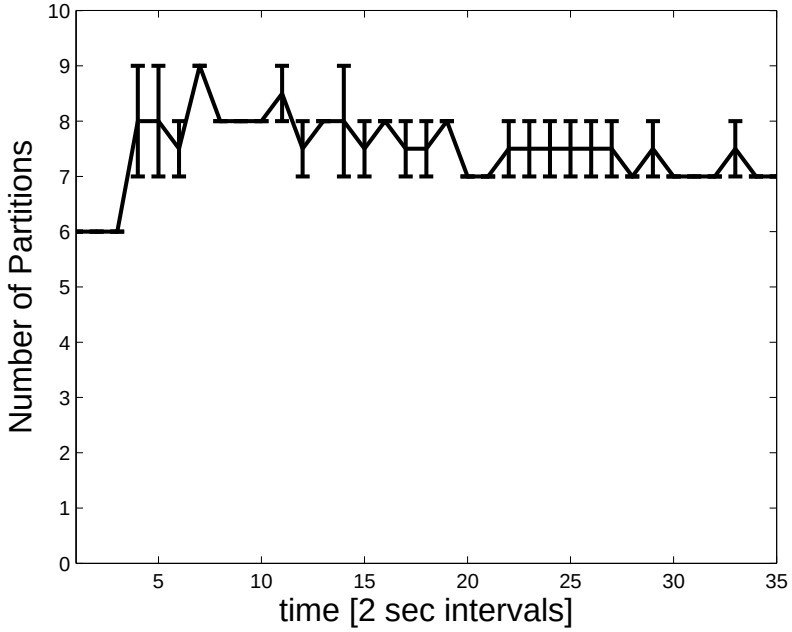


Figure 5.29: Scenario: six targets and three agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

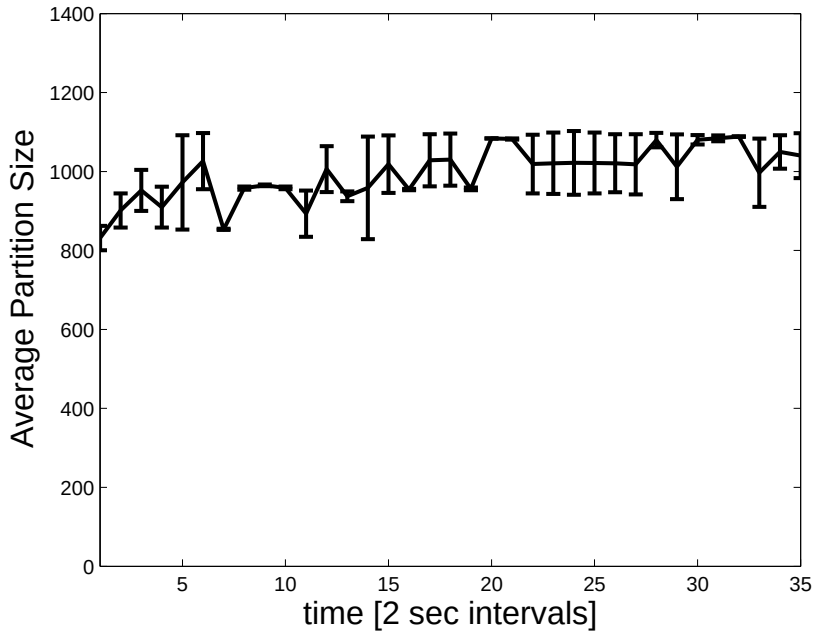


Figure 5.30: Scenario: six targets and three agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

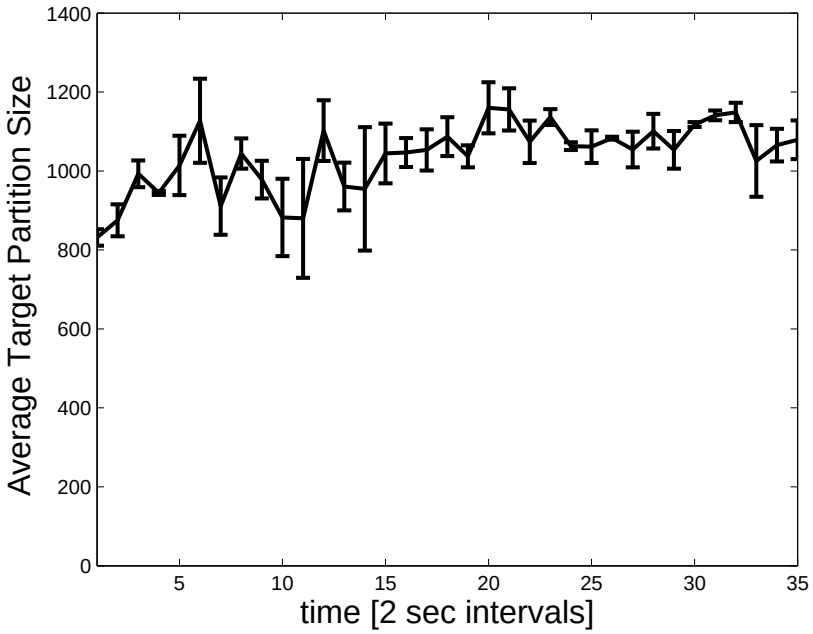


Figure 5.31: Scenario: six targets and three agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean average partition size of partitions containing targets over simulation time. Error bars represent sample standard deviation.

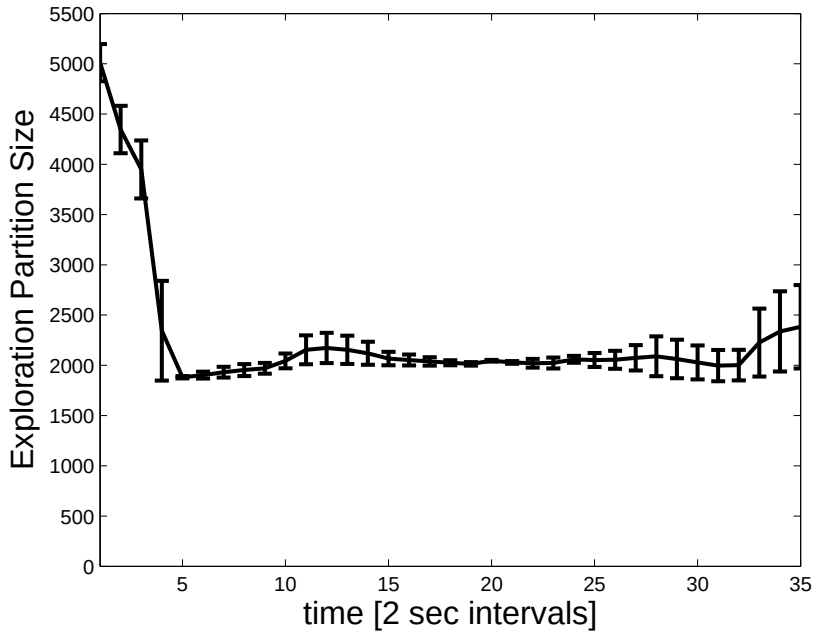


Figure 5.32: Scenario: six targets and three agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean exploration partition size over simulation time. Error bars represent sample standard deviation.

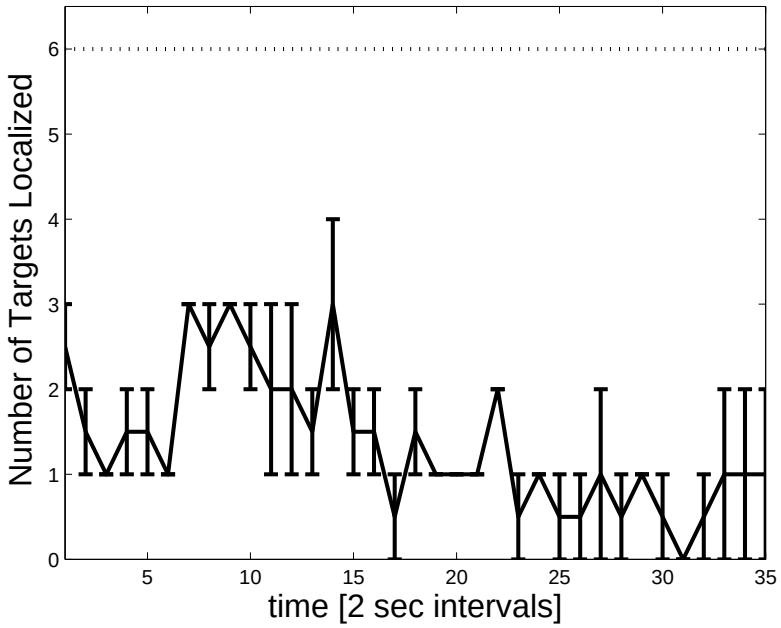


Figure 5.33: Scenario: six targets and three agents. Approach: state-of-the-art direct sensor coverage maximization over the distribution. Sample mean number of targets localized over simulation time. Error bars represent sample standard deviation.

5.6 Time-evolving partition classification Path Planning Performance

The performance of time-evolving partition classification path planning will now be analyzed. This analysis is accomplished in two steps. First, its performance will be compared with the performance of state-of-the-art path planning. To do this, figures will be presented that contain performance measures for both state-of-the-art and time-evolving partition classification path planning. In these figures, error bars are not plotted to reduce clutter and make the figures more readable. The second step is then to present detailed plots of time-evolving partition classification path planning.

5.6.1 Comparison with the State of the Art

As the first step of analyzing the performance of time-evolving partition classification path planning, the performance will be compared with the state of the art. Recall that the performance of the state of the art was analyzed for the scenarios in which

- The number of targets is equal to the number of agents.
- The number of targets is less than the number of agents.
- The number of targets is greater than the number of agents.

And specifically, these scenarios were tested by the cases of 1) there being six targets and six agents, 2) there being three targets and six agents, and 3) there being six targets and three agents. As will be apparent from the figures below, according to these tests, it turns out that the performance of time-evolving partition classification path planning is better than state-of-the-art path planning, according to the presented measures of performance. This is true only for the scenarios presented here.

Comparison: Number of Targets Equal to the Number of Agents

Fig. 5.34 plots the comparison of the sample mean number of targets within search or tracking partitions over simulation time. From this figure it is clear that the partition classification works well to quickly capture all targets, no matter if the path planning approach is the state of the art or time-evolving partition classification. For either case, all targets are quickly found within search or tracking partitions. Fig. 5.35 plots the comparison of the sample mean number of search and tracking partitions over simulation time. Notice that for both path planning methods, the number of partitions appears to be bounded. But, according to Fig. 5.36, the average partition size does not decrease over simulation time for state-of-the-art path planning. However, observe that the average partition size does decrease for time-evolving partition classification path planning. Partition sizes are further analyzed by

considering the average size of partitions containing targets, which is plotted in Fig. 5.37. From this figure it is clear that for the state of the art the average size of partitions containing targets did not decrease over simulation time. In contrast, for time-evolving partition classification the average size of partitions containing targets decreased significantly. Now, consider the exploration partition size. It is desired to have the exploration size decrease over time. In deed, according to Fig. 5.38, for both path planning approaches, the exploration partition size did decrease. However, for the state of the art, at a point in time the exploration partition size began to increase continually. Finally, the number of localized targets is plotted in Fig. 5.39. Note that for good tracking performance of the path planning approaches, the number of localized targets should reach the true number of targets. By the ending time presented in Fig. 5.39, time-evolving partition classification path planning clearly performed best to localize more targets. This is observed by noting that time-evolving partition classification almost localized all six targets. However, the state of the art did not come close to localizing all targets.

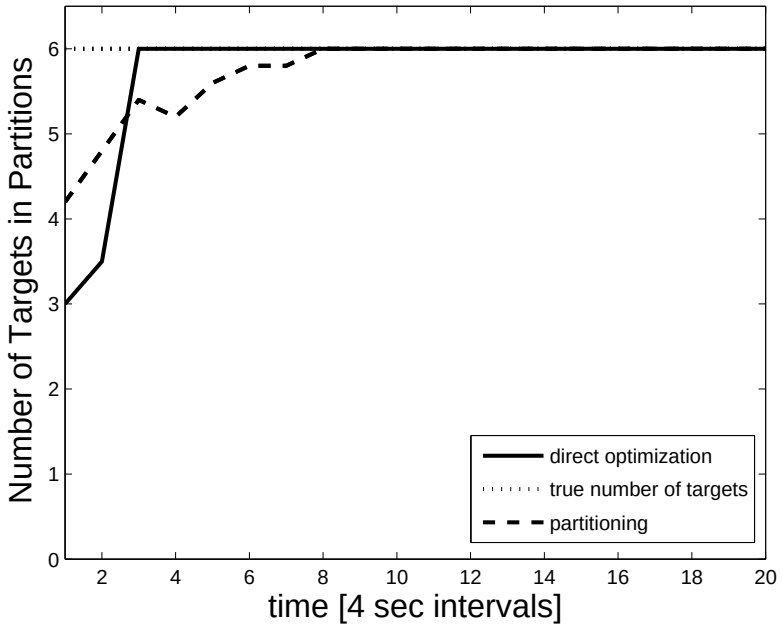


Figure 5.34: Comparison of sample mean number of targets within search or tracking partitions over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and six agents.

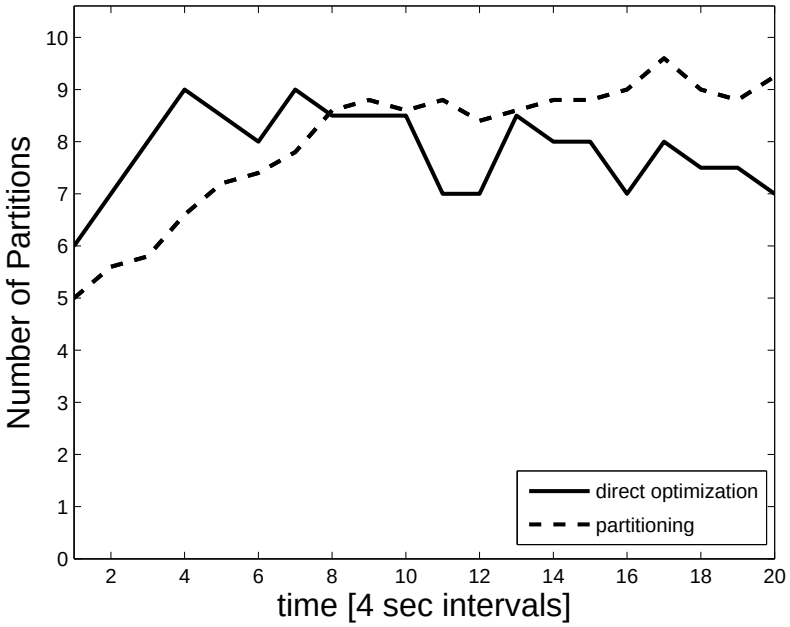


Figure 5.35: Comparison of sample mean number of search and tracking partitions over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and six agents.

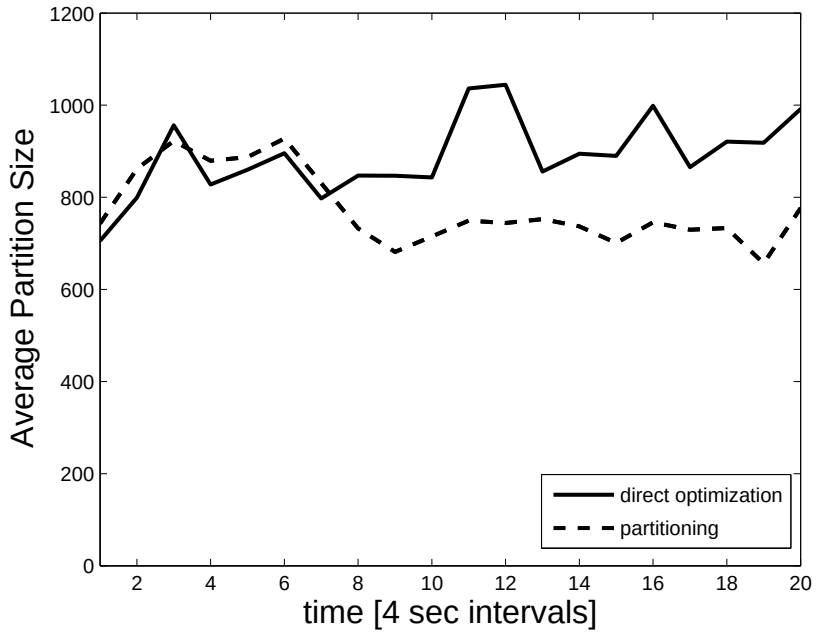


Figure 5.36: Comparison of sample mean average search and tracking partition size over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and six agents.

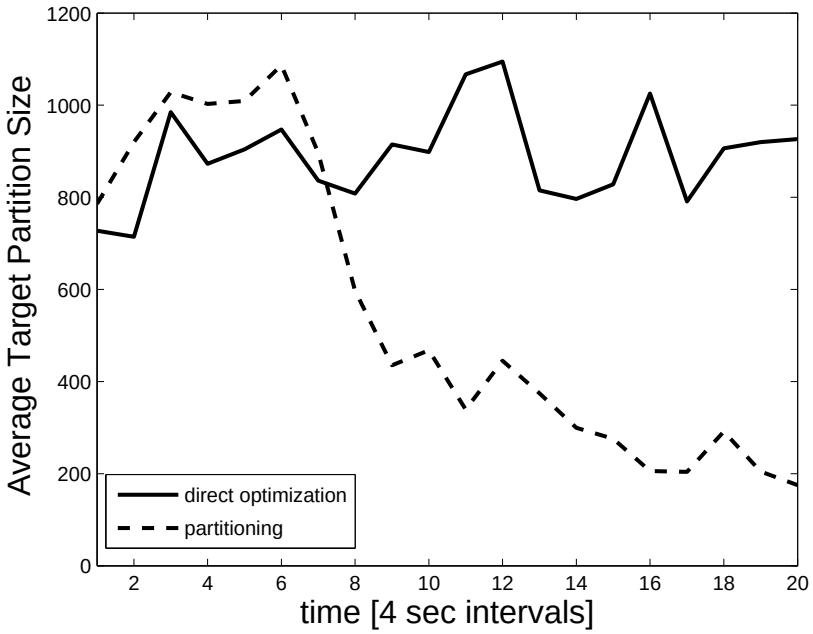


Figure 5.37: Comparison of sample mean average partition size of partitions containing targets over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and six agents.

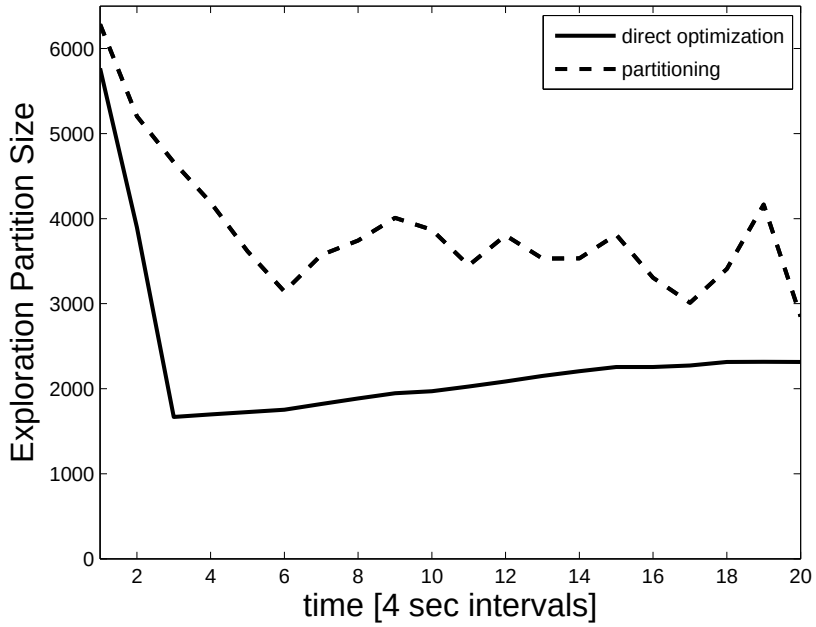


Figure 5.38: Comparison of sample mean exploration partition size over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and six agents.

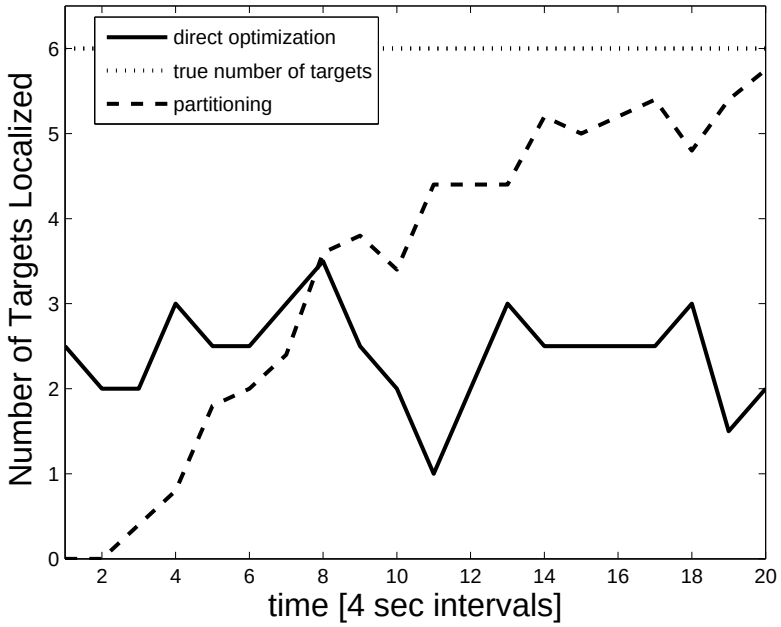


Figure 5.39: Comparison of sample mean number of targets localized over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and six agents.

Comparison: Number of Targets Less than the Number of Agents

For this scenario it was similarly true that the partition classification performed well. This is determined by observing Fig. 5.40 and Fig. 5.41. From these figures it is apparent that all targets are quickly captured within search or tracking partitions and the number of partitions is bounded. However, the performance of the two path planning approaches is very different. From Fig. 5.42, it can be seen that the average partition size does not decrease for state-of-the-art path planning. However, the average partition size decreases substantially for time-evolving partition classification path planning. A similar result is also true for the average size of partitions containing targets, which is plotted in Fig. 5.43. Additionally, according to Fig. 5.44, the exploration size continually decreases for time-evolving partition classification but tends to level off for state-of-the-art path planning. Now, recall that state-of-the-art path planning did not perform well to localize targets in this scenario. In contrast to this, time-evolving partition classification path planning performed well to eventually localize all targets. This can seen in Fig. 5.45. From the results presented here, it is then apparent that time-evolving partition classification path planning performs well to find and localize targets for this scenario, as compared to state-of-the-art path planning.

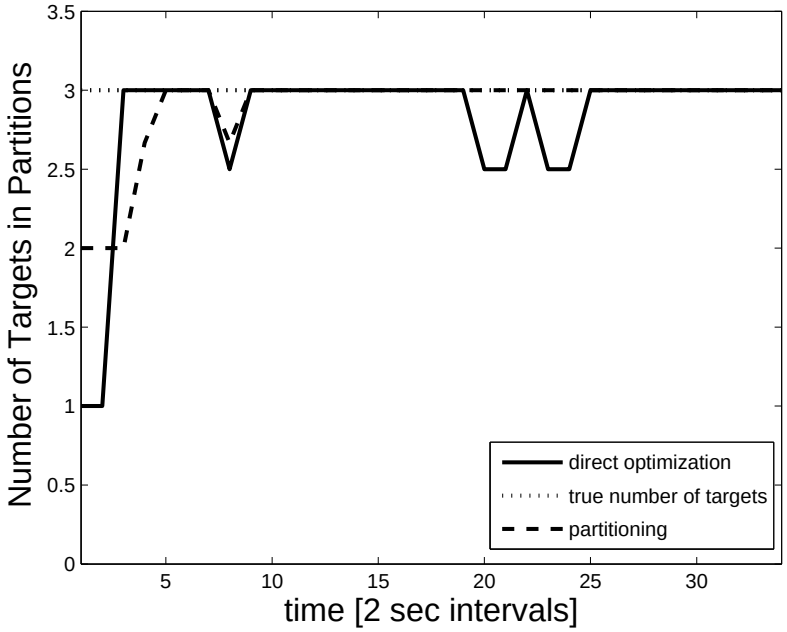


Figure 5.40: Comparison of sample mean number of targets within search or tracking partitions over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being three targets and six agents.

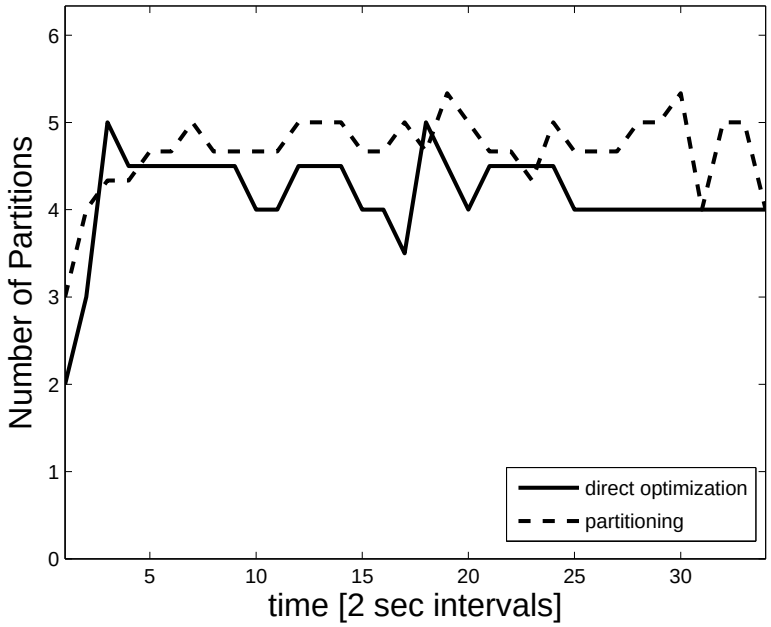


Figure 5.41: Comparison of sample mean number of search and tracking partitions over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being three targets and six agents.

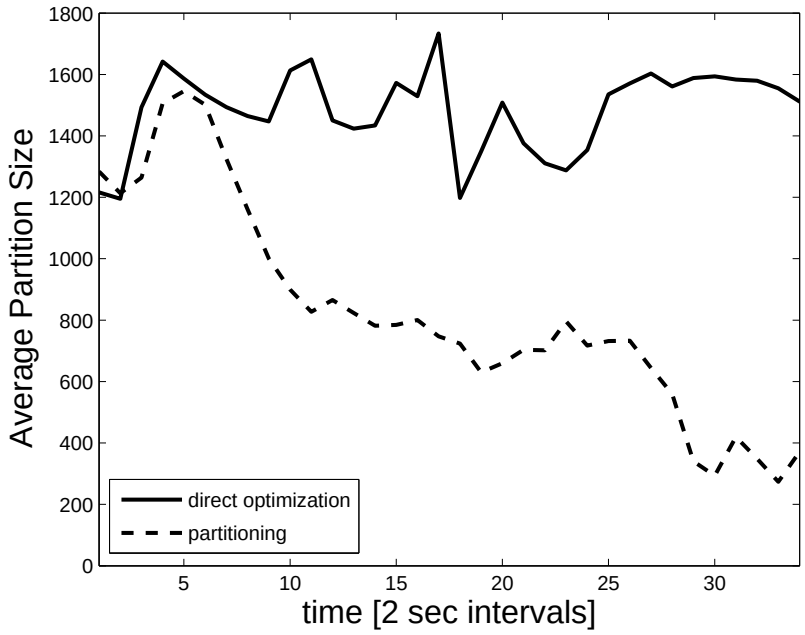


Figure 5.42: Comparison of sample mean average search and tracking partition size over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being three targets and six agents.

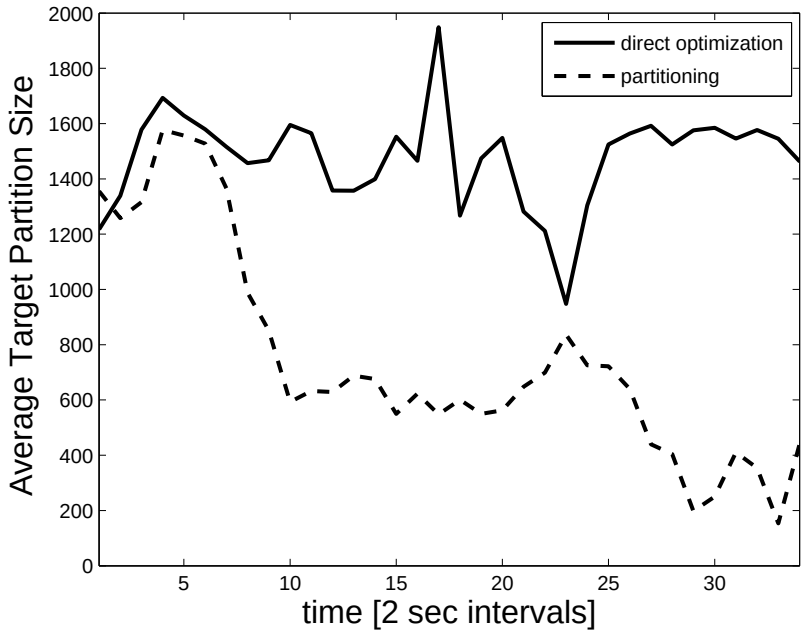


Figure 5.43: Comparison of sample mean average partition size of partitions containing targets over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being three targets and six agents.

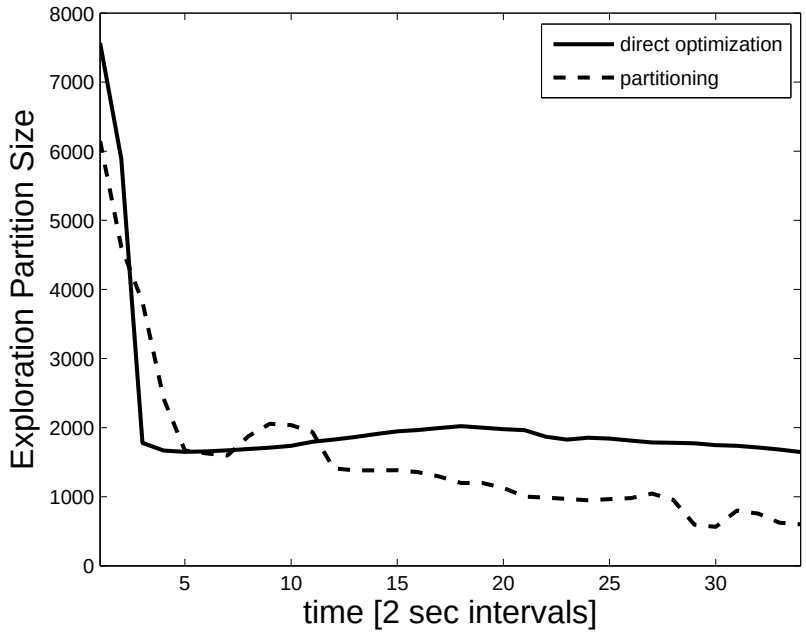


Figure 5.44: Comparison of sample mean exploration partition size over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being three targets and six agents.

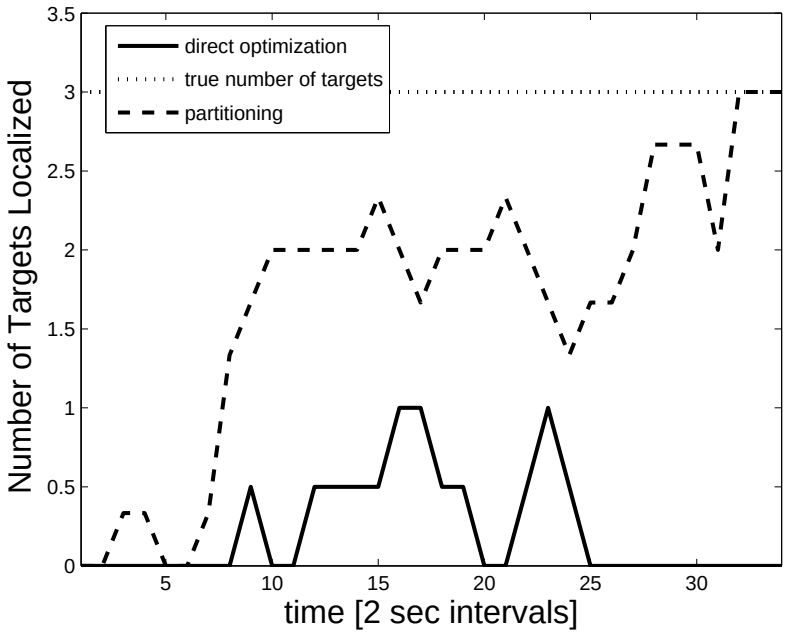


Figure 5.45: Comparison of sample mean number of targets localized over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being three targets and six agents.

Comparison: Number of Targets Greater than the Number of Agents

The final scenario for comparing the performance state-of-the-art path planning and time-evolving partition classification path planning is the case when the number of targets is greater than the number of agents. This is a difficult scenario to address in path planning because a trade-off must be made for switching between targets to track. Yet, once again, the partition classification performed well. This was true for both path planning approaches. This can been seen by observing Fig. 5.40 and Fig. 5.41. These figures show that all targets are quickly captured within search or tracking partitions and the number of partitions is bounded. For this scenario, both path planning approaches performed fairly similar, with a few exceptions. For example, Fig. 5.42 shows that the average partition size increased for state-of-the-art path planning, whereas it decreased for time-evolving partition classification path planning. Similarly, the average size of partitions containing targets increased for state-of-the-art path planning and decreased for time-evolving partition classification path planning. This can be seen in Fig. 5.43. However, the size of the exploration partition was approximately the same for both path planning approaches, as can be seen in Fig. 5.44. And according to Fig. 5.45, the performance of target localization for both approaches is similar. This suggests that when the number of targets is greater than the number of agents, time-evolving partition classification path planning no longer has significantly better performance over state-of-the-art path planning.

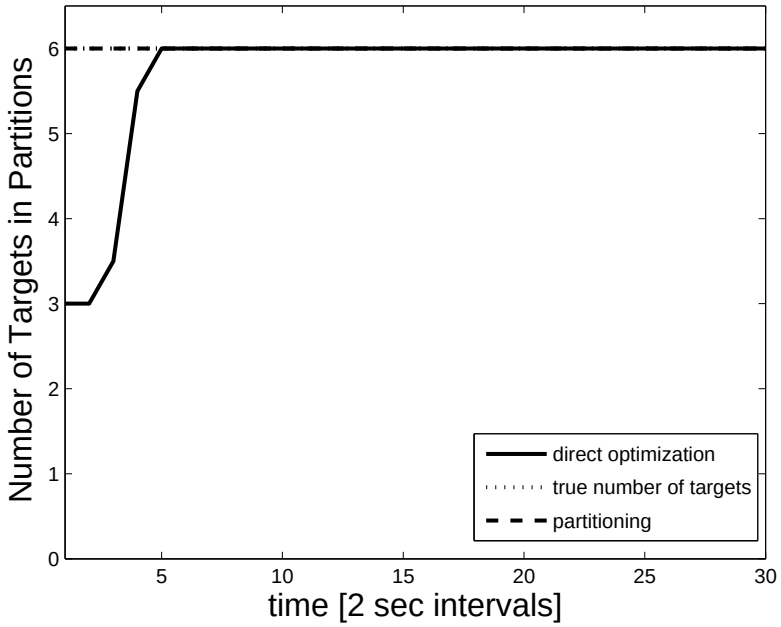


Figure 5.46: Comparison of sample mean number of targets within search or tracking partitions over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and three agents.

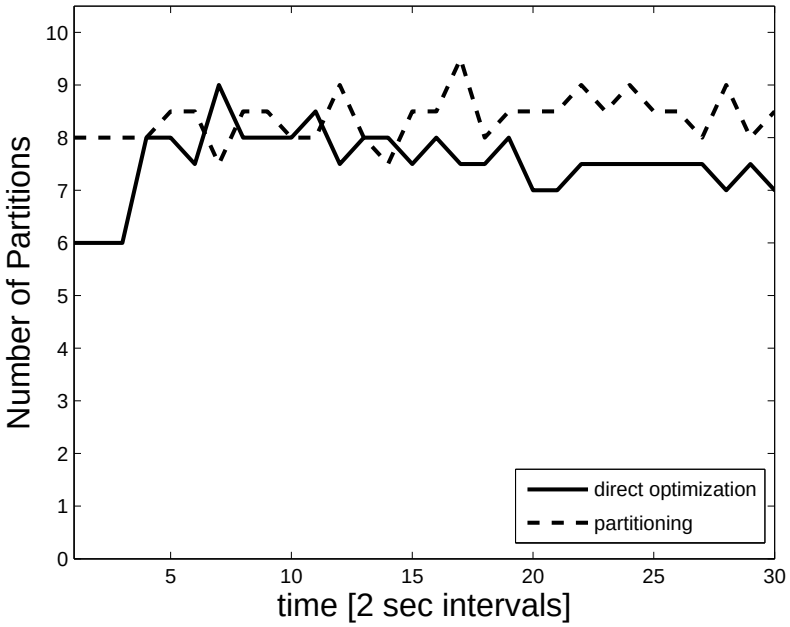


Figure 5.47: Comparison of sample mean number of search and tracking partitions over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and three agents.

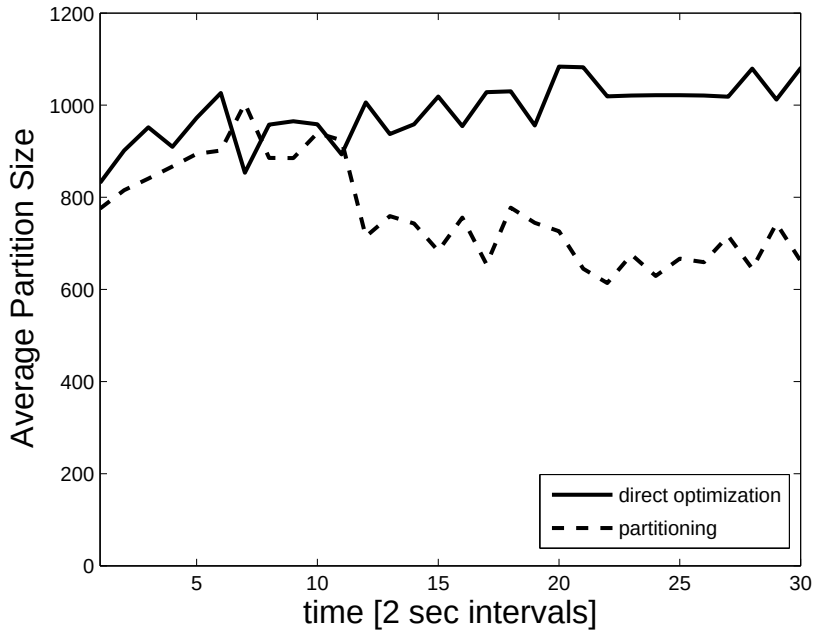


Figure 5.48: Comparison of sample mean average search and tracking partition size over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and three agents.

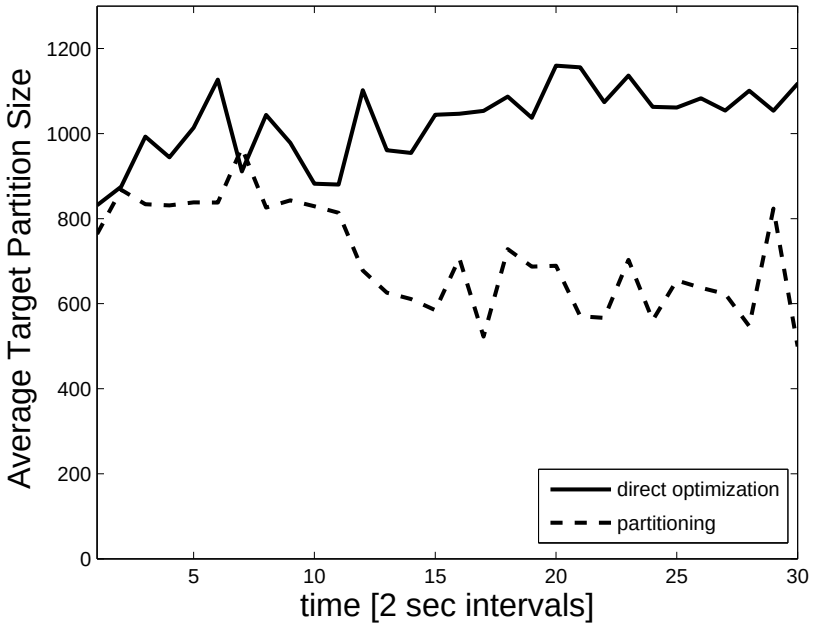


Figure 5.49: Comparison of sample mean average partition size of partitions containing targets over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and three agents.

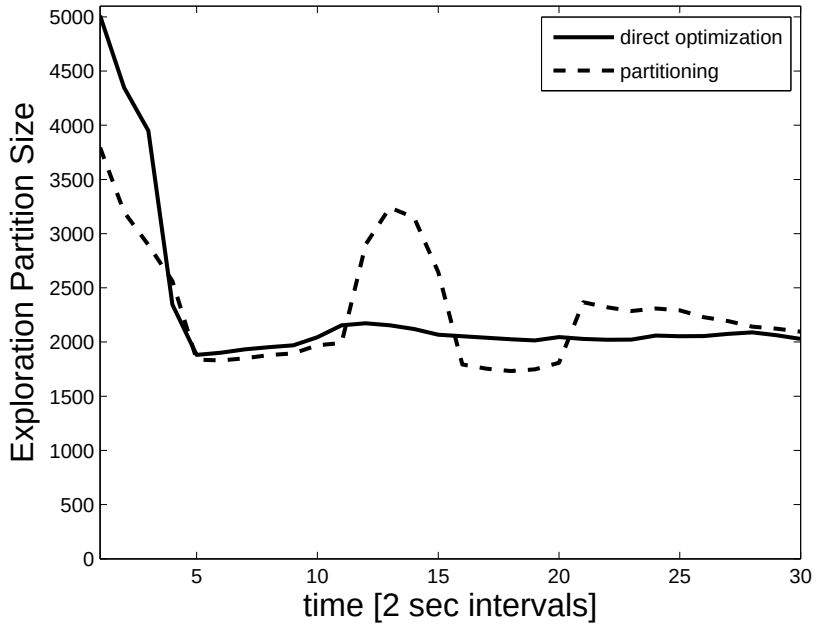


Figure 5.50: Comparison of sample mean exploration partition size over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and three agents.

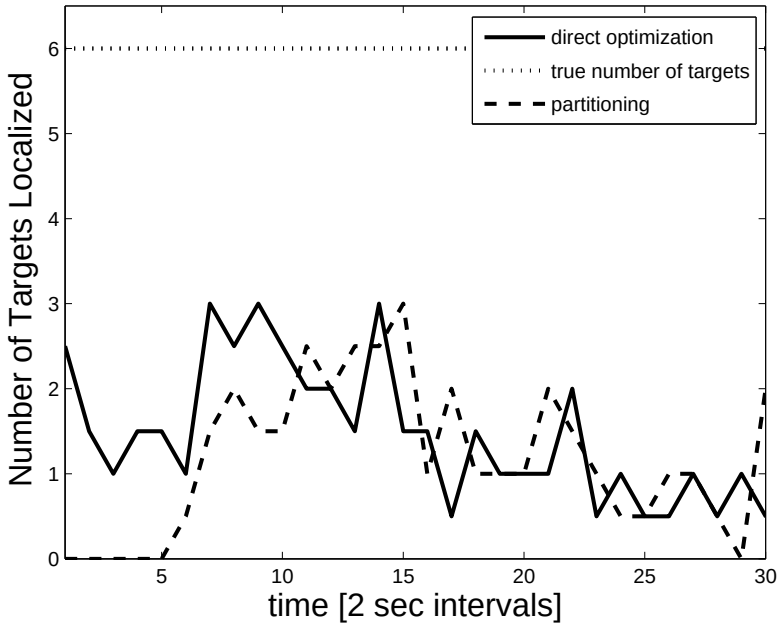


Figure 5.51: Comparison of sample mean number of targets localized over simulation time between state-of-the-art direct distribution sensor coverage maximization and time-evolving partition classification path planning approaches. This plot is for the case of there being six targets and three agents.

5.6.2 Additional Performance Details of Time-evolving partition classification Path Planning

The performance of time-evolving partition classification path planning has been compared to the performance of state-of-the-art path planning. Some of the details were left off of these comparison plots. For example, error bars representing the sample standard deviation were not shown so that the plots would be easily read. So, in this section, the results of time-evolving partition classification path planning will be considered in more detail.

A complete presentation of results would consist of permutations of the various number of agents and number of targets. Although not all of these conditions can be thoroughly tested, samples and results are presented based on scenarios that are considered representative of conditions that occur often. These scenarios consist of

- The number of targets equals the number of agents.
- There are more targets than agents.
- There are less targets than agents.
- There are zero targets.

For each of these cases, several simulation samples were performed with various initial target and agent positions as well as various prior target density distributions. Before presenting the performance for each of these scenarios, consider a sequence of the partitioning over time for a sample simulation. In order to understand the objects that will be plotted in this sequence, see Fig. 5.52, which provides labels for objects that will be seen in the sequence. Then see Fig. 5.53 for a sample sequence.

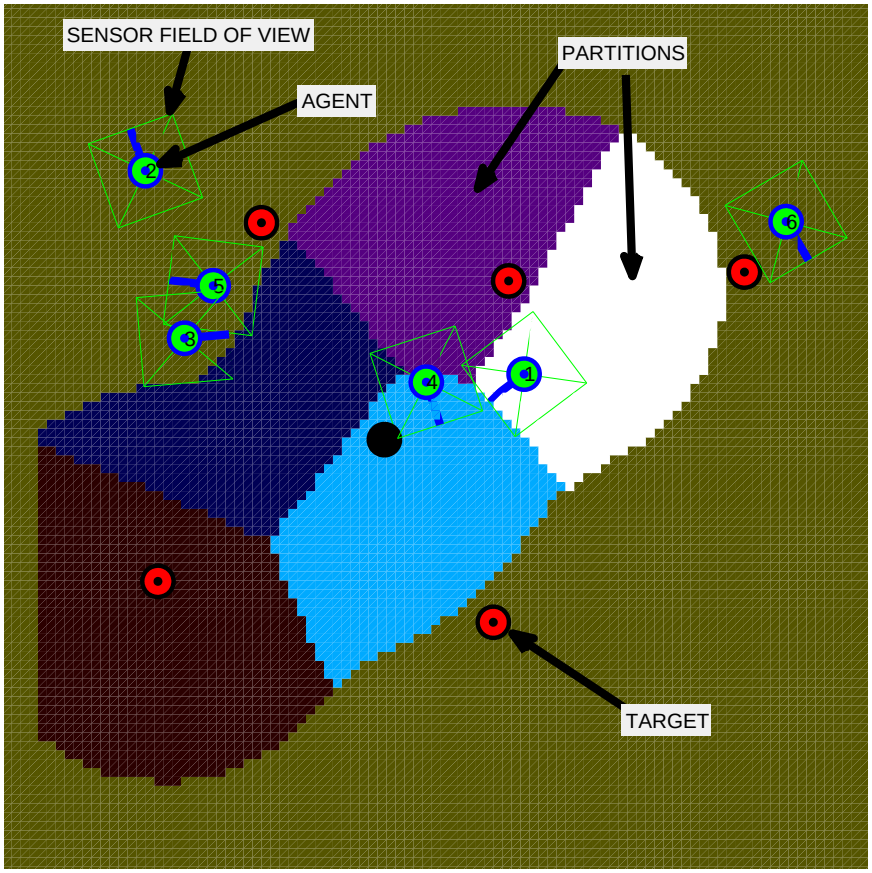
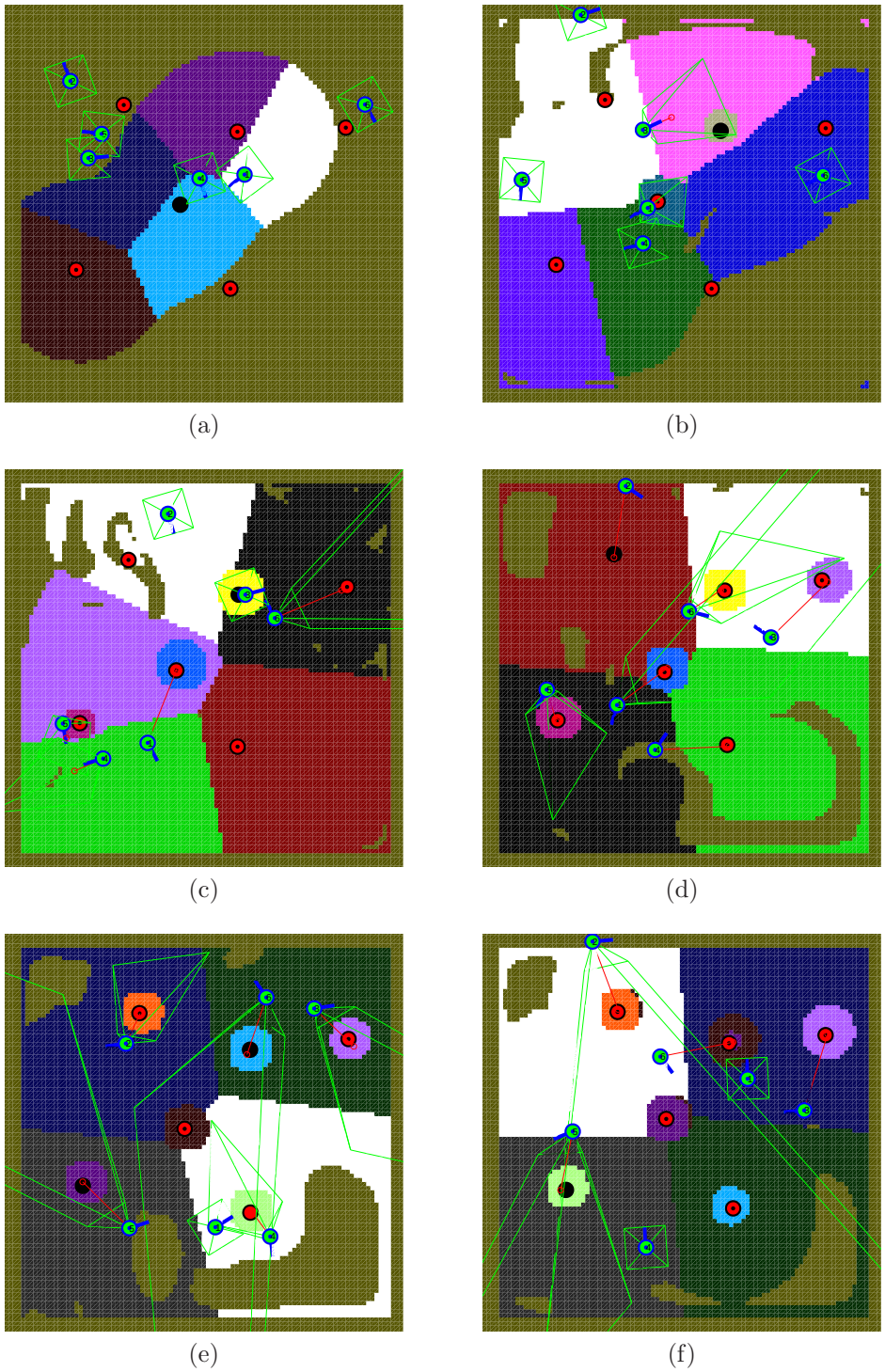


Figure 5.52: Description of the objects presented in the sample simulation sequence of time-evolving partition classification path planning over simulation time. Agents are represented as circles with protruding lines. Sensor fields of view are represented as additional thin lines around the agents. Targets are represented as circles without protruding lines. Partitions are represented by variations in color across the search area.



(a) (b) (c) (d) (e) (f)

Figure 5.53: Sequence of time evolving partitions for a sample simulation involving six agents and six targets (the number of targets was unknown to the agents). Follow the sequence from left to right and from top to bottom. In (a) initial partitions are formed. In (b) tracking partitions appear. By (f) all targets are within tracking partitions.

5.6.3 The Number of Targets Equals the Number of Agents

As an immediate check to see if the surveillance area is being partitioned to capture the targets, the sample mean number of targets within search and tracking partitions is plotted in Fig. 5.54. From Fig. 5.54, observe that all targets are eventually within search and tracking partitions.

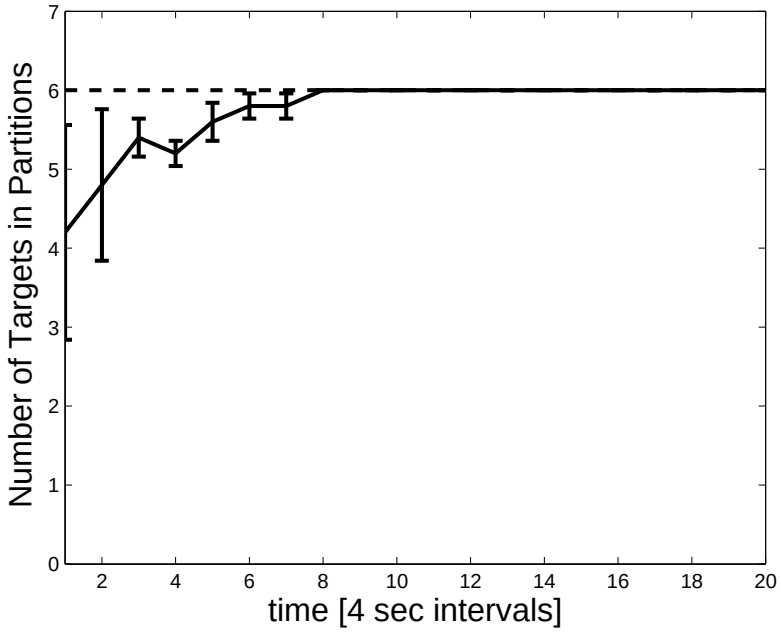


Figure 5.54: Scenario: six targets and six agents. Approach: time-evolving partition classification. Sample mean number of targets within search and tracking partitions over simulation time. Error bars represent sample standard deviation.

To further analyze the search and tracking partitions, consider their quantity and average size over simulation time. The sample mean number of search and tracking partitions are plotted in Fig. 5.55. Observe that the number of search and tracking partitions gradually increases but appears to level out. To make sense of this gradual increase in number of partitions, the sample mean average size of the search and tracking partitions is plotted in Fig. 5.56. From Fig. 5.56 it is observed that the average partition size tends to remain constant. The only way for the number of partitions to increase and the average size of the partitions to remain constant is for the additional partitions to be small tracking partitions and to have the average size of the search partitions increase. This is confirmed by observing the sample mean average partition size of partitions in which targets exist, plotted in Fig. 5.57. In these simulations, there were six agents and six targets. Consequently, once six targets are localized within tracking partitions and are being tracked by six agents, the

remaining search partitions will persist unless additional agents come to search those partitions. This particular control choice was made simply as an initial test. However, for this scenario it worked well. However, a more general searching control would be implemented to get better coverage of the surveillance area by temporarily abandoning a tracked target. In fact, this performance of this type of control will be presented for the scenario of the number of targets being greater than the number of agents.

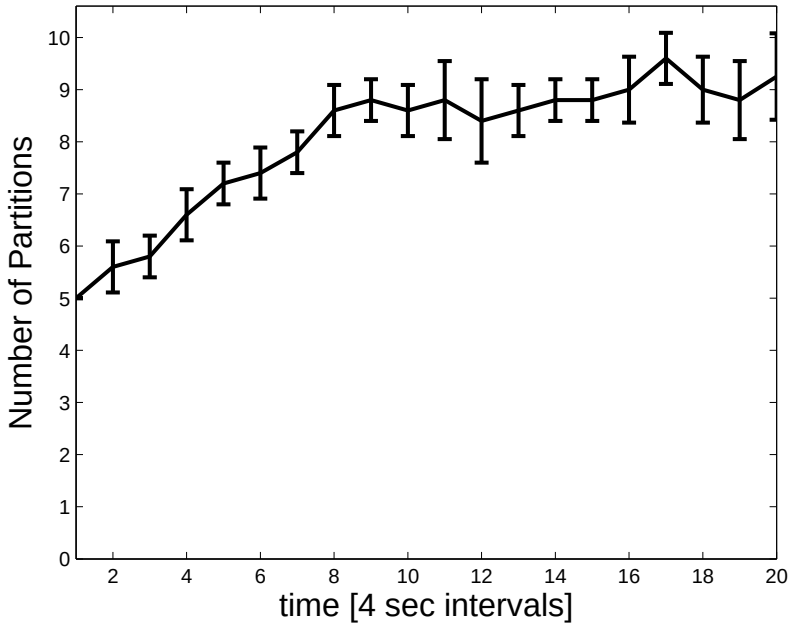


Figure 5.55: Scenario: six targets and six agents. Approach: time-evolving partition classification. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

Path planning is affected by the performance of time-evolving partition classification. Agent paths are determined by task allocation over the partitions and then path optimization directly over partition level target density distributions. The performance of path planning is analyzed by observing statistics of the number of localized targets over simulation time. Fig. 5.58 shows the sample mean number of localized targets over simulation time. As observed in Fig. 5.58 the general trend was to approach localization of all targets by the end of the simulation. This result suggests path planning based on time-evolving partition classification performs well to search for an unknown number of targets.

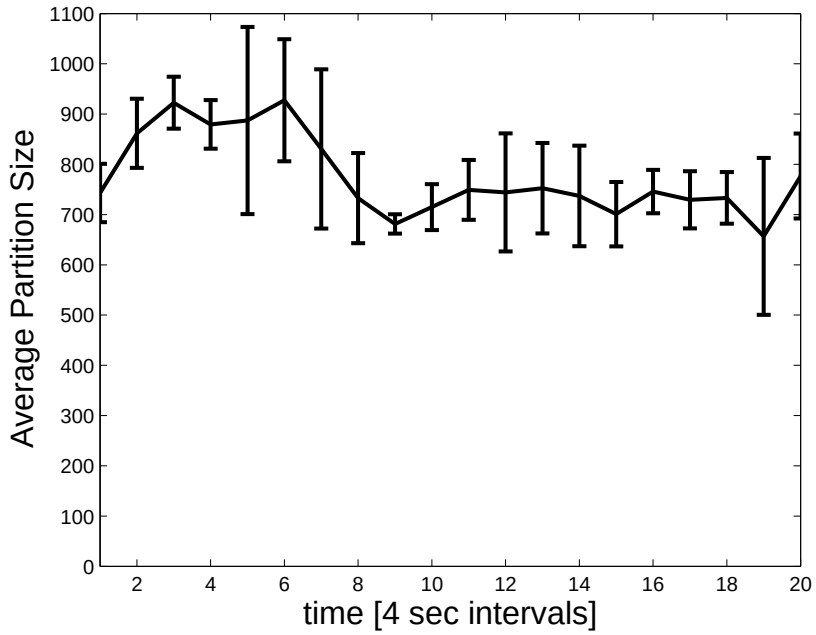


Figure 5.56: Scenario: six targets and six agents. Approach: time-evolving partition classification. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

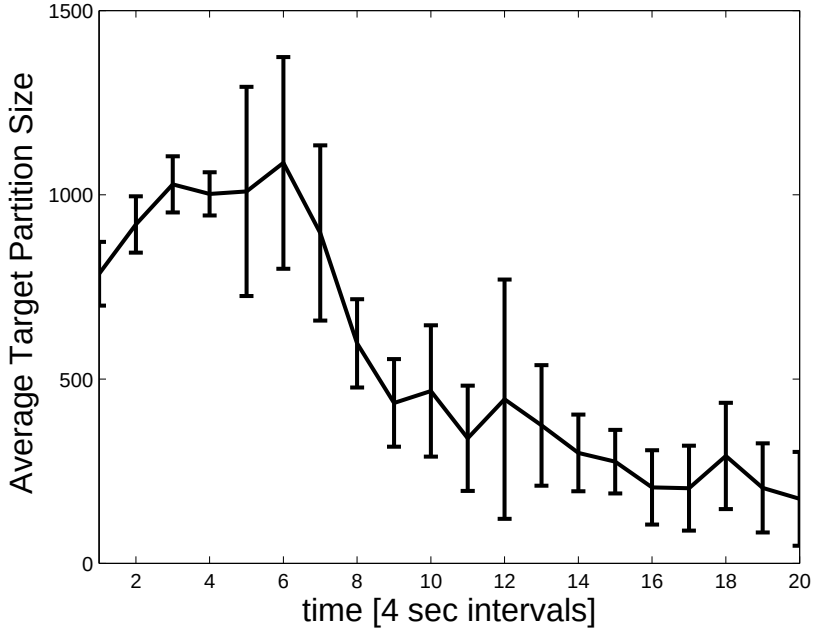


Figure 5.57: Scenario: six targets and six agents. Approach: time-evolving partition classification. Sample mean average partition size of partitions containing targets over simulation time. Error bars represent sample standard deviation.

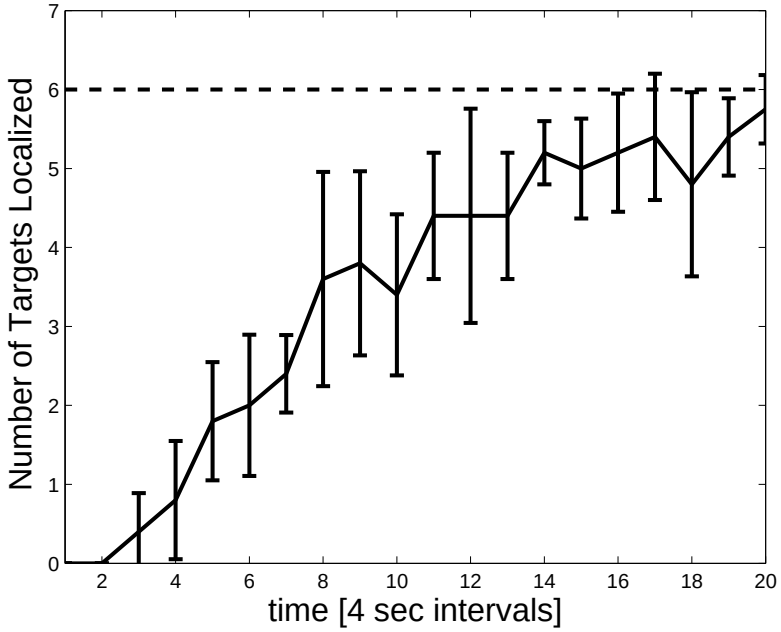


Figure 5.58: Scenario: six targets and six agents. Approach: time-evolving partition classification. Sample mean number of targets localized over simulation time. Error bars represent sample standard deviation.

5.6.4 Number of Targets Greater than Number of Agents

Recall that the scenario for equal number of targets and agents was implemented with a path planning choice that led to the agents attempting to track any targets that are detected, instead of attempting to continue to search the rest of the surveillance area. As a consequence, targets were tracked. However, the search partitions essentially never changed in size. This scenario, the one for which the number of targets is greater than the number of agents, poses a need to continue to search the rest of the surveillance area, even after a target has been detected. This need is realized because the number of targets exceeds the number of agents. In particular, for this case, the number of target was six and the number of agents was three. In this case the path planning choice was to given some time to detected targets but to then search out other search partitions, possibly returning to previously detected targets. Fig. 5.59 presents the number of targets found in search and tracking partitions. Note that eventually it is desired for this to be the true number of targets. Notice that in Fig. 5.59 this goal is reached at the very start of the simulation. The reason for this is because the prior target density distribution has a high expected number of targets and has characteristic shape over most of the surveillance area. Consequently, most regions within the surveillance area start out well classified in terms of eventually discovering each target. Fig. 5.60 presents the number of search and tracking partitions over time. Observe that this number stays approximately constant over the short simulation time. Fig. 5.61 presents the average search and tracking partition size over time. Notice that, as desired, this number does decrease a little over the simulation time. Fig. 5.62 presents the average partition size of partitions that contain targets. This figure more explicitly focuses on the desire to have this number decrease over time. Note that it does appear to decrease. However, much more simulation time would be required to see that this number would indeed decrease sufficiently low for this scenario. Fig. 5.63 presents the exploration partition size over simulation time. Interestingly this number appears to oscillate a bit. Fig. 5.64 presents the null target partitions size over simulation time. As desired, this number increases over time. However, its increase is limited by the high target density values and anticipated target motion. Fig. 5.65 presents the number of localized targets over time. This figure is actually very indicative of the process that occurred with this scenario. Targets were detected early, the certainty of these targets remained high enough to wait until new targets were detected. More targets were then detected as the certainty of the first targets decreased.

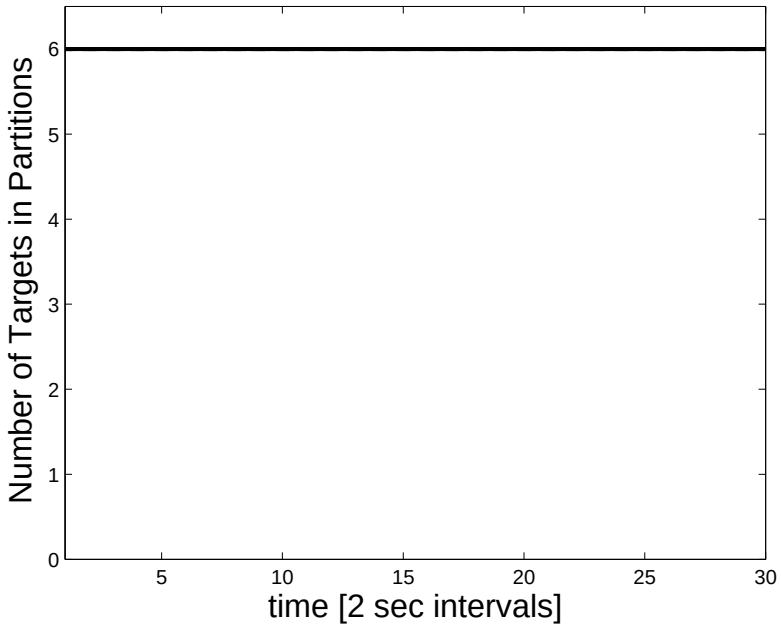


Figure 5.59: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean number of targets within search and tracking partitions over simulation time. Error bars represent sample standard deviation.

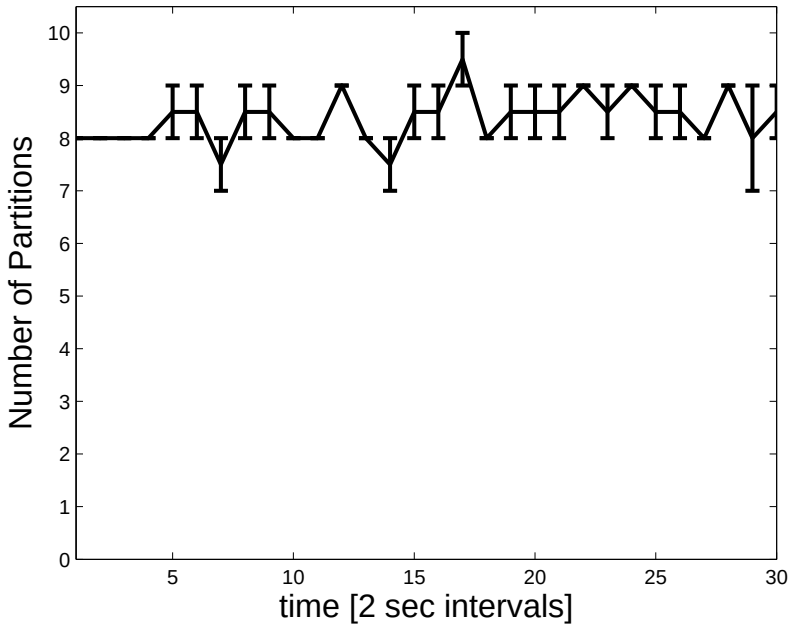


Figure 5.60: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

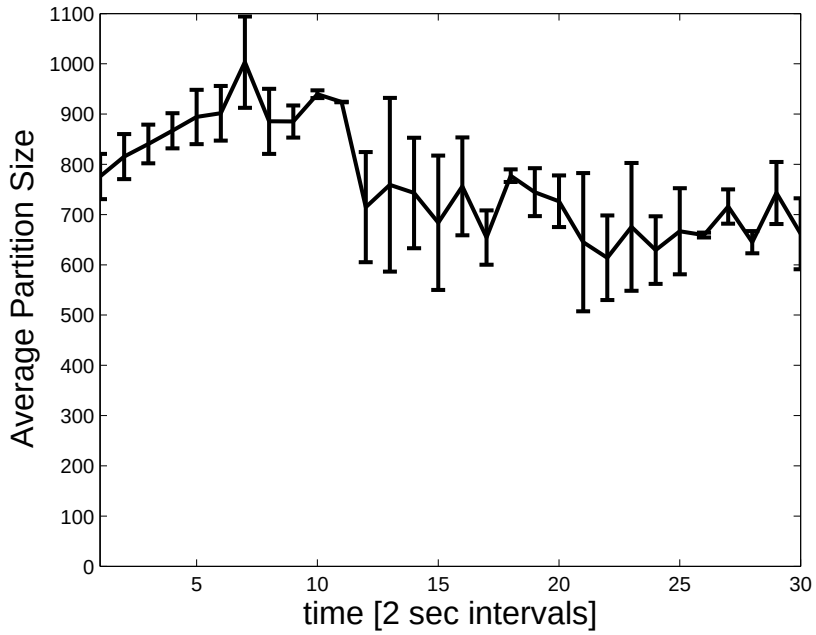


Figure 5.61: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

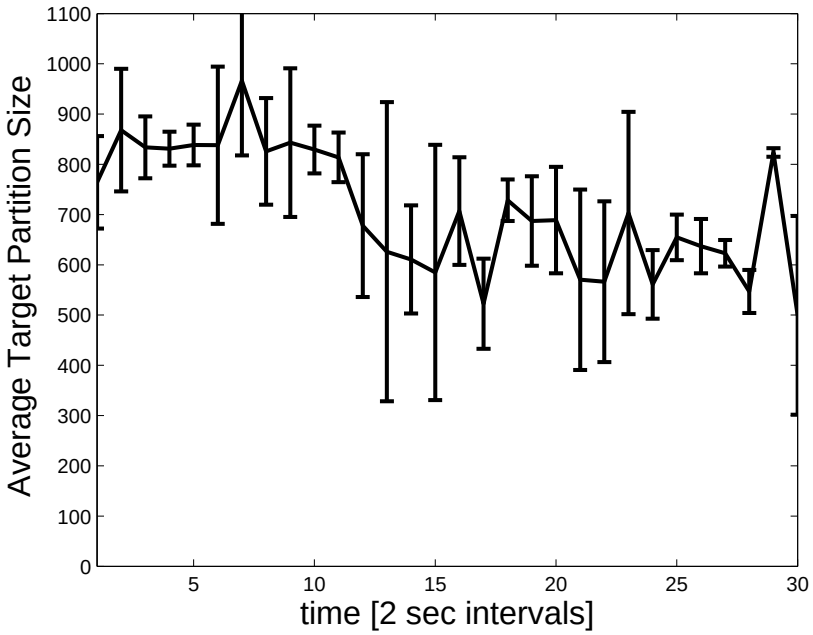


Figure 5.62: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean average partition size of partitions containing targets over simulation time. Error bars represent sample standard deviation.

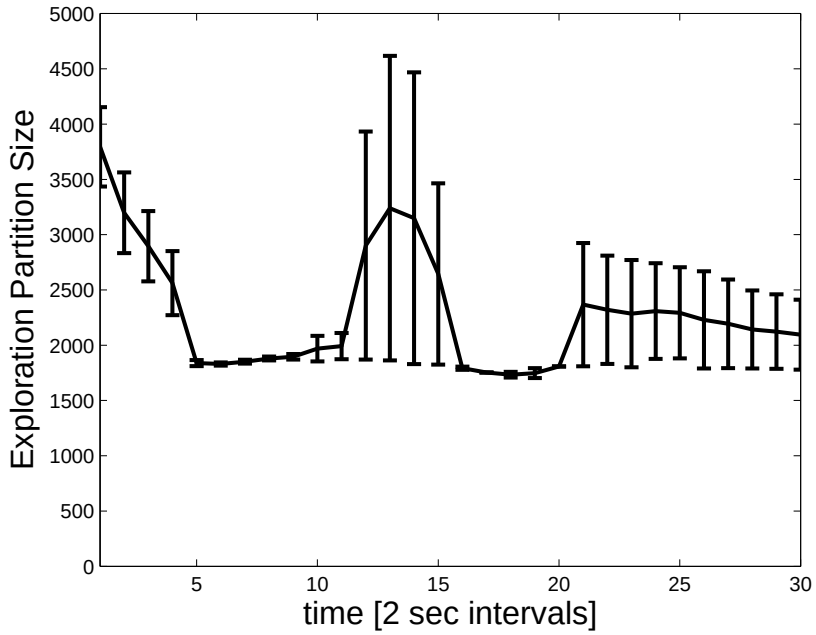


Figure 5.63: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean exploration partition size over simulation time. Error bars represent sample standard deviation.

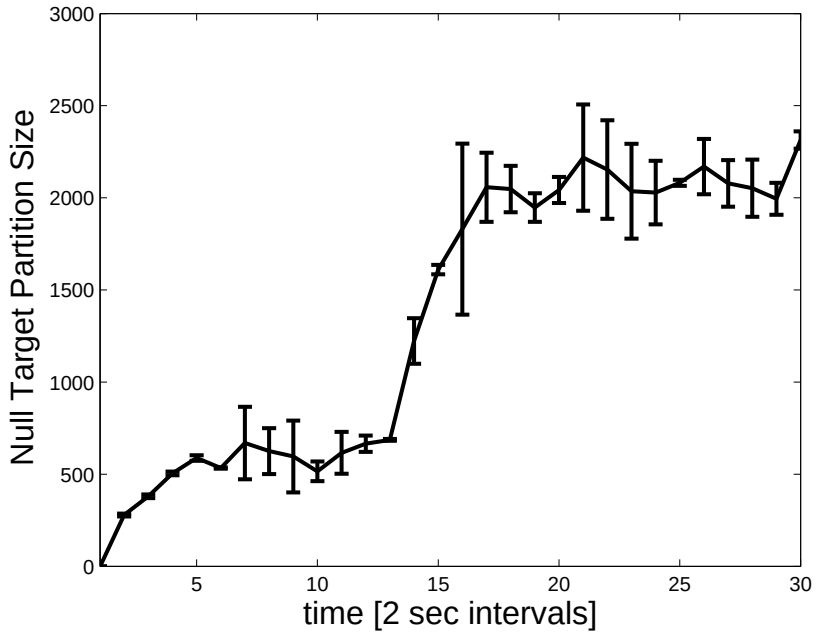


Figure 5.64: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean null target partition size over simulation time. Error bars represent sample standard deviation.

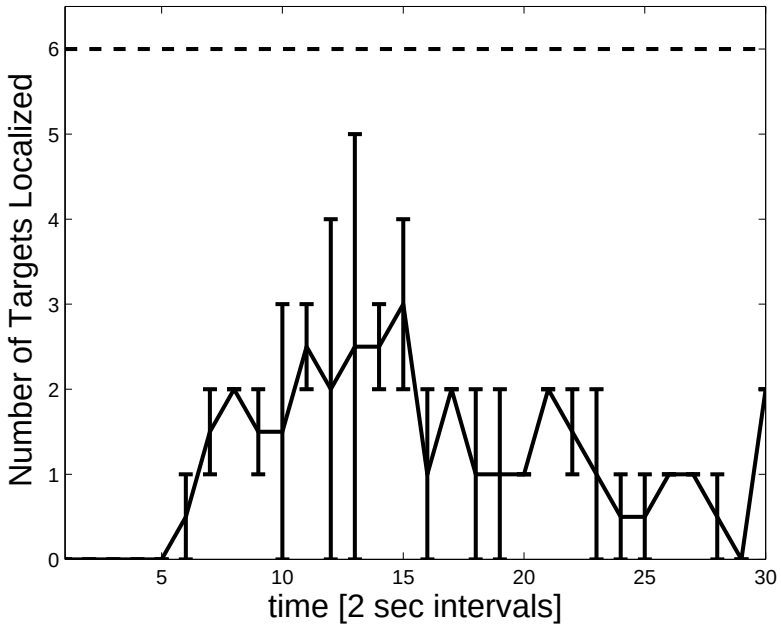


Figure 5.65: Scenario: six targets and three agents. Approach: time-evolving partition classification. Sample mean number of targets localized over simulation time. Error bars represent sample standard deviation.

5.6.5 Number of Targets Less than Number of Agents

The scenario of the number of target being less than the number of agents, with the specific case in which there are three targets and six agents, was tested. Judging from the previous scenario in which there were more targets than agents, it is natural to assume that the maintainability of target localizations will be easier in this scenario. In fact, the data suggests this is true. Fig. 5.66 presents the number of targets within search and tracking partitions over time. Over time this value should reach the true number of targets, as more of the surveillance area is searched. Along with the number of targets in search and tracking partitions, the total number of search and tracking partitions should be analyzed, which is presented in Fig. 5.67. Then to understand how all of the partitions are varying over time, they are presented in Fig. 5.68 for the sample mean average search and tracking partition size, Fig. 5.70 for the sample mean exploration partition size, and Fig. 5.71 for the null target partition size.

Ideally, over time it is desired for the average search and tracking partition size to decrease (ultimately reaching the point of these partitions consisting solely of tracking partitions). Additionally, it is desired for the exploration partition size to eventually vanish. Likewise, the null target partition size should eventually cover most of the surveillance area. In fact, the ideal performance would consist of a final partitioning formed by nothing but tracking partitions and a null target partition. Observe that this appears to be the general trend in the data for this scenario in which there were three targets and six agents. This trend is further seen by observing the sample mean number of localized targets over time, in Fig. 5.72.

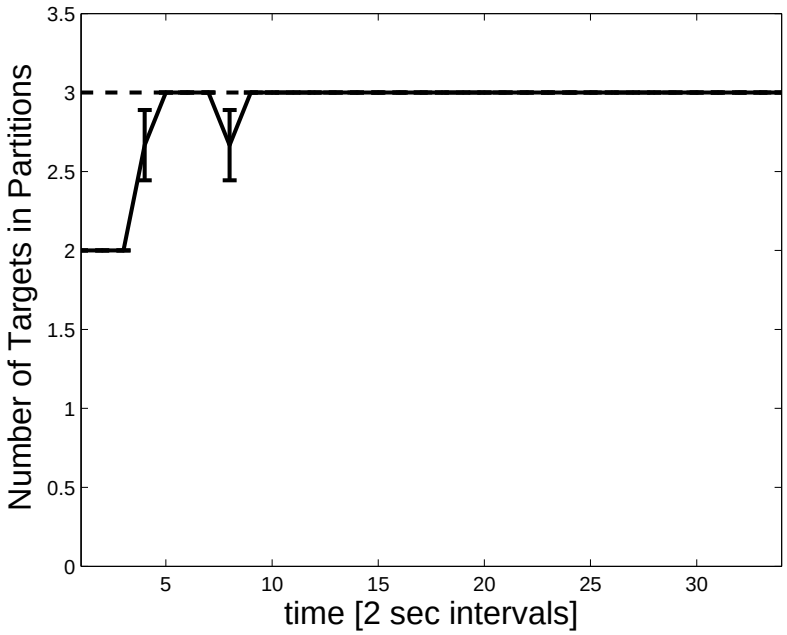


Figure 5.66: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean number of targets within search and tracking partitions over simulation time. Error bars represent sample standard deviation.

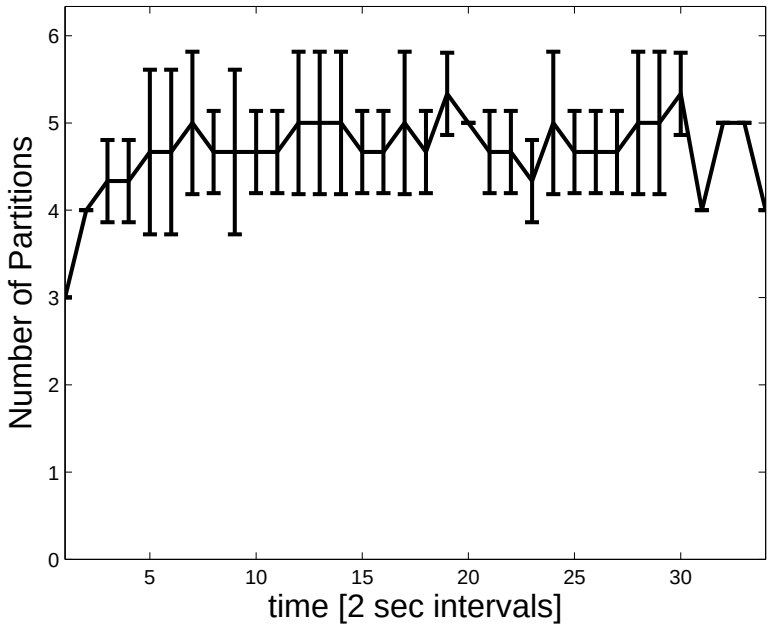


Figure 5.67: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

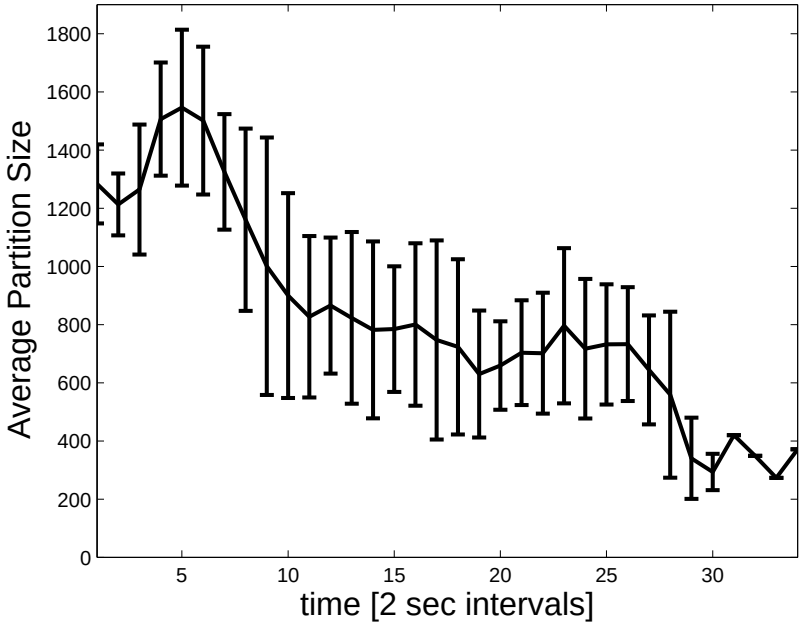


Figure 5.68: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

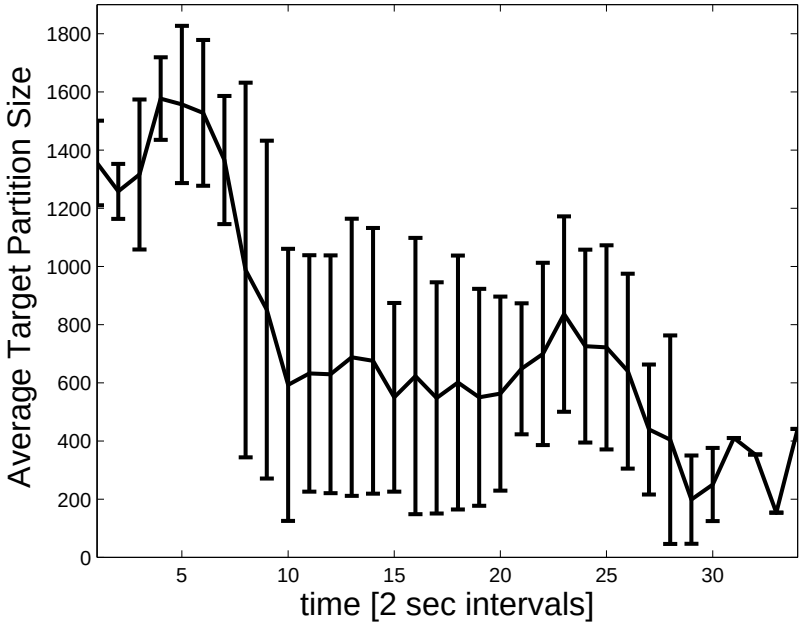


Figure 5.69: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean average partition size of partitions containing targets over simulation time. Error bars represent sample standard deviation.

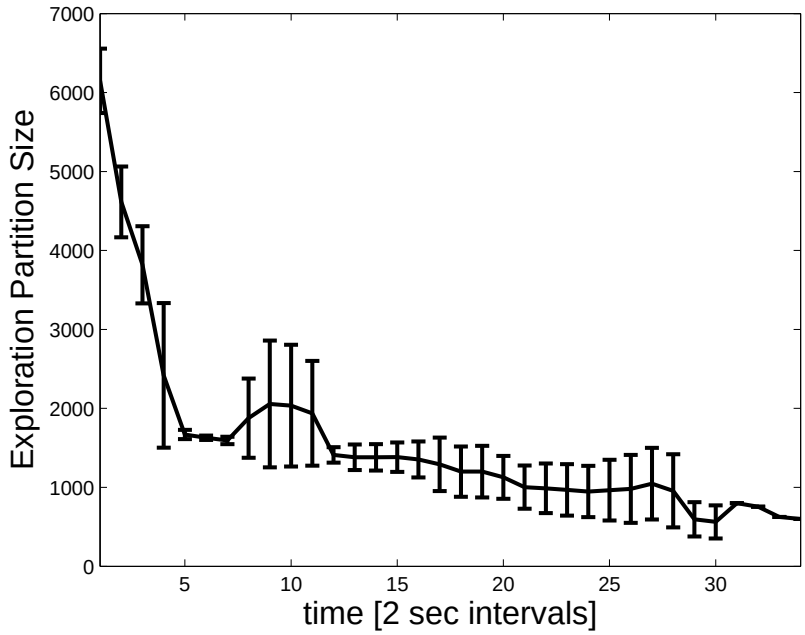


Figure 5.70: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean exploration partition size over simulation time. Error bars represent sample standard deviation.

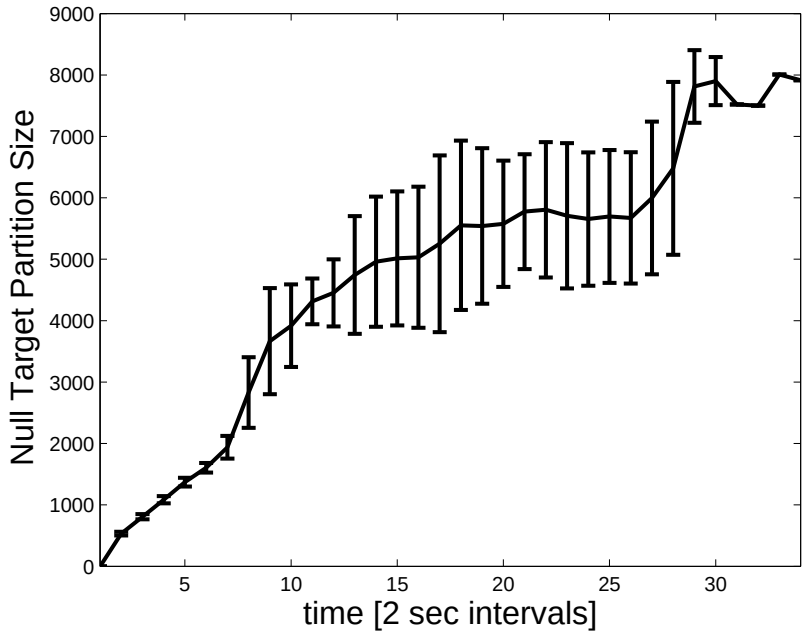


Figure 5.71: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean null target partition size over simulation time. Error bars represent sample standard deviation.

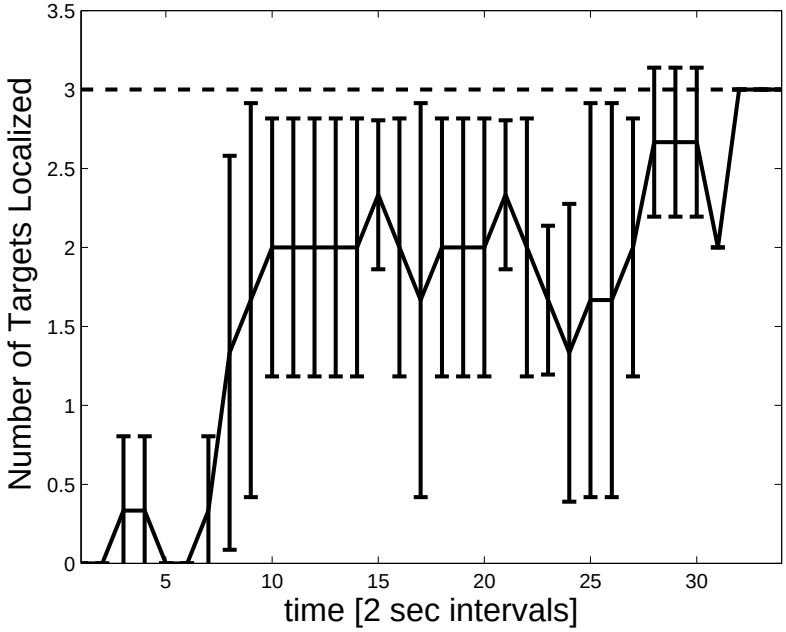


Figure 5.72: Scenario: three targets and six agents. Approach: time-evolving partition classification. Sample mean number of targets localized over simulation time. Error bars represent sample standard deviation.

5.6.6 Zero Targets in Surveillance Area

The scenario of zero targets being in the surveillance is an interesting scenario because it isolates the partition classification and path planning performance for eventually determining that the entire surveillance area is a null target partition and that no targets exist. As such, the type of path planning plays little influence for this scenario. This section is then mainly to check that time-evolving partitioning works well for the case when there are actually zero targets. To analyze the performance for this scenario, a smaller set of statistics is needed. These statistics are

- Number of search and tracking partitions.
- Average search and tracking partition size.
- Exploration partition size.
- Null target partition size.

For the performance to be good for this scenario, the average search and tracking partition size should eventually go to zero. Similarly, the exploration partition size should go to zero. Yet, the null target partition size should eventually reach the point in which it is defined over the entire surveillance area. In fact, the data verified this behavior to be the case, as can be seen in the figures below. Fig. 5.73 presents the number of search and tracking partitions. Fig. 5.74 presents the average search and tracking partition size. Observe that there is initially zero search and tracking partitions. This is because the entire surveillance area starts out as an exploration partition. Then, as information is obtained, the exploration partition eventually transitions to a general search partitions. Fig. 5.75 presents the size of the exploration partition over time. Observe that this partition is headed toward zero over time, as the surveillance area is searched. Fig. 5.76 presents the null target partition size over time. For this scenario this is the most important statistic because the goal is for this partition to eventually reach begin defined over the entire surveillance area.

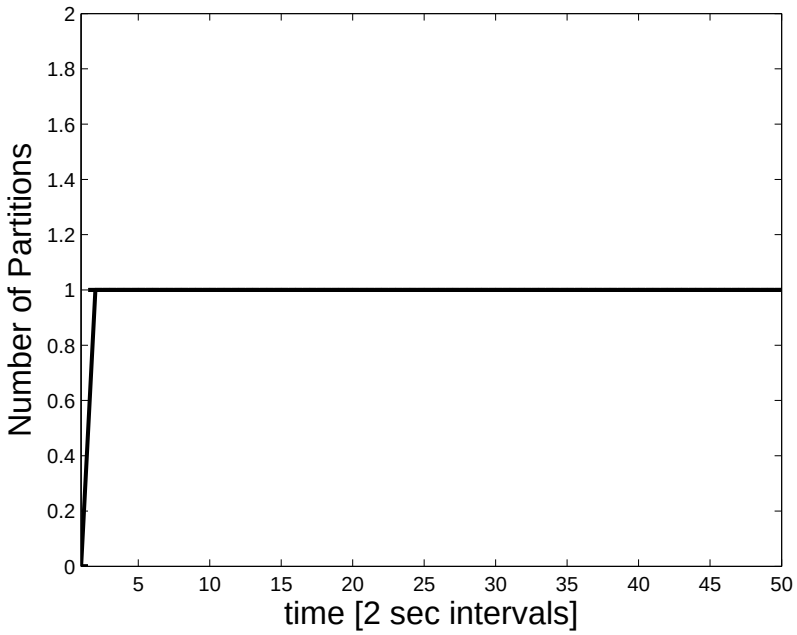


Figure 5.73: Scenario: zero targets and six agents. Approach: time-evolving partition classification. Sample mean number of search and tracking partitions over simulation time. Error bars represent sample standard deviation.

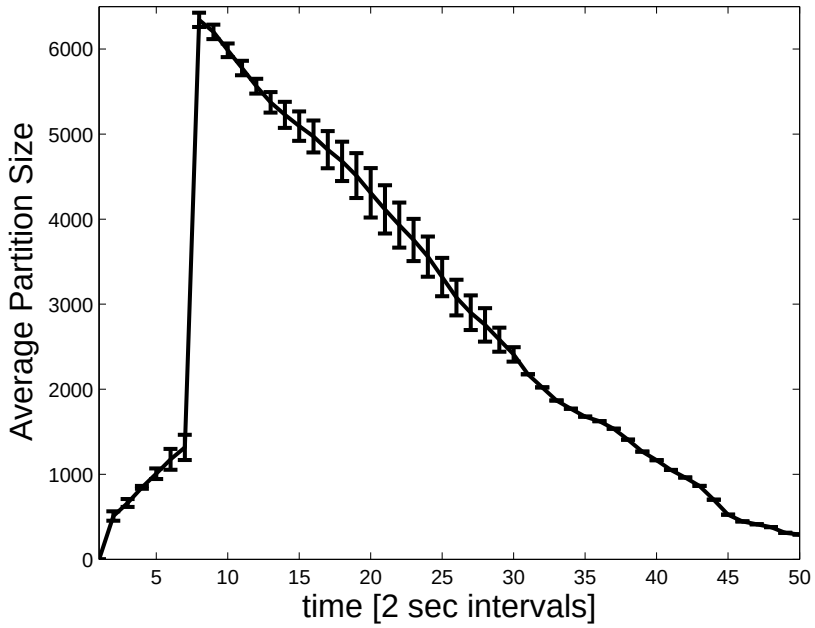


Figure 5.74: Scenario: zero targets and six agents. Approach: time-evolving partition classification. Sample mean average search and tracking partition size over simulation time. Error bars represent sample standard deviation.

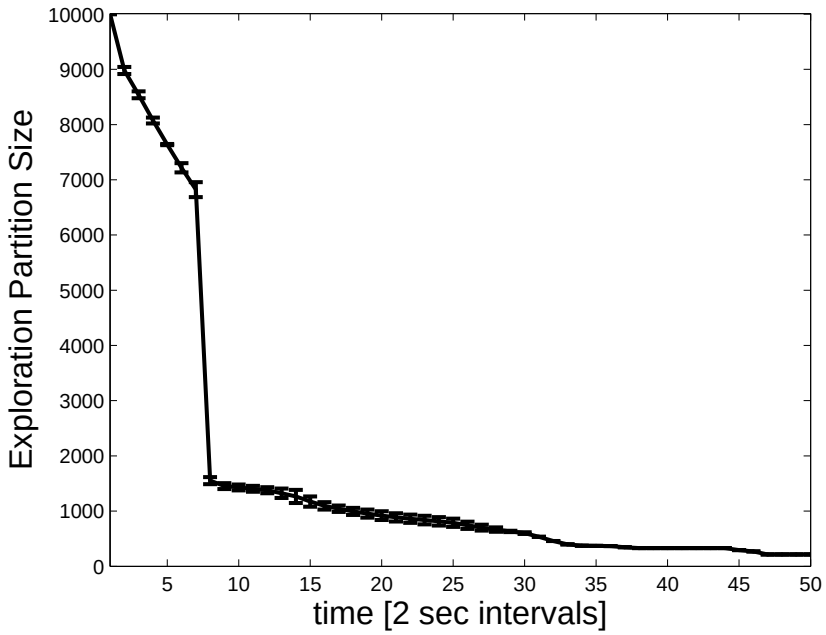


Figure 5.75: Scenario: zero targets and six agents. Approach: time-evolving partition classification. Sample mean exploration partition size over simulation time. Error bars represent sample standard deviation.

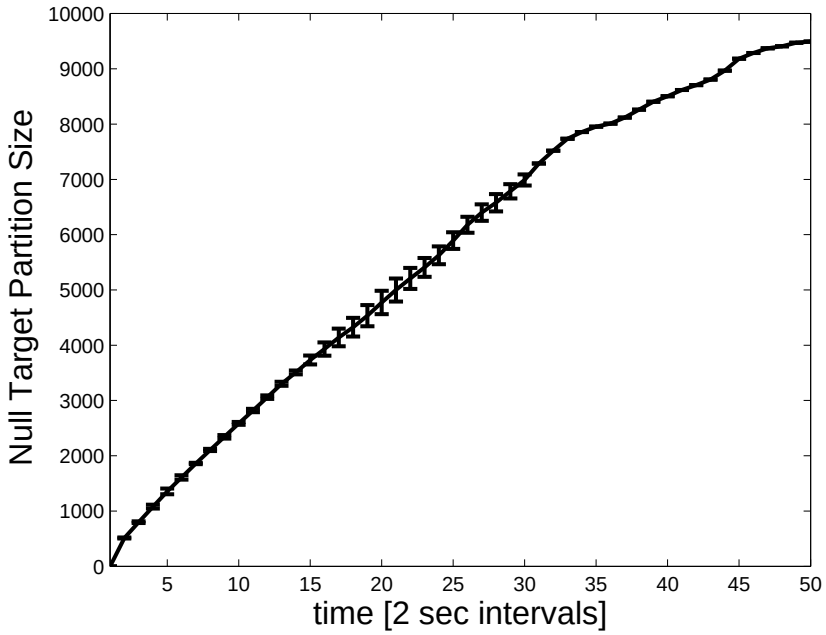


Figure 5.76: Scenario: zero targets and six agents. Approach: time-evolving partition classification. Sample mean null target partition size over simulation time. Error bars represent sample standard deviation.

5.7 Summary

In this chapter results were provided for the various elements of the time-evolving partition classification path planning. Before jumping straight into the partition classification path planning analysis, partition level path planning performance was analyzed. For partition level path planning, extensive tests were performed both in simulation, hardware-in-the-loop experiments, and in-flight experiments. From these results it was deemed that the established direct distribution path optimization algorithms performed well.

Then direct distribution path optimization was combined with time-evolving partition classification. To analyze the performance of time-evolving partition classification path planning, four scenarios were analyzed. These scenarios were

- The number of targets equals the number of agents.
- There are more targets than agents.
- There are less targets than agents.
- There are zero targets.

These scenarios were chosen to provide a representative sample of the possible scenarios that may be encountered in real area surveillance missions.

From these scenarios it was observed that time-evolving partition classification path planning performed best when the number of targets was equal to or less than the number of agents. The performance of time-evolving partition classification for the case when the number of targets is less than the number of agents was then compared to state-of-the-art path planning. The state-of-the-art path planning method chosen was target density distribution receding horizon sensor coverage maximization. The performance of the state of the art was then tested for the case when there are five targets and six agents. From these results it is clear that time-evolving partition classification path planning finds targets and covers more regions of the surveillance area in a shorter amount of time. This then leads to the conclusion that time-evolving partition classification path planning performs well when

- Target estimation is non-parametric.
- The sensor field of view is small relative to the surveillance area.
- The number of targets is unknown.
- The number of targets is less than or equal to the number of agents.

Note, however, that the partitioning still functions for the case when the number of targets is greater than the number of agents. It is just much more difficult to keep all targets localized because there is insufficient resource.

Chapter 6

Conclusions

6.1 Summary

The general problem that was addressed in this work was that of search and tracking an unknown number of targets utilizing a team of cooperative autonomous agents. This work then focused on this problem specific to the case when

- Points of sensor observation are kinematically or dynamically constrained to the previous points of sensor observation.
- The sensor field of view is small relative to the surveillance area.
- The method of target estimation is non-parametric to address nonlinear sensor observations made by visual spectrum cameras.

The state-of-the-art approach is to perform receding horizon optimization directly over a probability distribution representing probable target positions. However, for the case when the number of targets is unknown, the space of this probability distribution is unknown. The path planning problem is then very complicated. To account for the complications of the unknown number of targets, the approach taken in this work is to combine all target estimates into a target density distribution, which has a fixed dimension. The dimension of a target density distribution is the dimension of the surveillance area. Consequently, the dimension of the space over which path planning is performed goes from a complicated, unknown dimension to a fixed small dimension. The next step of the approach taken is to shift the planning from being target based to being region based. By doing this, the path planning problem can then focus on a known number of regions instead of an unknown number of targets. This is done by adding a layer of region learning before path planning. This region learning is called time-evolving partition classification. It is over these regions that path planning then operates.

6.2 Contributions

The contribution of this work is an approach toward solving the search and tracking path planning problem for the case when the number of targets is unknown. This approach has shown to find targets and search more diverse regions of the surveillance area in a shorter amount of time than state-of-the-art methods. The steps taken to accomplish this can be summarized as

1. Shift of search and tracking from being target based to being region based.
2. Decomposition of search and tracking into three components.
3. Learning over the target density distribution.
4. Time-evolving partition classification.
5. Path planning strategies over partitions.

The typical approach toward search and tracking is to generate estimates of each target detected and then to plan agent paths based on the target estimates. However, in reality information for some targets will be very uncertain, no information will exist for other targets, and the number of targets will not be known. The first contribution of this work is then to shift the focus of search and tracking away from the targets directly. The focus is then turned toward the types of regions that exist in the surveillance area. Search and tracking is then performed by considering the types of regions that exist and what type of path planning should be done for each region. Consequently, search and tracking is accomplished by defining regions of characteristic information and then searching over these various regions to gather information in ways tailored to the types of existing regions.

Recall that search and tracking is typically decomposed into two components. These components are 1) target track estimation and 2) agent path planning based on the target track estimation. Although the target track estimation is often further decomposed into detection and tracking components, and similarly path planning is often further decomposed, the estimation and path planning still stand as the only two components. This two level decomposition is typical because the focus of search and tracking is typically on the target estimates. Considering that in this work the focus is on the types of regions that exist in the surveillance area, the second contribution of this work is the decomposition of search and tracking into three components. Three components are needed to account for the step of defining regions within the surveillance area. This is done to better address the aspects of search and tracking for an unknown number of targets by a team of autonomous agents with sensors that have small fields of view relative to the surveillance area. The three components of this composition are

1. Target density estimation.

2. Surveillance area partition learning.

3. Path planning over surveillance area partitions.

The first component is to estimate the target density distribution. This distribution essentially combines an estimate of the exploration map with all target estimates to form an estimate of possible target density within regions of the surveillance area. The second component is surveillance area partition learning based on the target density distribution. This component is the step of search and tracking in which characteristic regions within the surveillance area are defined. This is the main focus of this work. The third component is agent path planning over the surveillance area partitions that were defined in the second component. Given that the surveillance area is partitioned, these partitions can be viewed as tasks to allocate to the team of agents. As such, in this work the path planning is further decomposed into 1) agent routing over partitions and 2) partition level path optimization based of partition level target density distributions.

As mentioned above, partition learning is the second component of search and tracking decomposition as presented in this work. The third contribution of this work is then methods for learning these partitions at each moment in time. This learning is referred to as time-evolving partition classification because the type of learning employed in this work is that of an unsupervised cascade of classifiers. This contribution is the main innovation presented in this work. All other contributions revolve around this.

The final theoretical contribution of this work is methods for planning agent paths over the time-evolving partitions. As mentioned above, path planning is decomposed into 1) agent routing over partitions and 2) partition level path optimization based on partition level target density distributions. In this work it is assumed that the agent routing step of path planning is given. Indeed, much development has been done to establish agent routing algorithms. Consequently, all focus of path planning was placed on partition level path optimization. Methods for accomplishing this level of path planning have been presented in this work.

Finally, in addition to the contribution specified above, another contribution is the implementation and performance analysis of the methods presented in this work. The results of the performance analysis suggest that the methods developed in this work perform well to enable search and tracking for an unknown number of targets by a team of autonomous agents with sensors that have small fields of view relative to the surveillance area.

6.3 Future Work

There are many possibilities for further work to build on the methods presented in this work for search and tracking. Some possibilities can be summarized as

- Implementation of direct target density distribution estimation.

- Particle based partition classification.
- Task costs tailored to the attributes of surveillance area partitions.
- Additional partition level path optimization methods.
- Distribution of partition classification over the team of agents.

In this work, the implementation utilized to analyze the performance of time-evolving partition classification and partition level agent path optimization methods did not directly estimate the target density distribution. Instead, an exploration map was maintained, as well as probability distributions for each detected target. A target density distribution was then formed by summing together the detected target distributions with the exploration map. Furthermore, the resulting target density distribution was grid based. Consequently, an immediate possibility for further work is the development of methods to directly estimate the target density distribution. There have been developed a few methods to estimate what is known as a Probability Hypothesis Density (PHD) filter, which turns out to be an approximation of the target density distribution [70]. Methods developed so far to estimate a PHD consist of either a Gaussian mixture or particle representation. But these methods were all developed to approximate Mahler's method of multiple target tracking [70]. Alternatively, other methods could be developed to directly estimate the target density distribution.

Considering that particle based approximations of the target density distribution have been developed (e.g., particle based PHD filters), the time-evolving partition classification algorithms should be extended to work with particle based estimates of the target density distribution. Particle based representations are typically used to quicken the speed of estimation. So extending the partition classification algorithms to work with particle based approximations of the target density distribution would enable compatibility with these types of distribution representations.

Another area that requires further work is the development of task costs tailored specifically to the attributes of surveillance area partitions. For focus of this work was mainly on developing the time-evolving partition classification. Focus was also placed on development of partition level agent path optimization over partition level target density distributions. As such, development of these task cost is still needed. These task costs should be made specific to the type of agent routing algorithm to be used.

Methods for partition level agent path optimization were developed and presented in this work. These methods were developed based on the approaches of information based distribution optimization, reinforcement learning, and receding horizon control. There are likely other methods that can be developed based on these fields. However, there may be yet other path optimization approaches.

A final possibility for further work that will be discussed here is the distribution and parallelization of the partition classification over the team of agents. In this work it was assumed that all distribution of computation would occur within the component of target

density distribution estimation, since distribution estimation parallelization is a topic that has received much attention and development. However, it may be possible to also distribute the computation of the partition classification over the team of agents.

Bibliography

[1] Mehdi Alighanbari and Jonathan P. How. A robust approach to the UAV task assignment problem. *International Journal of Robust and Nonlinear Control*, 18(2):118–134, 2008.

[2] Brian D. O. Anderson. *Linear Optimal Control*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[3] Brian D. O. Anderson. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[4] A. Antoniades, H. J. Kim, and S. Sastry. Pursuit-evasion strategies for teams of multiple agents with incomplete information. In *42nd IEEE Conference on Decision and Control, 2003*, volume 1, pages 756–761. IEEE, December 2003.

[5] R. T. Antony. *Principles of Data Fusion Automation*. Artech House, Norwood, MA, 1995.

[6] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

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

[8] Y. Bar-Shalom. *Tracking and data association*. Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[9] Yaakov Bar-Shalom and X. R. Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS, Urbana, IL, 1995.

[10] Yaakov Bar-Shalom and Xiao-Rong Li. *Estimation and Tracking : Principles, Techniques, and Software*. Artech House, Boston, 1993.

[11] Yaakov Bar-Shalom, Xiao-Rong Li, and Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley, New York, 2001.

[12] B. Basso, B. Kehoe, and J. K. Hedrick. A multi-level modularized system architecture for mobile robotics. In *Proceedings Dynamic Systems and Control Conference*. ASME, 2010.

[13] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 2010.

[14] Dimitri P. Bertsekas. *Introduction to Probability Theory*. Athena Scientific, 2002.

[15] R. E. Bethel and G. J. Paras. A PDF multisensor multitarget tracker. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):153 – 168, January 1998.

[16] S. S. Blackman. *Multiple Target Tracking with Radar Applications*. Artech House, Norwood, MA, 1986.

[17] S. S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, Inc., Norwood, MA, USA, 1999.

[18] F. Bourgault, T. Furukawa, and H.F. Durrant-Whyte. Coordinated decentralized search for a lost target in a bayesian world. In *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 48 – 53 vol.1, October 2003.

[19] F. Bourgault, T. Furukawa, and H.F. Durrant-Whyte. Decentralized bayesian negotiation for cooperative search. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2681 – 2686 vol.3, 2004.

[20] F. Bourgault, T. Furukawa, and H.F. Durrant-Whyte. Process model, constraints, and the coordinated search strategy. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 5256 – 5261 Vol.5, april-1 may 2004.

[21] Frederic Bourgault, Tomonari Furukawa, and Hugh Durrant-Whyte. Optimal search for a lost target in a bayesian world. In Shinichi Yuta, Hajima Asama, Erwin Prassler, Takashi Tsubouchi, and Sebastian Thrun, editors, *Field and Service Robotics*, volume 24 of *Springer Tracts in Advanced Robotics*, pages 209–222. Springer Berlin / Heidelberg, 2006. 10.1007/10991459_21.

[22] Sergiy Butenko, Robert Murphey, and P. M. Pardalos. *Recent Developments in Cooperative Control and Optimization (Cooperative Systems, "3)*. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[23] Zhiqiang Cao, Min Tan, Lei Li, Nong Gu, and Shuo Wang. Cooperative hunting by distributed mobile robots based on local interaction. *IEEE Transactions on Robotics*, 22(2):402–406, April 2006.

[24] Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, Aug 1995.

[25] P. R Chandler, M. Pachter, and S. Rasmussen. UAV cooperative control. In *American Control Conference, 2001. Proceedings of the 2001*, volume 1, pages 50 –55 vol.1, 2001.

[26] Hongda Chen, KuoChu Chang, and Craig Agate. Tracking with UAV using tangen-plus-lyapunov vector field guidance. In *12th International Conference on Information Fusion*, Seattle, WA, USA, July 2009.

[27] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 2001.

[28] I. Csiszar. Information-type measures of difference of probability distributions and indirect observations. In *Sudia Sci. Math. Hungar.*, volume 2, pages 299–318, 1967.

[29] D. J. Daley and David Vere-Jones. *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Springer, New York, 2007.

[30] G. B Dantzig and J. H Ramser. The truck dispatching problem. *Management Science*, 6(1):pp. 80–91, 1959.

[31] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

[32] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10:197–208, 2000. 10.1023/A:1008935410038.

[33] Arnaud Doucet and Adam M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *Handbook of Nonlinear Filtering*. Cambridge University Press, Cambridge, 2009.

[34] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[35] David A. Forsyth and Jean Ponce. *Computer Vision: A Moder Approach*. Pearson Education, Upper Saddle River, NJ, 2003.

[36] C. Fraser, L. F. Bertuccelli, and J. P. How. Reaching consensus with imprecise probabilities over a network. In *AIAA Guidance, Navigation, and Control Conference*, 2009.

[37] T. Furukawa, F. Bourgault, B. Lavis, and H.F. Durrant-Whyte. Recursive bayesian search-and-tracking using coordinated uavs for lost targets. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2521 –2526, May 2006.

[38] Jared Garvey, Benjamin Kehoe, Brandon Basso, Mark Godwin, Jared Wood, Joshua Love, Shih-Yuan Liu, Zuwhan Kim, Stephen Jackson, Yaser Fallah, Tony Fu, Raja Sengupta, and J. Karl Hedrick. An autonomous unmanned aerial vehicle system for sensing and tracking. In *Infotech@Aerospace Conference*, 2011.

[39] Brian P. Gerkey, Sebastian Thrun, and Geoff Gordon. Visibility-based pursuit-evasion with limited field of view. *The International Journal of Robotics Research*, 25(4):299–315, April 2006.

[40] Mark Godwin and J. Karl Hedrick. Stochastic approximation of an online vehicle routing problem for autonomous aircraft. In *Infotech@Aerospace Conference*, 2011.

[41] Mark Godwin, S. Spry, and J. Karl Hedrick. Distributed collaboration with limited communication using mission state estimates. In *American Control Conference*, 2006.

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

[43] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte. Information-theoretic coordinated control of multiple sensor platforms. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 1521 – 1526 vol.1, 2003.

[44] D. L. Hall. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Norwood, MA, 1992.

[45] J. P. Hespanha, Hyoun Jin Kim, and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *Proceedings of the 38th IEEE Conference on Decision and Control, 1999*, volume 3, pages 2432–2437, Phoenix, AZ, USA.

[46] D. Hladek, J. Vascak, and P. Sincak. Hierarchical fuzzy inference system for robotic pursuit evasion task. In *6th International Symposium on Applied Machine Intelligence and Informatics, 2008*, pages 273–277, Herlany, January 2008. IEEE.

[47] Y. C. Ho and R. C. K. Lee. *IEEE Transactions on Automatic Control*, AC-9:333 – 339, 1964.

[48] G.M. Hoffmann and C.J. Tomlin. Mobile sensor network control using mutual information methods and particle filters. *Automatic Control, IEEE Transactions on*, 55(1):32 –47, 2010.

[49] G. Hollinger, A. Kehagias, and S. Singh. Probabilistic strategies for pursuit in cluttered environments with multiple robots. In *2007 IEEE International Conference on Robotics and Automation*, pages 3870–3876, Roma, April 2007. IEEE.

[50] V. Isler, C. Belta, K. Daniilidis, and G. J. Pappas. Hybrid control for visibility-based pursuit-evasion games. In *Proceedings 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1432–1437. IEEE, October 2004.

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

[52] Yan Jin, Yan Liao, A. A. Minai, and M. M. Polycarpou. Balancing search and target response in cooperative unmanned aerial vehicle (UAV) teams. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(3):571–587, June 2005.

[53] E. W. Kamen and C. R. Sastry. Multiple target tracking using products of position measurements. *IEEE Transactions on Aerospace and Electronics Systems*, 29(2):476 – 493, April 1993.

[54] K. Kastella. Event-averaged maximum likelihood estimation and mean-field theory in multitarget tracking. *IEEE Transactions on Automatic Control*, AC-40(6):1070 – 1074, 1995.

[55] K. Kastella. Discrimination gain for sensor management in multitarget detection and tracking. In *IEEE-SMC and IMACS Multiconference Transactions on Automatic Control*, pages 1 – 6, July 1996.

[56] K. Kastella. Joint multitarget probabilities for detection and tracking. *Proceedings of SPIE, Acquisition, Tracking and Pointing XI*, 3086:122 – 128, April 1997.

[57] Robert W. Keener. *Statistical Theory: A Medley of Core Topics*. 2005.

[58] Z. Kim and R. Sengupta. Target detection and position likelihood using an aerial image sensor. In *Proceedings International Conference on Robotics and Automation*, 2008.

[59] Chris Kreucher, Keith Kastella, and Alfred O. Hero III. Information-based sensor management for multitarget tracking. *SPIE Signal and Data Processing of Small Targets*, 5204:480–489, 2003.

[60] C.M. Kreucher, A.O. Hero, K.D. Kastella, and M.R. Morelande. An information-based approach to sensor management in large dynamic networks. *Proceedings of the IEEE*, 95(5):978 –999, May 2007.

[61] S. Kullback and R. Leibler. On information and sufficiency. *Annals Mathematical Statistics*, 22:79–86, 1951.

[62] A. D. Lanterman, M. I. Miller, D. L. Snyder, and W. J. Miceli. Jump diffusion processes for the automated understanding of FLIR scenes. *SPIE Proceedings*, 2234:416 – 427, 1994.

[63] B. Lavis, T. Furukawa, and H.F. Durrant-Whyte. Dynamic space reconfiguration for bayesian search and tracking with moving targets. *Autonomous Robotics*, 2008.

[64] Benjamin Lavis and Tomonari Furukawa. HyPE: hybrid particle-element approach for recursive bayesian searching-and-tracking. In *Robotics: Science and Systems IV*. MIT Press, 2009.

[65] E. L. Lawler. *The Traveling Salesman Problem: a Guided Tour of Combinatorial Optimization*. Wiley, Chichester, New York, 1985, 1985.

[66] Jusuk Lee, Rosemary Huang, Andrew Vaughn, Xiao Xiao, J. Karl Hedrick, Marco Zennaro, and Raja Sengupta. Strategies of path-planning for a UAV to track a ground vehicle. In *Second Annual Symposium on Autonomous Intelligent Networks and Systems*, 2003.

[67] E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer-Verlag, New York, 1998.

[68] D. V. Lindley. On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[69] R. Mahler. Unified sensor management using CPHD filters. In *Information Fusion, 2007 10th International Conference on*, pages 1 –7, 2007.

[70] Ronald P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., Norwood, MA, USA, 2007.

[71] G. Mathew, A. Surana, and I. Mezic. Uniform coverage control of mobile sensor networks for dynamic target detection. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7292 –7299, dec. 2010.

[72] George Mathew and Igor Mezi. Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Physica D: Nonlinear Phenomena*, 240(4-5):432 – 442, 2011.

[73] George M. Mathews, Hugh Durrant-Whyte, and Mikhail Prokopenko. Decentralised decision making in heterogeneous teams using anonymous optimisation. *Robotics and Autonomous Systems*, 57(3):310 – 320, 2009. Selected papers from 2006 IEEE International Conference on Multisensor Fusion and Integration (MFI 2006), 2006 IEEE International Conference on Multisensor Fusion and Integration.

[74] T.G. McGee and J.K. Hedrick. Guaranteed strategies to search for mobile evaders in the plane. In *American Control Conference, 2006*, page 6 pp., june 2006.

[75] Ilya Molchanov. *Theory of Random Sets*. Springer, 2011 edition, 2005.

[76] Hung T. Nguyen. *Introduction to Random Sets*. Chapman and Hall/CRC, Boca Raton, FL, 2006.

[77] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. On divergences, surrogate loss functions, and decentralized detection. Technical Report 695, Department of Statistics, University of California, Berkeley, October 2005.

[78] A. Renyi. On measures of entropy and information. In *Proc. 4th Berkeley Symp. on Mathematical Statistics and Probability*, volume 1, pages 547–561, 1961.

[79] Christian P. Robert. *The Bayesian Choice: from Decision-Theoretic Foundations to Computational Implementation*. Springer, New York, 2007.

[80] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, NJ, USA, 2nd edition, 2003.

[81] A. Ryan, H.F. Durrant-Whyte, and J. K. Hedrick. Information-theoretic sensor motion control for distributed estimation. In *Proceedings International Mechanical Engineering Congress and Exposition*, 2007.

[82] R. W. Schaefer and A. V. Oppenheim. *Discrete-time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[83] Raja Sengupta, John Connors, Ben Kehoe, Zu Kim, Tom Kuhn, and Jared Wood. Autonomous search and rescue with ScanEagle. Technical report, prepared for Evergreen Unmanned Systems and Shell International Exploration and Production Inc., September 2010.

[84] Madhavan Shanmugavel, Antonios Tsourdos, Brian White, and Rafal Zbikowski. Cooperative path planning of multiple UAVs using dubins paths with clothoid arcs. *Control Engineering Practice*, 2009.

[85] C. Shannon. A mathematical theory of communication. *Bell Systems Thechnical Journal*, 27:379–423 and 623–656, 1948.

[86] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice-Hall, Upper Saddle River, NJ, 2001.

[87] A. Sinha, T. Kirubarajan, and Y. Bar-Shalom. Autonomous ground target tracking by multiple cooperative UAVs. In *Aerospace Conference, 2005 IEEE*, pages 1 –9, march 2005.

[88] Per Skoglar. *Planning Methods for Aerial Exploration and Ground Target Tracking*. Licentiate thesis, Linkoping University, Linkoping, 2009.

[89] H. W. Sorensen. Recursive estimation for nonlinear dynamic systems. In J. C. Spall, editor, *Bayesian Analysis of Time Series and Dynamic Models*. Marcel Decker, New York, 1988.

[90] S. C. Spry, A. R. Girard, and J. K. Hedrick. Convoy protection using multiple unmanned aerial vehicles: Organization and coordination. In *2005 American Control Conference*, volume 5, pages 3524–3529. IEEE, June 2005.

[91] L. Stone. *Theory of Optimal Search*. Academic Press, New York, NY, 1975.

[92] Lawrence D. Stone, Thomas L. Corwin, and Carl A. Barlow. *Bayesian Multiple Target Tracking*. Artech House, Inc., Norwood, MA, USA, 1st edition, 1999.

[93] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*. Richar S. Sutton and Andrew G. Barto, 1998.

[94] J. Tisdale, Zuwhan Kim, and J. Hedrick. Autonomous UAV path planning and estimation. *Robotics Automation Magazine, IEEE*, 16(2):35 –42, 2009.

[95] J. Tisdale, A. Ryan, Z. Kim, D. Tornqvist, and J. K. Hedrick. A multiple UAV system for vision-based search and localization. In *Proceedings American Controls Conference*, 2008.

[96] John Tisdale, Hugh Durrant-Whyte, and J. Karl Hedrick. Path planning for cooperative sensing using unmanned vehicles. In *Proceedings of IMECE 2007*, Seattle, USA, Nov 2007.

[97] P. Toth and D. Vigo. The Vehicle Routing Problem. *SIAM Monographs on Discrete Mathematics and Applications*, 9, 2002.

[98] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 2008.

[99] A. Wald. *Sequential Analysis*. Wiley, New York, 1947.

[100] E. Waltz and J. Llinas. *Multisensor Data Fusion*. Artech House, Boston, MA, 1990.

[101] R. B. Washburn. A random point process approach to multiobject tracking. In *Proceedings of the American Control Conference*, volume 3, pages 1846 – 1852, June 10 - 12 1987.

[102] El-Mane Wong, F. Bourgault, and T. Furukawa. Multi-vehicle bayesian search for multiple lost targets. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3169 – 3174, 2005.

[103] Jared Wood and J. Karl Hedrick. Multi-agent path planning for an unknown number of targets over dynamic space partitions. In *50th IEEE Conference on Decision and Control and European Control Conference*, December 2011.

[104] Jared Wood and J. Karl Hedrick. Space partitioning and classification for multi-target search and tracking by heterogeneous unmanned aerial system teams. In *Infotech@Aerospace 2011*. AIAA, 2011.

[105] Jared Wood, Benjamin Kehoe, and J. Karl Hedrick. Target estimate PDF-based optimal path planning algorithm with application to UAV systems. In *Proceedings Dynamic Systems and Control Conference*. ASME, 2010.