Worcester Polytechnic Institute
Digital WPI

Doctoral Dissertations (All Dissertations, All Years) | Electronic Theses and Dissertations

2018-11-30

Interactive Planning and Sensing for Aircraft in Uncertain Environments with Spatiotemporally Evolving Threats

Benjamin S. Cooper
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-dissertations

Repository Citation
Cooper, B. S. (2018). *Interactive Planning and Sensing for Aircraft in Uncertain Environments with Spatiotemporally Evolving Threats.* Retrieved from https://digitalcommons.wpi.edu/etd-dissertations/513

This dissertation is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Doctoral Dissertations (All Dissertations, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.

INTERACTIVE PLANNING AND SENSING FOR AIRCRAFT IN UNCERTAIN ENVIRONMENTS WITH SPATIOTEMPORALLY EVOLVING THREATS

by

Benjamin Cooper

A dissertation submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN AEROSPACE ENGINEERING.

December 2018

APPROVED:

Dr. Raghvendra V. Cowlagi, Advisor
Assistant Professor, Aerospace Engineering Program.

Dr. Michael A. Demetriou, Committee Member
Professor, Aerospace Engineering Program, WPI.

Dr. Nikolaos A. Gatsonis, Committee Member
Professor, Aerospace Engineering Program, WPI.

Dr. Nikhil Karanjgaonkar, Graduate Committee Representative
Assistant Professor, Aerospace Engineering Program, WPI.

Dr. Puneet Singla, Committee Member
Associate Professor, Department of Aerospace Engineering,
The Pennsylvania State University.

Acknowledgements

Contained in this dissertation is an exploration of the relationship between planning and estimating. While the discussion and results within are technical and mathematical in nature, years of studying these topics have led me to a deeper philosophical appreciation of the interaction between planning and estimation. The solution I present takes an iterative form in which plans are made given our best understanding (estimation) of the world. Then, additional data is gathered to verify this plan, which if found deficient, must be revised.

It is easy to find parallels in your own life. If you are a PhD student, perhaps you planned to solve a rather impressive problem for your degree, that is, until you began working on that problem in depth and had to revise your ambitious plan. Now consider there is an element of time permeating your problem. The difficulty of planning and gathering relevant information dramatically increases. Your hope of graduating in five years has suffered from your initial ambitious and uncertain plan. Wouldn't it be nice if there were an algorithm to guide you in achieving your goal, given all these complications and constraints? I wasn't nearly that ambitious to propose such an algorithm for my PhD, but I did benefit greatly from the guidance and support of a series of mentors, family, colleagues, and friends. Their guidance and support optimized my path and reduced the uncertainty in my goals, and I must thank them immensely.

I thank my advisor, Dr. Raghvendra Cowlagi, who took the biggest risk by taking in this former structural dynamics experimentalist to delve into theoretical aspects of planning and estimation. I can't imagine why he thought that was a good idea at the time, perhaps he was desperate. Either way, I am extremely grateful for his bold investment. Raghu has transformed a once blunt tool relying on intuition and hard work into a slightly more refined device capable of classifying and dissecting complex problems. I look forward to any future collaborations we may have.

I would like to thank my committee members. Dr. Nikhil Karanjgaokar, for serving as the committee representative. Dr. Nikolaos Gatsonis for his perspective on the fluid behaviour implications of one of my examples. Thank you to Dr. Puneet Singla who served as an external member

and suffered through the technical limitations of remote collaboration. Finally, thank you to Dr. Michael Demetriou, from whom I took many courses and for whom I presented many seminar talks. I am sure I haven't heard the last of him.

I am grateful for the financial support of the Mechanical Engineering department through Teaching Assistant positions, and later through the United States Air Force Office of Scientific Research (AFOSR grant # FA9550-17-1-0028). This support made this research possible.

Thank you to the great many friends, coaches, and mentors I have had throughout Massachusetts and New Mexico. I have absorbed many great lessons about leadership, sacrifice, compassion, failure, and success from these individuals. Though I could list the names of these important people given enough time, there isn't space to sufficiently detail their contributions. So I will leave it at that. I give a final thanks to my lab mates, Jighjigh Ivase, Zetian Zhang, Jie Fang, Chase St. Laurent, who have helped me grow technically and emotionally, and Ruixiang Du who encouraged my appreciation of proper programming practices.

It took me a while to recognize the influence my parents had on my success. I don't remember needing much direct help on homework, or pestering to complete projects. Their lessons were more abstract and learned through simple exposure. My father worked incredibly hard both in his career and in providing opportunities to play and grow. My mother endured reading the same bedtime stories over and over, answering all my "but why?" questions, and demonstrated many other examples of calm patience. A PhD favors not the brilliant but the persistent and patient. From those two qualities, work ethic and patience, I have derived all my successes.

I thank my one and only sibling, who is far braver than I could ever be. Again, the lesson comes by way of example. We live fairly distinct and different lives, yet I have tried to become more self-aware of the person I am, and the risks I might take to find happiness and success.

At last, thank you to Jodie who reminds me that success and happiness do not always require working too hard, sometimes you only need to work one hard (possibly my favorite joke). It is from her that I learned my final lesson: balance. A solution is not really good if it requires maximum effort for success. Such a solution is not robust to disturbances. I can finally appreciate that a well balanced approach yields the optimal life in the presence of life's uncertainties.

Abstract

Autonomous aerial, terrestrial, and marine vehicles provide a platform for several applications including cargo transport, information gathering, surveillance, reconnaissance, and search-and-rescue. To enable such applications, two main technical problems are commonly addressed. On the one hand, the *motion-planning problem* addresses optimal motion to a destination: an application example is the delivery of a package in the shortest time with least fuel. Solutions to this problem often assume that all relevant information about the environment is available, possibly with some uncertainty. On the other hand, the *information gathering problem* addresses the maximization of some metric of information about the environment: application examples include such as surveillance and environmental monitoring.

Solutions to the motion-planning problem in vehicular autonomy assume that information about the environment is available from three sources: (1) the vehicle's own onboard sensors, (2) stationary sensor installations (e.g. ground radar stations), and (3) other information gathering vehicles, i.e., *mobile sensors*, especially with the recent emphasis on collaborative teams of autonomous vehicles with heterogeneous capabilities. Each source typically processes the raw sensor data via estimation algorithms. These estimates are then available to a decision making system such as a motion-planning algorithm. The motion-planner may use some or all of the estimates provided. There is an underlying assumption of "separation" between the motion-planning algorithm and the information about environment. This separation is common in linear feedback control systems, where estimation algorithms are designed independent of control laws, and control laws are designed with the assumption that the estimated state is the true state.

In the case of motion-planning, there is no reason to believe that such a separation between the motion-planning algorithm and the sources of estimated environment information will lead to optimal motion plans, even if the motion planner and the estimators are themselves optimal. The goal of this dissertation is to investigate whether the removal of this separation, via *interactive* motion-planning and sensing, can significantly improve the optimality of motion-planning.

The major contribution of this work is interactive planning and sensing. We consider the problem of planning the path of a vehicle, which we refer to as the *actor*, to traverse a threat field with minimum threat exposure. The threat field is an unknown, time-variant, and strictly positive scalar field defined on a compact 2D spatial domain – the actor's *workspace*. The threat field is estimated by a network of mobile sensors that can measure the threat field pointwise. All measurements are noisy. The objective is to determine a path for the actor to reach a desired goal with minimum *risk*, which is a measure sensitive not only to the threat exposure itself, but also to the uncertainty therein. A novelty of this problem setup is that the actor can communicate with the sensor network and request that the sensors position themselves in a procedure we call *sensor reconfiguration* such that the actor's risk is minimized.

This work continues with a foundation in motion planning in time-varying fields where waiting is a control input. Waiting is examined in the context of finding an optimal path with considerations for the cost of exposure to a threat field, the cost of movement, and the cost of waiting. For example, an application where waiting may be beneficial in motion-planning is the delivery of a package where adverse weather may pose a risk to the safety of a UAV and its cargo. In such scenarios, an optimal plan may include "waiting until the storm passes." Results on computational efficiency and optimality of considering waiting in path-planning algorithms are presented. In addition, the relationship of waiting in a time-varying field represented with varying levels of resolution, or multiresolution is studied.

Interactive planning and sensing is further developed for the case of time-varying environments. This proposed extension allows for the evaluation of different mission windows, finite sensor network reconfiguration durations, finite planning durations, and varying number of available sensors. Finally, the proposed method considers the effect of waiting in the path planner under the interactive planning and sensing for time-varying fields framework.

Future work considers various extensions of the proposed interactive planning and sensing framework including: generalizing the environment using Gaussian processes, sensor reconfiguration costs, multiresolution implementations, nonlinear parameters, decentralized sensor networks and an application to aerial payload delivery by parafoil.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Literature Review

1.1 Motivation and Problem Statement

Autonomy has been driven by the technological leaps of digital computing. Ever smaller and cheaper computers have spurred the proliferation of the embedded system such that autonomous systems are found in factories, processing plants, cars, planes, coffee machines, and more. Control and autonomy is the hidden technology that gains public view when it intersects science fiction such as in driverless cars, UAV's, and robots. Another technology that rests just below the public's attention is the network. Many will be familiar with the Internet and telephones as elements of a network, but their design and operation remain in the background. The continual development of autonomy and networks gives rise to the *control of networks* and the *networks of control*. Control of networks considers the topology, interactions, and demands of elements and connections in a network, such as coordinating telephone calls and webpage requests. Networks of control is the paradigm of interest in this thesis and we explore a particular formulation of *distributed autonomy*, where a heterogenous team of mobile vehicles collaboratively executes a common task. Mirroring the evolution of computers from single to multi-core processors, networking has brought an age of distributed control.

There are two major technical components of vehicular autonomy: (1) sensing and situational awareness in the vehicle's environment, and (2) motion-planning and control to achieve autonomous motion in this environment. These two components can be identified with the observer/estimator and the controller subsystems, respectively, in a traditional control systems framework. Mirroring a common practice in control systems design, a "principle of separation" between these two components is often assumed, as evident from discussions in textbooks on autonomous mobile vehicles, e.g., (Siegwart et al., 2011). This separation entails that the collection and pro-

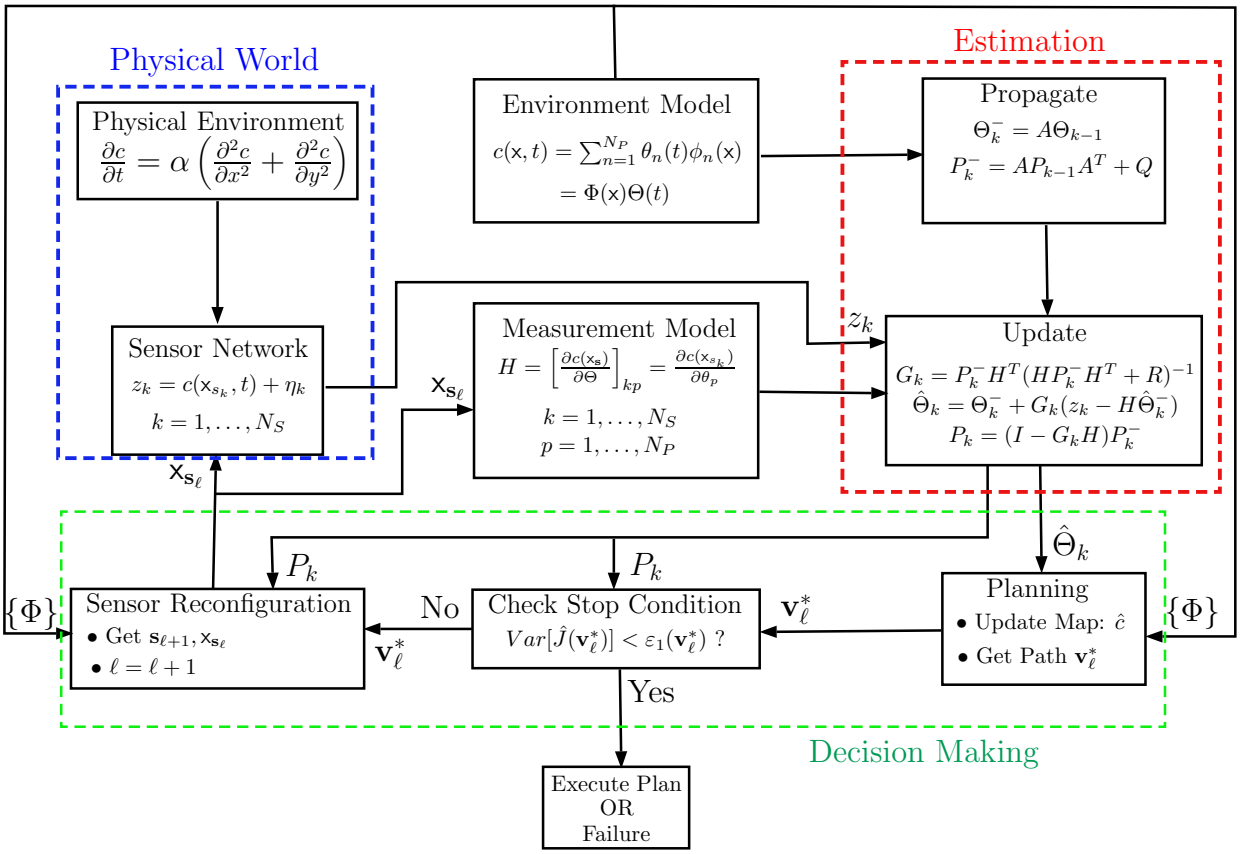


Figure 1.1: Information flow diagram demonstrating the connections between the physical world, modeled worlds, estimation, and decision making elements.

cessing of sensor data is independent of the specific motion-planning problem at hand. In this dissertation, we study a specific motion-planning problem and a sensor setup, *Interactive Planning and Sensing* (IPAS), where the removal of such separation between the two components of autonomy can be beneficial. The diagram in Fig. 1.1 provides an overview of the main ideas explored in this dissertation and emphasizes the coupled and interconnected nature of a decision making and estimation problem.

Motion-planning is an important component of autonomy. There are a large variety of applications of unmanned aerial and terrestrial vehicles to drive the need for motion-planning. Motion-planning for an unmanned aerial or terrestrial vehicle (UxV) is the process of finding control inputs that enable the vehicle to travel from a prespecified initial position to a prespecified destination. The vehicle's motion is described by a controlled dynamical system. The environment in which the vehicle moves may contain obstacles that the vehicle must avoid. Finally, the vehicle trajectory

may be associated with a quality metric, i.e. a *cost* function, and the objective of motion-planning is to find an obstacle-free vehicle trajectory of minimum cost. The solution of the motion-planning often involves discretization of the environment to first determine an obstacle-free *path*, which is a finite sequence of "waypoints" that approximately specifies the vehicle's actual trajectory.

This dissertation concerns the following concepts: path-planning, spatiotemporally varying fields, and sensor limits. These are all major elements that collectively inform but do not define the primary theme of this thesis: autonomy. Although autonomy is a broad term that alludes to decision making and self-sufficiency, we specifically study point-to-point path-planning to achieve a minimum cost traversal in spatiotemporally-varying environments, given limitations on available information. Two limitations on sensor capabilities will be investigated: sensor *noise* and sensor *resolution*.

We investigate the general problem of path-planning in a 2D workspace in the presence of a spatiotemporal threat field, where minimal exposure to the threat is desired. We include the possibility of waiting for finite intervals of time. We consider first a scenario where the threat field is fully known, and then a scenario where the field is partially known in vehicle-centric multiresolution detail. This multiresolution representation can portray sensor resolution limits and serve as a method to reduce computational burden as will be discussed later. The solutions to these problems are conceptually simple: the dimension of the search space is expanded to include the temporal dimension. However, computational implementations are challenging because the search space along this temporal dimension is in principle unbounded even if the 2D spatial workspace is compact.

One action not often explicitly considered in path planning in time-varying fields is the decision to not move, or to *wait* in one form or another. We envision scenarios where commanding the vehicle to wait in the form of parking, loitering, or circling provide the optimal solution. When the environment is a time-invariant field and when the quality metric penalizes path traversal time, waiting in not beneficial. Waiting implies the trade-off of some unit of time in exchange for a reduced cost elsewhere (e.g. reduced danger, reduced travel cost, reduced uncertainty in field/state) which minimizes the overall cost. Therefore, waiting needs to be considered only when the environment is time-varying.

For a concrete example, consider the problem of planning a driving route from one place to another. Minimum-time solutions to this problem are provided by many commercially available gadgets (CNET Editorial Board, 2017) and freely available software applications such as Google Maps (Google Inc., 2018). Furthermore, such applications are now capable of using real-time traffic data to provide estimates of the travel duration (Lowe, 2017). Table 1.1 shows estimated travel durations between the cities Springfield, MA, Boston, MA, and an intermediate town Millbury, MA, which is approximately equidistant from Springfield and Boston[1]. These data are obtained using the Google Maps application on a desktop computer browser, queried at 06:15 AM and again at 07:07 AM on April 10, 2018. Notice the following peculiarity of this result. Per the 06:15 AM query, the estimated travel duration from Springfield to Millbury, MA is 52 min, and that from Millbury to Boston is 1 hr. 11 min. A vehicle starting from Springfield at 06:15 AM will ostensibly reach Millbury at 07:07 AM. However, in the 07:07 AM query, the estimated travel duration from Millbury to Boston increases by 18%, and the total travel duration from Springfield to Boston will be larger than the 06:15 AM estimate of 2 hrs. 1 min. An explanation of this discrepancy is that the Google Maps application possibly[2] ignores temporal variations in traffic, which are indicated in Figs. 1.2(a) and 1.2(b).

Table 1.1: Estimated travel times obtained from the Google Maps application. Data obtained on April 10, 2018.

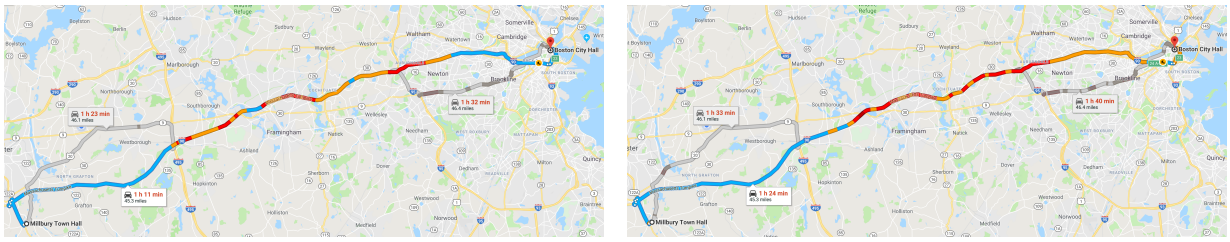
	Spr. to Bos.	Spr. to Mil.	Mil. to Bos.
Query at 06:15 AM	2 hrs. 3 min.	52 min	1 hr. 11 min.
Query at 07:07 AM	2 hrs. 16 min.*	–	1 hr. 24 min.

*: This duration is calculated by adding the Spr. $\rightarrow$ Mil. duration to the Mil.$\rightarrow$ Bos. duration estimated in the 07:07 AM query.

Consider now that the objective of the same path planning problem were not to minimize travel duration, but rather to minimize a weighted sum of travel duration and exposure to traffic. Such an objective may be of importance to reduce the health risks to long-haul truck drivers by reducing their exposure to automobile emissions (Peters et al., 2004; Raaschou-Nielsen et al., 2011; Rice et al., 2015). This problem is also of renewed importance because in recent times, a wealth of traffic data has become available to make accurate predictions of traffic (Lv et al., 2015; Tan et al.,

[1]The city halls of each city are chosen as specific starting and ending street addresses.

[2]The algorithmic details of this application are proprietary and not publicly known.



(a) (b)

Figure 1.2: Traffic density variations over timescales comparable to the duration of travel. Images acquired using Google Maps (Google Inc., 2018). Areas with higher traffic density are indicated with yellow and red colors. (a) Mil. to Bos.; query at 06:15 AM. (b) Mil. to Bos.; query at 07:07 AM.

2016). In this case, an optimal route may involve *waiting* (e.g. at a rest area) for traffic to subside.

This problem is an example of the general problem of path-planning problem with minimum exposure to a scalar field (e.g. traffic density in this example) that varies with space and time. Other applications of this problem include motion-planning for aerial vehicles in inclement weather with physics-based predictive models, and for mobile robots with continually updated estimates of workspace features from noisy measurements.

The above problems of time-varying environments and interactive planning and sensing in uncertain environment maps can be developed independently. However, in a time-varying and uncertain map where the external mobile sensors collect measurements to estimate the field, an important research question is: **"Should the primary vehicle wait while the mobile sensor network collects measurements, and if so, for how long should it wait?"** In other words, consider the waiting time as a function of information gathered or cost of path currently evaluated. The solution to this question is the natural extension, *Interactive Planning and Sensing for Time-varying fields* (IPAST), which dynamically reallocates mobile sensors in time-varying environment to discover and reduce the uncertainty of the minimum-cost path.

1.2 Background and Literature Review

Path- and motion-planning for autonomous mobile vehicles is a well-studied research area (Choset et al., 2005). The canonical problem formulations are of finding obstacle-free paths and/or

motions (i.e., paths in the vehicle state space) between prespecified initial and destination points in a compact 2D or 3D workspace. A large variety of methods including geometric methods (Kambhampati and Davis, 1986; Latombe, 1991), randomized sampling-based methods (Frazzoli et al., 2002; Karaman and Frazzoli, 2011; LaValle and Kuffner, 2001), trajectory optimization methods (Betts, 1998; Garg et al., 2010), and combinations thereof (Cowlagi and Tsiotras, 2012a; Plaku et al., 2010) are reported in the literature to solve these canonical problems. Planning in partially known workspaces (Stentz, 1994, 1995), or in workspaces with dynamic obstacles (Xu et al., 2010) is also well-studied. However, problems involving predictive models of changes in the workspace, where the planner can actively choose to wait for a more "favorable" workspace, are less thoroughly studied. Examples of applications of such problems include motion-planning for aerial vehicles in inclement weather with physics-based predictive models, motion-planning with continually updated estimates of workspace features from noisy measurements, and path-planning for cars (with considerations of waiting at rest areas) with known traffic flow patterns.

1.2.1 Modeling Environments with Basis Functions

Before describing the literature directly relevant to the extensions on path-planning presented in this work, it is valuable discuss the environment used in the following investigations. The environment, often referred to herein as the *threat field*, is a linear combination of basis functions defined on a *workspace*, $\mathcal{W} \subset \mathbb{R}^2$, which is a closed square region. Each individual basis function is considered a *threat* component, and a threat field is made of N_{P} threats or basis functions distributed within a predefined workspace $\mathcal{W}$. The basis functions and the number of parameters N_{P} are prespecified. Specifically, we assume a Gaussian basis function

$$\phi_n(\mathsf{x}) := \exp(-\frac{1}{2\nu_n} \cdot (\mathsf{x} - \bar{\mathsf{x}}_n)^{\mathrm{T}}(\mathsf{x} - \bar{\mathsf{x}}_n)), \tag{1.1}$$

for each $n \in [N_{\mathrm{P}}]$, where $\nu_n \in \mathbb{R}_{>0}$ and $\bar{\mathsf{x}}_n \in \mathcal{W}$ are prespecified constants.

The threat field is finitely and linearly parametrized as

$$c(\mathsf{x}) = B + \sum_{n=1}^{N_{\mathrm{P}}} \theta_n \phi_n(\mathsf{x})$$

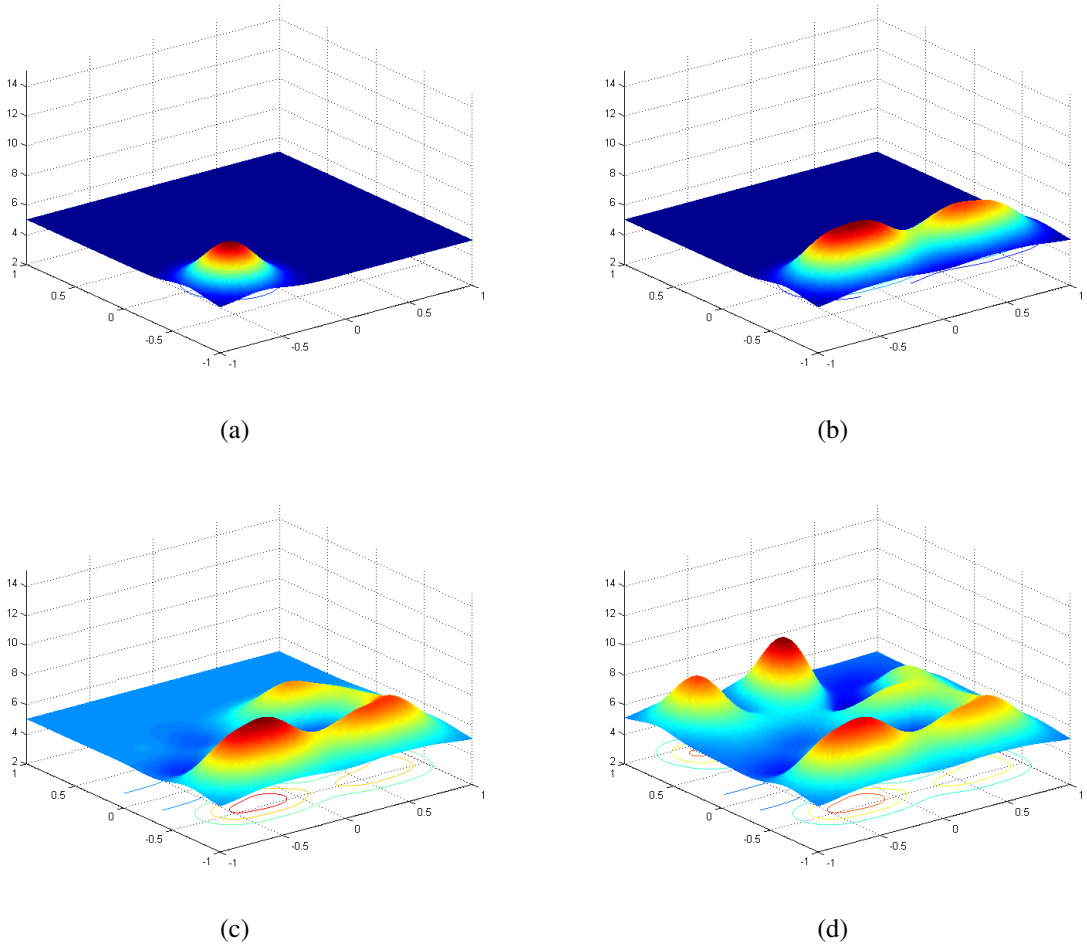
$$= B + \Phi^{\mathrm{T}}(\mathsf{x})\Theta \tag{1.2}$$

where B is a bias that offsets the field, and θ_n are coefficients associated with each basis ϕ_n and represent the intensity of that threat.

The finite parameterization of the threat field c using Gaussian functions is justified by the fact that a large class of functions on $\mathbb{R}$, namely, square integrable functions, can be approximated with arbitrary precision by linear combinations of Gaussian functions (Calcaterra, 2008). Furthermore, 2D Gaussian functions of varying width can be used to represent varying levels of resolution by including narrower Gaussian functions into higher resolution images or video (Goshtasby and Oneill, 1994). Finally, Gaussian function appear in series solutions to several partial differential equations such as the diffusion equation (Crank, 1979), which enables the application of the proposed work to path- and motion-planning with threat field modeled by physical phenomena such as advection-diffusion of gases or radiation in the atmosphere (Demetriou et al., 2013). In the context of this work, Gaussian functions also provide the smoothness in the derivatives, which we assume later in Chapter 4 when addressing multiresolution representations of the threat field.

Figure 1.3 demonstrates the process of generating a generic threat field distributed across the workspace. Individual basis may partially overlap with neighboring basis to create various different environment topologies. In the next section, we demonstrate how a path-planning problem is formulated and solved for this example static environment.

Remark (Discontinuous environments) The environment models presented in this dissertation are primarily smooth fields due to the use of the Gaussian function as a basis. However, the primary contribution, Interactive Planning and Sensing, does not explicitly rely on the use of smooth fields as the environment model. It would be possible in include environments with discontinuous features such as jumps in the field, or piecewise representations of the field. For the case of Gaussian functions, this could be accomplished by properly sizing and locating additional basis functions to approximate a jump or box function. This approach is similar to the superposition of sin and cosine waves in Fourier series approximations. Additionally, other basis functions can be chosen such as Super Gaussian functions which can be designed to have very sharp drops while still remaining smooth. Finally, various wavelet functions may be used for discontinuous



(a) (b)

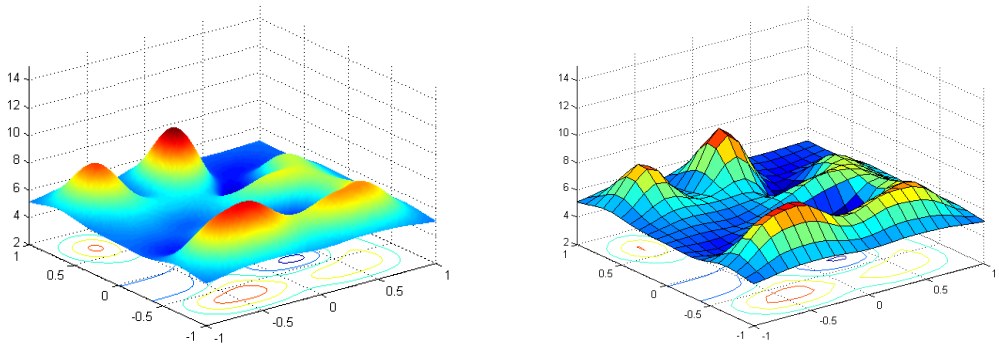
(c) (d)

Figure 1.3: Generating a threat field for the component basis functions.

environments. In fact, we make use of wavelet functions to represent the multi-resolution path planning problem in Chapter 4. One of the extensions of interactive planning and sensing discussed in the future work section is the incorporation of the multi-resolution representation of the environment to explore its effect on the formulation and operation of the interactive planning and sensing framework.

1.2.2 Path-planning in Static Environments

Workspace cell decomposition is widely used in path-planning (Brooks and Lozano-Pérez, 1985; Latombe, 1991). The vehicle's workspace (i.e., the planar region in which the vehicle moves) is partitioned into convex regions called *cells*, that are either free or full of obstacles. A graph is



(a) (b)

Figure 1.4: Discretizing the threat field.

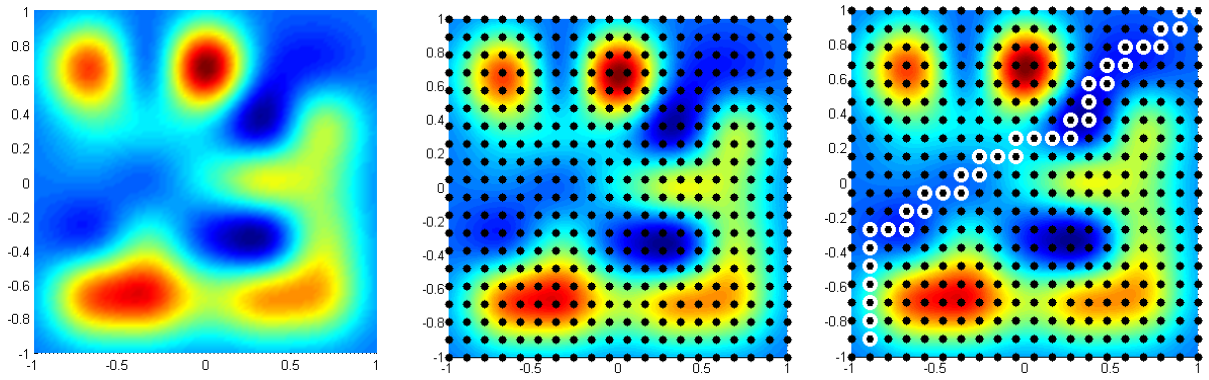
defined such that each vertex of $\mathcal{G}$ uniquely corresponds to a free cell, and edges in this graph are defined according to geometric adjacency of cells. The vehicle's path-planning is then transformed to a problem of searching for a minimum cost path in this graph, for which techniques such as Dijkstra's algorithm and the A* algorithm (Russell and Norvig, 2003) are available.

Using the threat field generated in Fig. 1.3, the continuous field can be uniformly discretized to produce a discrete representation of the field, as in Fig. 1.4. Then each cell of the discretization can be assigned as grid point as in Fig. 1.5(b). As described, each grid point is associated with a vertex on a graph, where edge connects adjacent grid points. The traversal of an edge is assigned a cost based on the local threat intensity (e.g. the value of the threat field at the destination grid point). Finally, a graph search based algorithm such as Dijkstra is used to determine the minimum cost path, as in Fig. 1.5(c).

The rest of the literature review and thesis describe the problems that arise and solutions needed when the threat field is time-varying, when the discretization is non-uniform, and/or when the intensity of the threat field is unknown and must be measured with some level of uncertainty.

1.2.3 Path-planning in Time-Varying Evironments

In the Dijkstra's algorithm path-planning technique, it is straightforward to include traversal costs based on a *time-invariant* field defined over the workspace. Optimal path-finding algorithms



(a) (b) (c)

Figure 1.5: The smooth threat field (a) is discretized, a graph is associated with each grid point (b), and a minimal cost path is obtained using Dijkstra's algorithm (c) or other grid-based planners.

for graphs with time-varying edge costs appear in (Akgüna et al., 2007; Cai et al., 1997; Orda and Rom, 1991; Philpott and Mees, 1993), but their applications to path-planning in a time-varying spatial field have not appeared in the literature.

Time-dependent path planning has its' most direct applications in travel and transportation. Much of the work in this area is motivated by transportation route-planning (Chen and Yang, 2004; Pallottino, 1998; Sung et al., 2000) and travel planning between cities (Berube et al., 2006). The typical objective in transportation problems is minimum time or minimum delay paths (a.B. Philpott and a.I. Mees, 1992; Orda and Rom, 1990). In such applications, the cost of a path is the total time taken to travel from a source to a destination, and often evaluating the all sources to a single destination (Chabini, 2013). A scenario in which **waiting** can be beneficial is given in (Orda and Rom, 1990) where a passenger contemplates taking the local slower train stopping in front of him or waiting for the express train to take him to the final destination. Such scenarios consider time-varying edge costs where the edge cost has a different delay or equivalent travel speed depending on the current time. This time-dependent edge delay cost may result in behavior violating the first-in-first-out (FIFO) condition of the network and has consequences which will be discussed and addressed later. When the FIFO condition holds, waiting never benefits the minimum time path. However, there are other objectives in path planning besides minimal time, which are usually collected under the term *minimum cost* and may included fuel consumption, minimal exposure to some threat, or some preferred travel terrain. Minimum cost paths with waiting are

covered extensively in (Dean, 2004a; Orda and Rom, 1991) and include proper assumptions on FIFO conditions. The other major example of waiting in time-varying networks involves data transmission (a.B. Philpott and a.I. Mees, 1992; Cai et al., 1997; Dell'Amico et al., 2008; Gangulay et al., 2004), however waiting in time-varying spatial fields has also not been addressed.

1.2.4 Multiresolution Path Planning

Path-planning using multiresolution cell decompositions involves representing the vehicle's environment with different levels of accuracy. For example, the quadtree method (Kambhampati and Davis, 1986; Noborio et al., 1990; Samet, 1984), generates a planar cell decomposition consisting of small cell sizes that accurately capture obstacle boundaries, and larger cell sizes that efficiently represent large areas in free space. Other examples of multiresolution path-planning techniques include (Behnke, 2004; Hwang et al., 2003; Kim and Lee, 2005; Prazenica et al., 2005; Verwer, 1990). Though many of these hierarchical representations are present in path-planning, the incorporation of waiting into this method is yet unseen.

Computationally efficient multiresolution cell decompositions, and associated path-planning techniques based on the discrete wavelet transform (DWT) have been discussed in (Cowlagi and Tsiotras, 2012b; Jung, 2007; Tsiotras and Bakolas, 2007). The wavelet transform is a powerful tool widely used in multiresolution signal processing (Daubechies, 1994; Mallat, 1989; Rao and Bopardikar, 1998), and in vision-based navigation, localization, and mapping (Cho and Nam, 2000; Ilkyun et al., 2009; Lui and Jarvis, 2010; Shim et al., 1999). Autonomous vehicles use data collected from multiple onboard sensors as well as ground-based and satellite-based sensors. To assimilate data from such a variety of sensors, the wavelet transform is expected become the common standard of the representation and analysis of signals (Cipra, 1994; Paulson et al., 2010; Wei and Fwa, 2004; Wiesemann et al., 2002; Yguel et al., 2006). Other examples of the application of wavelet transforms in multiresolution path-planning include (Carrioli, 1991; Narayanaswami and Pang, 2003; Pai and Reissell, 1998; Sinopoli et al., 2001).

1.2.5 Motion-planning with Uncertainty

Another important research area in motion-planning is the development and evaluation of planners that account for uncertainty in one or more elements of the problem formulation. Applications including information-driven path planning, environment mapping, SLAM, and motion-planning under uncertainty can be categorized by three main sources (Kurniawati et al., 2012): (1) motion or action uncertainty due to modeling errors, (2) vehicle state uncertainty due to state measurement noise, and (3) environment map uncertainty due to measurement noise and limits on resolution.

The first two uncertainty sources, action and state, are typically addressed under Stochastic Motion Planning techniques. These problems are formulated as a Markov Decision Process (MDP) or a partially observable MDP (POMDP), and can be solved using dynamic programming (DP) (Bertsekas, 2000). Unfortunately, a naïve implementation of DP is, in general, computationally intractable for practical applications. Another tool is the sampling-based methods, such as Probabilistic Roadmaps (PRM's) and Rapidly-exploring Random Trees (RRT's) (Alterovitz et al., 2007; Missiuro and Roy, 2006). Following on this, belief space roadmaps are discussed for motion-planning under uncertainty (Alterovitz et al., 2007; Prentice and Roy, 2009). The literature on simultaneous localization and mapping (SLAM) algorithms addresses (Chakravorty and Saha, 2008; Lerner et al., 2007) the simultaneous reduction in uncertainty in the environment map and in the vehicle's state, by estimating parameters that describe environmental features. SLAM methods may also involve planning, typically formulated as an MDP (Chakravorty and Saha, 2008) that reflects the uncertainties in the map and in the vehicle state.

1.2.6 Sensor Management and Optimal Placement

distributed estimation of processes/fields, Information-driven path planning The literature on **sensor management** addresses optimal placement of sensors to estimate distributed processes (Krause et al., 2008; Ucinski, 2010), including the atmospheric dispersal of gases (Demetriou and Ucinski, 2011), the spread of volcanic ash (Madankan et al., 2014), and the identification and control of structural vibrations (Padula and Kincaid, 1999). Guidance and coordination strategies for mobile

sensors are also discussed (Demetriou et al., 2013; Martinez, 2010; Martinez and Bullo, 2006) for envisioned implementations using teams of unmanned aerial, terrestrial, or underwater vehicles (UXVs).

A typical *performance metric* used to characterize optimal sensor placement is maximum information, or equivalently, minimum entropy (Cochran and Hero, 2013). Maximization of a characteristic, such as the determinant, the trace, or the largest eigenvalue, of the Fisher Information Matrix (FIM) is another frequently used performance metric. Other metrics include the classical least squared error in parameter estimates, mutual information, Battacharya coefficients, Hellinger distance, and Kullback-Lieber divergence (Cochran and Hero, 2013; Krause et al., 2008; Mu et al., 2015).

Target tracking and localization has benefited from using information-driven approaches to sensor measurements. It could be considered as a subset of the parameter estimation problem and, as stated, subject to information driven methods as in (Adurthi and Singla, 2014; Farmani et al., 2014). When framed as a problem for UAV's, two more concerns arise, coordination/planning and vehicle performance of the UAV's as mobile sensors whether they are fixed wing aircraft suited to long-term monitoring or quadrotors more suited for short-term agile pursuit. In (Martinez and Bullo, 2006), Martinez and Bullo address the question of motion coordination for target tracking and express that both the motion control algorithm and estimation process should be optimally integrated to make the most of network performance.

The idea of an integrated and coordinated planning and sensing scheme is broaching closer to the heart of this paper. Singla and Adurthi tackle the problem of optimally routing a UAV to better estimate the targets state while using as little energy as possible (Adurthi and Singla, 2014). In this problem, they make use of a Mutual Information construction using an information distance metric based on Hellinger distance in terms of Battacharya coefficients. The balance between information gathering, vehicle performance, and task fulfillment (target tracking) leads to a paper by Kreucher (Kreucher et al., 2005) in which the merits of Task Driven versus Information Driven sensor management were compared for the case of target tracking. Within the frame of the paper, they concluded that for a specific task (eg. target tracking), a task-specific objective function (eg. minimize the tracking error) works best. However, when there are multiple competing performance

criteria then information-driven approaches perform consistently as well. Kreucher's paper gives justification for the use of information driven approaches, but leaves the particular implementation for further pursuit and exploration.

1.2.7 Estimation and Sensor Placement for Time-varying systems

Sensor selection and scheduling for Kalman filtering is of direct relevance to this work. This continually evolving topic is driven in large part by work on sub- and supermodular functions whose properties allow greedy algorithms to select sets of sensors which are nearly optimal. Since this property was originally analyzed (Nemhauser et al., 1978), greedy algorithms invoking submodular functions have been explored extensively (Krause et al., 2008; Ranieri et al., 2014). The sensor placement problems fall under finite horizon solutions (Gupta et al., 2006; Mo et al., 2011; Tzoumas et al., 2016) and infinite horizon solutions (Asghar et al., 2017; Mo et al., 2014; Ye et al., 2018; Zhang et al., 2017). Infinite horizon problems often focus on minimizing the statistical steady state error covariance (Zhang et al., 2017) or finding periodic sensor schedules (Mo et al., 2014; Shi and Chen, 2013). Furthermore, the *sensor selection* (or design-time) problem maximizes performance for a single set of fixed sensors (Tzoumas et al., 2016; Zhang et al., 2017), whereas the *sensor scheduling* (or run-time) problem considers possibly different sensor configurations at each time step (Asghar et al., 2017; Jawaid and Smith, 2015; Mo et al., 2011). In this work, we study path-planning in the scenario of sensor scheduling under a finite time horizon.

1.2.8 Interactive Planning and Sensing

We are concerned with environment uncertainty insofar as it affects the optimal planning for a point to point path. A similar concern is presented in (Al-Sabban et al., 2013) where the goal is point to point planning in uncertain, time-varying wind fields with the extra objective of leveraging wind energy to reduce overall energy costs. The extra objective in our work is to leverage external mobile sensors to reduce the uncertainty in the field in the relevant domain of interest, which is the environment around the optimal path. Another effort in the spirit of our goal is the work by Skoglar (Skoglar et al., 2006) which seeks an information driven planning approach. The vehicle

has an on-board gimbaled electro-optical/infrared camera which can be controlled. The goal is to concurrently plan the vehicle path and the 'path' of the sensor or rather the direction of the gaze to maximize the information necessary to plan the route, but also the route plan maximizes the ability to gain information. Their work touches on ideas of optimal design of experiments and active sensing.

The recent works (Tzoumas et al., 2016, 2018) consider the sensor selection problem for LQG control. Initially noting the problem requires jointly designing sensing, estimation, and control and thus breaking the traditional principle of separation, the authors present an argument which allows design of sensor selection for estimation separate from the control problem. However, they consider the *sensor selection* case in which a fixed set of sensors is chosen for measurements at all time steps. In comparison, we seek a solution to the planning and sensing problem in which a subset of a network of sensors can assume different configurations for any given time step during the finite time horizon. To this end, the proposed work studies a method to determine a domain of the actor's interest for guiding sensor placement: specifically, the iterative process that results from the computation of this domain, the subsequent sensor placement, and replanning by the actor with newly acquired sensor data.

1.3 Thesis Overview and Statement of Contributions

The goal of this dissertation is to develop planning algorithms to find optimal routes in spatiotemporally varying fields subject to map uncertainty or varied resolutions.

1.3.1 Overview

We observe the following missing links in the existing literature: while many problems address the optimal use of sensor networks to estimate a distributed environment, to the best of the author's knowledge, there is no work focused on using a sensor network to directly optimize the objective of a path planning problem, i.e. minimizingthe sum of edge costs in point-to-point motion-planning implemented on a graph; applications of planning may consider time-varying costs, but few con-

sider the direct implication of *waiting* on algorithm complexity and optimality. In this thesis, we first develop an interactive planning and sensing (IPAS) algorithm which iteratively plans a path and redirects a sensor network to identify the optimal path with a required level of certainty, and extend the concept to spatiotemporally-varying environments. Second, we address the explicit case of *waiting* as part of the solution to a path-planning problem in a spatiotemporally-varying environment. We investigate the computational complexity as well as optimality through implementation of a multiresolution decomposition of the time-varying environment, with comparisons to standard informed search heuristics.

One of the themes explored in this thesis is the relationship between the *level* of information available and the *quality* of the path given that information. When referring to the *level* of information, we may refer to the available resolution of the environment as explored in the multiresolution representation of the environment, or the *uncertainty* of the environment as formulated in the interactive planning and sensing work in Chapters 2 and 5. These two formulations arise naturally from the common limits on sensor measurements, namely measurement resolution and measurement noise.

Therefore, this thesis explores the practical but fundamental question of optimizing the performance of path-planning algorithms given the very real limits of resolution and noise. This work not only considers that the information available is limited in some way, but makes the claim that it is not necessary or even desirable to maximize the level of information when the goal is finding optimal paths.

1.3.2 Contributions

We make the following contributions toward path-planning in spatially- and spatiotemporally-varying environments in which information may be limited by resolution or uncertainty.

Interactive Planning and Sensing Optimizes Path Performance For the case of environments formulated by a combination of linear basis functions, the interactive planning and sensing algorithm using a sensor network optimizes the path performance to within an arbitrarily small value

of the true path cost. We present theorems and proofs demonstrating both the termination and convergence of this algorithm in a finite number of iterations.

Interactive Planning and Sensing Minimizes Sensor Network Resources with respect to Information-Driven Sensor Placement Methods In addition to optimizing path performance, the IPAS algorithm uses a minimal amount of sensor resources to accomplish the required path cost tolerance. We demonstrate that for a given amount of sensor measurements, the IPAS algorithm achieves better path performance with higher confidence than several other information-based sensor placement strategies. This advantage increases both as the number of parameters in the environment increases and the number of available sensors decreases.

Pruning of the Search Graph Preserves Optimality of Waiting Path Planners By imposing a *local wait check* on the neighbors of a vertex, we can preserve the most beneficial routes that include waiting while reducing the computational burden of exhaustively searching the entire search graph space.

Waiting has Negligible Impact on Computational Expense in Multiresolution Path-planning When using a multiresolution representation of the environment, the explicit inclusion of *waiting* in the search algorithm has negligible effect on computational expense. Whenever the search algorithm explores waiting, the *spatial* multiresolution decomposition does not change. Therefore, in the multiresolution path-planning problem, the allowance of waiting does not incur significant additional computational expense.

Interactive Planning and Sensing for Time-varying Environments The algorithm and principles behind the interactive planning and sensing algorithm are extended to environments with both spatially and temporally varying parameters. This is accomplished by reformulating the least squares parameter estimation as a Kalman filter in which the parameter system is the dynamic system being estimated. While the extension to time-varying fields is straightforward, the inclusion of the time dimension significantly complicates the convergence properties of the algorithm. We present a revised convergence criterion and discuss the influence of various problem elements

including sensor reconfiguration time, planning time, the rate of evolution of the threat field (as capture by a diffusivity parameter α), as well as the number of parameters of the field and number of sensors available.

Waiting as a Side Effect of Interactive Planning and Sensing for Time-varying Environments In the noiseless information case we discussed the benefits and issues involved with waiting in uniform high resolution and multiresolution representations of the threat field. As a consequence of the reconfiguration and planning required during iterations of the IPAST algorithm, we observe additional evidence that waiting should be considered when planning. This provides additional motivation that waiting should be incorporated explicitly in path-planning problems including the IPAST algorithm presented.

Chapter 2

Interactive Planning and Sensing for Path-planning in a Threat Field

2.1 Problem Elements Overview

In what follows, we denote by $\mathbb{R}$ and $\mathbb{N}$ the sets of real and natural numbers, respectively; by $[N]$ the set $\{1, \ldots, N\}$ for any $N \in \mathbb{N}$; and by $\mathbb{I}_{(N)}$ the identity matrix of size N. For the reader's convenience, a nomenclature table is provided.

Symbol	Meaning
$\bar{\mathsf{x}}_n, \nu_n$	Constants used in spatial basis functions (2.2)
$N_{\mathrm{S}}, N_{\mathrm{P}}$	Numbers of sensors, parameters, resp.
N_{G}	Number of grid points
x_i	Grid point location of vertex $i \in [N_{\mathrm{G}}]$
$c, \hat{c}$	True, estimated threat, resp.
$\hat{\Theta}, P$	Threat estimate and error covariance, resp.
$\mathcal{J}, \hat{J}$	True, estimated path cost, resp.
$\mathbf{v}^*$	True optimal path
$\mathbf{s}_\ell$	Sensor placement at iteration ℓ
$\bar{\ell}$	# iterations for IPAS algorithm termination
$\mathcal{I}$	Set of identified parameters, see (2.5)
$\mathcal{K}$	Minimal cover of path, see (2.10)
$\mathcal{L}$	Sensor reconfiguration, see (2.11)

Let $\mathcal{W} \subset \mathbb{R}^2$ be a closed square region, called the *workspace*, in which the actor and the sensors move. In this workspace, we formulate a grid consisting of N_{G} points uniformly placed the workspace. The coordinates in a prespecified Cartesian coordinate axis system of the i^{th} grid point are denoted by x_i, for each $i \in [N_{\mathrm{G}}]$. We consider a strictly positive scalar field $c : \mathcal{W} \to \mathbb{R}_{>0}$, called the *threat field*, which represents unfavorable regions with higher intensity. The actor is assumed to traverse grid points according to a "$4-$connectivity rule." Let δ denote the distance

between adjacent grid points. We neglect vehicle kinematic and dynamic constraints, while noting that such constraints can in the future be easily incorporated in the proposed grid-world problem setup (Cowlagi and Tsiotras, 2012a), and that multiresolution grids can also be considered (Cowlagi and Tsiotras, 2012b). We also assume that the actor vehicle has no uncertainties in localization or in motion on the grid: i.e., the current grid-point location of the actor is known, and the effect of moving to an adjacent grid-point is deterministic and known.

The actor's motion-planning problem is formulated as a graph search problem on a graph $\mathcal{G} = (V, E)$, where each vertex in V is uniquely associated with a grid point, and labeled by integers $1, 2, \ldots N_{\mathrm{G}}$. The edge set E is the set of pairs of vertices associated with adjacent grid points. Edge transition costs are assigned by a scalar function $g : E \to \mathbb{R}_{>0}$ defined as

$$g((i,j)) = c(\mathsf{x}_j), \text{ for } i, j \in [N_{\mathrm{G}}],\ (i,j) \in E. \tag{2.1}$$

A *path* in the graph $\mathcal{G}$ between two prespecified vertices i_{s} and i_{g} is a sequence $\mathbf{v} = (v_0, v_1, \ldots, v_P)$ of successively adjacent vertices with $v_0 = i_{\mathrm{s}}$ and $v_P = i_{\mathrm{g}}$. The *cost* $\mathcal{J}(\mathbf{v}) \in \mathbb{R}_{>0}$ of this path is the sum of edge transition costs, i.e., $\mathcal{J}(\mathbf{v}) := \sum_{k=1}^{P} g((v_{k-1}, v_k))$. The actor's motion-planning problem is the problem of finding a path with minimum cost between initial and goal grid points $i_{\mathrm{s}}, i_{\mathrm{g}} \in [N_{\mathrm{G}}]$. We will refer to this path as the *true optimal* path, denoted by $\mathbf{v}^*$.

2.2 Uncertain Environment Formulation

Uncertainty in the actor's motion-planning problem arises from uncertainty in the knowledge of the threat field. The threat field is finitely and linearly parametrized as $c(\mathsf{x}) = 1 + \sum_{n=1}^{N_{\mathrm{P}}} \theta_n \phi_n(\mathsf{x}) = 1 + \Phi^{\mathrm{T}}(\mathsf{x})\Theta$, where $\phi_n : \mathcal{W} \to \mathbb{R}_{>0}$ are spatial basis functions, $\Phi(\mathsf{x}) := [\ \phi_1(\mathsf{x}) \ \ \ldots \ \ \phi_{N_{\mathrm{P}}}(\mathsf{x})\ ]^{\mathrm{T}}$, and $\Theta := [\ \theta_1 \ \ \ldots \ \ \theta_{N_{\mathrm{P}}}\ ]^{\mathrm{T}}$. The basis functions and the number of parameters N_{P} are prespecified. Specifically, we assume

$$\phi_n(\mathsf{x}) := \exp(-\frac{1}{2\nu_n} \cdot (\mathsf{x} - \bar{\mathsf{x}}_n)^{\mathrm{T}}(\mathsf{x} - \bar{\mathsf{x}}_n)), \tag{2.2}$$

for each $n \in [N_{\mathrm{P}}]$, where $\nu_n \in \mathbb{R}_{>0}$ and $\bar{\mathsf{x}}_n \in \mathcal{W}$ are prespecified constants.

Definition 1 (Region of significant support)**.** The region $\mathcal{R}_n^{\sup} := \{\mathsf{x} : \|\mathsf{x} - \bar{\mathsf{x}}_n\| \leqslant 3\sqrt{\nu_n}\} \cap \mathcal{W}$ is defined as the *region of significant support* for the basis function ϕ_n.

Whereas the functions ϕ_n do not have compact support in $\mathbb{R}^2$, 99.74% of the volume enclosed under each ϕ_n is enclosed by the restriction of ϕ_n to $\mathcal{R}_n^{\sup}$.

Assumption 1. The constants ν_n and $\bar{\mathsf{x}}_n$ are chosen such that the union of the interiors of the regions of significant support cover the entire workspace.

When N_{P} is a perfect square, the workspace coverage in Assumption 1 is achieved by choosing $\bar{\mathsf{x}}_n$ on a uniform $\sqrt{N_{\mathrm{P}}} \times \sqrt{N_{\mathrm{P}}}$ grid on the workspace $\mathcal{W}$. To ensure coverage and to minimize overlap of the basis functions, ν_n is chosen such that the $\mathcal{R}_n^{\sup}$ extends to half the diagonal distance between basis functions. Specifically, for adjacent basis functions with indices n and p, we choose $\nu_n := (\frac{\sqrt{2}}{6}(\bar{\mathsf{x}}_n - \bar{\mathsf{x}}_p))^2$.

This choice of Gaussian basis functions is justified by the fact that square integrable functions can be approximated with arbitrary precision by linear combinations of Gaussian functions (Calcaterra and Boldt, 2008). The proposed algorithm does not specifically depend on this choice, and other basis functions such as orthogonal wavelets (Rao and Bopardikar, 1998) can be used.

A finite number of sensors $N_{\mathrm{S}} \ll N_{\mathrm{G}}$ are available to take pointwise measurements in the workspace. A number $N_{\mathrm{L}} \leqslant N_{\mathrm{S}}$ of these sensors are placed at distinct grid points. The set of these grid points, called a *placement,* is denoted by $\mathbf{s} = \{s_1, \ldots, s_{N_{\mathrm{L}}}\}$. The measurement taken by each sensor is $z_k := c(\mathsf{x}_{s_k}) + \eta_k$, where $\eta_k \sim \mathcal{N}(0, \sigma_k^2)$, for each $k \in [N_{\mathrm{L}}]$.

We define a *region of estimability* for each basis ϕ_n, which is a region in which a sensor must be located to confidently estimate the parameter θ_n. To this end, let $\bar{\sigma} := \max\{\sigma_k\}_{k=1}^{N_{\mathrm{S}}}$. For Gaussian measurement noise, if the signal-to-noise ratio (SNR) is greater than three, $\frac{\theta_n \phi_n}{\bar{\sigma}} \geqslant 3$, the probability of detecting a signal is at least 0.9974 (Miller et al., 2005). Therefore, placing sensors outside of the region where $\theta_n \phi_n \geqslant 3\bar{\sigma}$ is not advisable[1]. By (2.2), it follows that this region is defined by the ball $\left\{\mathsf{x} \in \mathcal{W} : \|\mathsf{x} - \bar{\mathsf{x}}_n\| \leqslant \sqrt{2\nu_n \log(\theta_n/3\bar{\sigma})}\right\}$.

Whereas the parameter θ_n is unknown, the radius of this ball scales logarithmically[2] with θ_n.

[1]Note that the maximum SNR is obtained when the sensor is placed at $\mathsf{x} = \bar{\mathsf{x}}_n$.

[2]To be precise, the radius scales slower: i.e., with the square root of the logarithm of θ_n.

Therefore, this region can be determined using a rough order of magnitude estimate of θ_n, which is typically available in practice. We assume that upper and lower bounds $\bar{\theta}_n$ and $\underline{\theta}_n$ are known, and that these bounds are within two orders of magnitude of each other.

It is always possible to place a sensor at a grid point inside $\mathcal{R}^{\text{est}}$ if $\delta < 2\sqrt{2\nu_n \log(\bar{\theta}_n/3\bar{\sigma})}$. We tighten this region further to ensure that the regions of estimability of adjacent basis functions do not significantly overlap. By doing so, estimates of parameters made from measurements taken within this region are independent from estimates of other parameters. That is, if z_n, z_p are measurements taken by sensors placed in $\mathcal{R}_n^{\text{est}}$ and $\mathcal{R}_p^{\text{est}}$, respectively, with $n \neq p$ and $n, p \in [N_{\text{P}}]$, then $\text{Covar}(\hat{\theta}_n, \hat{\theta}_p | z_n, z_p) \approx 0$. It is always possible to place a sensor at a grid point inside $\mathcal{R}_n^{\text{est}}$ if $\delta < 2(1 - \frac{1}{\sqrt{2}})\Delta\bar{\mathsf{x}}$, where $\Delta\bar{\mathsf{x}} := \min\{\|\bar{\mathsf{x}}_n - \bar{\mathsf{x}}_p\| : n, p \in [N_{\text{P}}]\}$.

Definition 2 (Region of estimability)**.** The *region of estimability* for the basis function ϕ_n is the set

$$\mathcal{R}_n^{\text{est}} := \left\{ \mathsf{x} : \|\mathsf{x} - \bar{\mathsf{x}}_n\| \leqslant (1 - 1/\sqrt{2})\Delta\bar{\mathsf{x}} \right\}.$$

2.2.1 Path-planning in Uncertain Environment Problem Statement

The actor can avail of estimates of the threat field parameters generated using measurements taken by the sensors over a sequence of placements $\{\mathbf{s}_\ell\}_{\ell \in \mathbb{N}}$. Therefore, the actor's path-planning problem is reformulated by considering deterministic but unknown edge transition costs based on the *estimated* threat field.

Problem 1. *Find a sequence of sensor placements $\{\mathbf{s}_\ell\}_{\ell \in \mathbb{N}}$, and a path $\hat{\mathbf{v}}^*$ in $\mathcal{G}$ with minimum estimated cost.*

A precise definition of estimated cost is provided in Section 2.3. Problem 1 encodes an explicit dependence between the sensor locations and the actor's path-planning problem. It is of course desirable that the path with minimum *estimated* cost is close to or identical to the *true* optimal path as defined by (2.1), i.e., and $|\mathcal{J}(\hat{\mathbf{v}}^*) - \mathcal{J}(\mathbf{v}^*)|$ is small. A significant finding of this paper is that if Problem 1 is solved by decoupling the sensor placement and path optimization subproblems, then the *true* cost of the resulting path can be significantly suboptimal (see Section 6.1.1). More importantly, the proposed approach finds solutions to Problem 1 such that $\hat{\mathbf{v}}$ is near-optimal.

2.3 Interactive Planning and Sensing Algorithm

For a given set $\mathbf{s}$ of sensor locations, let $\hat{\Theta}$ and P denote the mean and estimation error covariance matrix of the least squares estimate of the parameter Θ. Precisely:

$$\hat{\Theta} := H^{\mathrm{L}}\mathbf{z}, \qquad P := (H^{\mathrm{T}}R^{-1}H)^{-1} \tag{2.3}$$

Here $R := \mathrm{diag}(\sigma_1^2, ..., \sigma_{N_{\mathrm{L}}}^2)$ is the measurement error covariance matrix, $\mathbf{z} = [z_1 \dots z_{N_{\mathrm{L}}}]^{\mathrm{T}}$,

$$H := \begin{bmatrix} \Phi(\mathsf{x}_{s_{l,1}}) & \Phi(\mathsf{x}_{s_{l,2}}) & \dots & \Phi(\mathsf{x}_{s_{l,N_{\mathrm{L}}}}) \end{bmatrix}^{\mathrm{T}},$$

and H^{L} is the left pseudo-inverse. Using these estimated parameters, a threat field estimate is constructed as $\hat{c}(\mathsf{x}) := 1 + \Phi^{\mathrm{T}}(\mathsf{x})\hat{\Theta}$, and an estimated edge transition cost function $\hat{g} : E \to \mathbb{R}_{>0}$ is defined similar to (2.1).

This least squares estimation problem is well-posed when $N_{\mathrm{L}} \geqslant N_{\mathrm{P}}$, and the resulting estimates will be unbiased with minimum variance. Two complications can arise. First, the problem can become *ill-conditioned* when sensors are placed outside the regions of significant support of the spatial basis functions ϕ_n. In the proposed problem formulation, these basis functions are prespecified and their regions of support are known. Therefore, ill-conditioning is easily avoided.

Second, the problem becomes *ill-posed* when $N_{\mathrm{L}} < N_{\mathrm{P}}$. In this case, the parameter estimation is an underdetermined linear system, and has infinitely many solutions. The proposed approach addresses ill-posedness by identifying a smaller set of basis functions of interest, thereby restricting the estimation problem to a smaller number of parameters.

The estimated cost of a path $\mathbf{v} = (v_0, \dots, v_P)$ in $\mathcal{G}$ is $\hat{J}(\mathbf{v}) = \sum_{k=1}^{P} \hat{g}((v_{k-1}, v_k)) = P + \sum_{k=1}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_k)\hat{\Theta}$. For a given parameter estimate $\hat{\Theta}$, the actor's problem of finding an optimal path in $\mathcal{G}$ can be solved using a standard path optimization algorithm such as Dijkstra's algorithm (Bertsekas, 2000).

The novelty of the proposed work is that Problem 1 is solved by the following iterative method. At each iteration $\ell \in \mathbb{N}$, this method finds a parameter estimate $\hat{\Theta}_\ell$, the estimation error covariance

matrix P_ℓ, a sensor placement $\mathbf{s}_\ell$, and a path $\mathbf{v}_\ell^*$ in $\mathcal{G}$ with minimum estimated cost, such that with a high probability

$$\lim_{\ell\to\infty} \hat{J}(\mathbf{v}_\ell^*) = \lim_{\ell\to\infty} \mathcal{J}(\mathbf{v}_\ell^*) \leqslant \mathcal{J}(\mathbf{v}^*) + \varepsilon, \tag{2.4}$$

where ε is a suboptimality bound precisely identified in Theorem 1 below. The process of finding the next sensor placement $\mathbf{s}_{\ell+1}$ based on the actor's current optimal path $\mathbf{v}_\ell^*$ is called *sensor reconfiguration.*

The algorithm terminates when the variance of the estimated path cost variance, $\mathrm{Var}[\hat{J}(\mathbf{v}^*)]$, reaches a value smaller than a prespecified threshold. The details of this termination condition, including the variance computation, are discussed in Section 2.6.

This interactive planning and sensing (IPAS) algorithm is described in Lines 1–9 of Fig. 2.1 and the sensor reconfiguration policy is described in Lines 1–3.

The IPAS algorithm is initialized in Line 1 with an arbitrary sensor placement denoted $\mathbf{s}_0$. The algorithm maintains a set $\mathcal{I} \subset [N_\mathrm{P}]$ of parameter indices, called *identified* parameters. At each iteration $\ell \in \mathbb{N}$, in Line 4, the algorithm determines the set $\mathcal{I}$ from the current sensor placement $\mathbf{s}_\ell = \{s_{\ell,1}, \ldots, s_{\ell,N_\mathrm{L}}\}$ recursively as

$$\mathcal{I} := \mathcal{I} \cup \{n : \exists k \in [N_\mathrm{L}] \text{ with } \mathsf{x}_{s_{\ell,k}} \in \mathcal{R}_n^{\mathrm{sup}}\}. \tag{2.5}$$

Informally, for each $n \in \mathcal{I}$, at least one sensor is placed within the region of significant support of ϕ_n in at least one iteration of the algorithm including the ℓ^{th} iteration. The complement of $\mathcal{I}$ is $\mathcal{I}^{\mathrm{c}} := [N_\mathrm{P}] \backslash \mathcal{I}$.

The algorithm also maintains a set of indices $\mathcal{L} = \{m_1, \ldots, m_{N_\mathrm{L}}\} \subset [N_\mathrm{P}]$. The sensor reconfiguration policy determines $\mathcal{L}$ as discussed in Section 2.4. The size of this set $\mathcal{L}$ is $N_\mathrm{L} \leqslant N_\mathrm{S}$. Informally, these indices are such that the k^{th} sensor is placed in the region of estimability $\mathcal{R}_{m_k}^{\mathrm{est}}$ of the m_k^{th} basis function. From a practical perspective, the placement $\mathbf{s}_\ell$ is interpreted as *new* locations for sensors to be placed, and that N_L out of N_S sensors move at each iteration.

Next in Line 5, the parameter estimate $\hat{\Theta}_\ell$, and the estimation error covariance P_ℓ are computed from the sensor measurements $\mathbf{z}_\ell \in \mathbb{R}^{N_\mathrm{L}}$. The well-posedness of the least squares estimation

problem is ensured as follows. The matrix $T \in \mathbb{R}^{N_{\mathrm{L}} \times N_{\mathrm{P}}}$ is defined as

$$T_{ij} := \begin{cases} 1 & \text{if } j = m_k, \\ 0 & \text{otherwise.} \end{cases}$$

for $k \in [N_{\mathrm{L}}], j \in [N_{\mathrm{P}}]$. Next, H_{wp} is defined as

$$H_{\mathrm{wp}} := \begin{bmatrix} \phi_{m_1}(\mathsf{x}_{s_{\ell,1}}) & \phi_{m_2}(\mathsf{x}_{s_{\ell,1}}) & \dots \phi_{m_{N_{\mathrm{L}}}}(\mathsf{x}_{s_{\ell,1}}) \\ \phi_{m_1}(\mathsf{x}_{s_{\ell,2}}) & \phi_{m_2}(\mathsf{x}_{s_{\ell,2}}) & \dots \phi_{m_{N_{\mathrm{L}}}}(\mathsf{x}_{s_{\ell,2}}) \\ & \dots & \\ \phi_{m_1}(\mathsf{x}_{s_{\ell,N_{\mathrm{L}}}}) & \phi_{m_2}(\mathsf{x}_{s_{\ell,N_{\mathrm{L}}}}) & \dots \phi_{m_{N_{\mathrm{L}}}}(\mathsf{x}_{s_{\ell,N_{\mathrm{L}}}}) \end{bmatrix}.$$

Recall that $\mathsf{x}_{s_{\ell,k}}$ denotes the grid point at which the k^{th} sensor is placed and is the closest available grid point to the center of $m_{k^{\mathrm{th}}}$ basis. The construction of the set $\mathcal{L}$ ensures sensor placement to maximize SNR ensures each row has a non-zero entry. Then by Definition 2, all columns of H_{wp} are linearly independent therefore the matrix H_{wp} has full rank N_{L}. The parameter estimates and estimation error covariance can then be computed using the standard recursive least squares method augmented with a transformation matrix T as follows:

$$P_{\mathrm{wp}} := T P_\ell T^{\mathrm{T}} \tag{2.6}$$

$$G_\ell := P_{\mathrm{wp}} H_{\mathrm{wp}}^{\mathrm{T}} (H_{\mathrm{wp}} P_{\mathrm{wp}} H_{\mathrm{wp}}^{\mathrm{T}} + R)^{-1}, \tag{2.7}$$

$$\hat{\Theta}_{\ell+1} := \hat{\Theta}_\ell + T^{\mathrm{T}} G_\ell (\mathbf{z}_\ell - H_{\mathrm{wp}} T \hat{\Theta}_\ell), \tag{2.8}$$

$$P_{\ell+1} := (\mathbb{I}_{(N_{\mathrm{P}})} - T^{\mathrm{T}} G_\ell H_{\mathrm{wp}} T) P_\ell. \tag{2.9}$$

This recursive method is initialized with $\hat{\Theta}_0 := \mathbf{0}$, and $P_0 := \lambda_0 \mathbb{I}_{(N_{\mathrm{P}})}$, where λ_0 is an arbitrary large constant. The threat field is then estimated as $\hat{c}(\mathsf{x}) := 1 + \Phi^{\mathrm{T}}(\mathsf{x}) \hat{\Theta}_\ell$. We refer to this threat field estimate as an *optimistic estimate* because for all indices $m \in \mathcal{I}^{\mathrm{C}}$, the parameter $\hat{\theta}_m$ retains its initialized value of zero. This property contributes towards the optimality of the proposed algorithm, as discussed in Section 2.6.

Definition 2 implies that the parameter estimates $\hat{\theta}_n$ can be treated, for practical purposes, as mutually independent random variables (r.v.). Consequently, at each iteration, N_{L} sensors can be

Interactive Planning and Sensing

1: Set initial sensor placement $\mathbf{s}_0 \subset \{1, \ldots, N_{\mathrm{G}}\}$.
2: Set $\ell := 0$, $StopCondition :=$ false, and $\mathcal{I} := \varnothing$.
3: **while** $\neg StopCondition$ **do**
4: Find the set $\mathcal{I}$ by (2.5).
5: Update $\hat{\Theta}_\ell$, P_ℓ, by (2.7)–(2.9) and construct $\hat{c}(\mathsf{x})$.
6: **if** $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] \leqslant \varepsilon_2(\mathbf{v}_\ell^*)$ **then**
7: Execute Dijkstra's algorithm to find $\mathbf{v}_\ell^*$.
8: SENSOR RECONFIGURATION
9: Set $\ell := \ell + 1$, and evaluate $StopCondition$.

Figure 2.1: Pseudocode for the proposed IPAS algorithm to solve Problem 1.

placed to estimate N_{L} parameters – i.e., the other $(N_{\mathrm{P}} - N_{\mathrm{L}})$ parameters are treated as "nuisance variables" for that iteration (Routtenberg and Tong, 2015), which leads to the following important feature of the proposed algorithm.

Remark 1. The least-squares estimation problem in Line 5 is well-posed, and the the estimates $\hat{\theta}_n$ are unbiased for each identified parameter, i.e., for each $n \in \mathcal{I}$.

Using the transformation matrix T, (2.7)–(2.9) provide the computational convenience of maintaining the sizes of $\hat{\Theta}_\ell$ and P_ℓ constant at each iteration.

In Line 7, Dijsktra's algorithm is executed to determine an optimal path $\mathbf{v}_\ell^*$ in $\mathcal{G}$, i.e., $\mathbf{v}_\ell^*$ has minimum estimated cost with the estimate of the threat field at the ℓ^{th} iteration. Next, the sensor reconfiguration policy is executed to obtain a new sensor placement $\mathbf{s}_{\ell+1}$. The search for a new optimal path in Line 7 is conditional on the confidence in the estimated cost of the current path (Line 6). Specifically, a new optimal path is searched in Line 7 only if the variance of estimated path cost is lower than a threshold $\varepsilon_2(\mathbf{v}_\ell^*)$. Exact expressions of the path cost variance and a definition of the threshold $\varepsilon_2(\mathbf{v}_\ell^*)$ are provided in Section 2.5.

Sensor Reconfiguration

SENSOR RECONFIGURATION

1: Find the set $\mathcal{K}$ by Eqn. (2.10).
2: Find the set $\mathcal{L}$ according to Eqn. (2.11).
3: For each $m \in \mathcal{L}$, place a sensor in $\mathcal{R}_m^{\text{est}}$ to get $\mathbf{s}_{\ell+1}$.

Figure 2.2: Pseudocode for SENSOR RECONFIGURATION *procedure called at Line 17 of main algorithm in Fig 2.1.*

2.4 Task-Driven Sensor Reconfiguration

The sensor reconfiguration policy in the proposed IPAS algorithm is tailored to collect measurements that are of most relevance to the "task" (i.e., the path planning problem) at hand. To this end, this policy first identifies (Line 1) a minimal set of indices $\mathcal{K} \subseteq [N_{\text{P}}]$ such that the current path lies within the total region of significant support defined by the corresponding basis functions. To be precise, let $\mathcal{V}_\ell \subset \mathcal{W}$ be the set of grid point locations in the workspace associated with each of the vertices in the path $\mathbf{v}_\ell^*$. Then

$$\mathcal{K} := \{n \in [N_{\text{P}}] : \mathcal{V}_\ell \cap \mathcal{R}_n^{\text{sup}} \neq \varnothing\}. \tag{2.10}$$

We call the set $\mathcal{K}$ the *minimal cover* of the path $\mathbf{v}_\ell^*$.

The rationale of the proposed IPAS algorithm is that, typically, $\mathcal{K}$ has fewer than N_{P} elements. The parameters θ_n for $n \in \mathcal{K}$ are considered to be of the "most relevance" to the path-planning problem. Therefore, sensors can be placed to improve the confidence in the estimation of these parameters, possibly at the expense of reduced confidence in estimates of the other remaining parameters, i.e., those θ_n where $n \notin \mathcal{K}$.

Next, in Line 2, we compute a set of indices $\mathcal{L}$ defined as

$$\mathcal{L} := \begin{cases} \mathcal{K} & \text{if } \mathcal{K} \cap \mathcal{I}^{\text{C}} = \varnothing, \\ \mathcal{K} \cap \mathcal{I}^{\text{C}} & \text{otherwise.} \end{cases} \tag{2.11}$$

Informally, $\mathcal{K} \cap \mathcal{I}^{\text{C}}$ is the set of indices of parameters that are of relevance to the path-planning

problem but not yet identified. The sensor placement $\mathbf{s}_{\ell+1}$ in Line 3 consists of locations within the region of estimability of basis function ϕ_m, for each $m \in \mathcal{L}$. Ideally, sensors should be placed at grid points closest to the locations $\bar{\mathsf{x}}_m$ for each $m \in \mathcal{L}$, i.e., the maxima of the basis functions ϕ_m.

The situation $\mathcal{K} \cap \mathcal{I}^{\mathrm{C}} = \varnothing$ can occur when the path $\mathbf{v}_\ell^*$ inside the region $\bigcup_{n\in\mathcal{K}} \mathcal{R}_n^{\mathrm{sup}}$ has not fulfilled $StopCondition$ described in Section 2.5. In this situation, the sensor reconfiguration policy considers *all* parameters with indices in $\mathcal{K}$ to be relevant.

It is possible that $\mathcal{L}$ has more than N_{S} elements. In this case, $\mathcal{L}$ is sorted $\{m_1, m_2, \ldots, \}$ such that the distances of $\bar{\mathsf{x}}_{m_i}$ to their closest grid points are in non-decreasing order with increasing i. Then $\mathcal{L}$ is redefined as $\mathcal{L} := \{m_1, m_2, \ldots, m_{N_{\mathrm{S}}}\}$. Recall that $\bar{\mathsf{x}}_{m_i}$ is the maximum of the basis function ϕ_{m_i}, and therefore this ordering of $\mathcal{L}$ is equivalent to ordering by SNR for sensor measurements.

After computing the set $\mathcal{L}$ with $N_{\mathrm{L}} \leqslant N_{\mathrm{S}}$ elements, as above, the sensor reconfiguration policy finds a placement $\mathbf{s}_{\ell+1}$ such that for each $m \in \mathcal{L}$, there is at least one sensor placed within the region of estimability $\mathcal{R}_m^{\mathrm{est}}$ of the basis ϕ_m (Line 3).

2.5 Termination Criterion

The termination criterion for this algorithm, denoted by the boolean variable $StopCondition$, is based on the variance of the estimated path cost variance, which is in turn computed from the parameter estimation error covariance. Because the path cost depends linearly on the parameters, it is a Gaussian r.v. Because parameter estimates are unbiased, the path cost estimate is also unbiased whenever all parameters with indices in the set $\mathcal{K}$ are identified. Note that the sensor reconfiguration policy *ensures* that parameters with indices in $\mathcal{K}$ become identified as the algorithm iterates.

At the ℓ^{th} iteration of the IPAS algorithm, the variance of the estimated cost of path $\mathbf{v}_\ell^* = (v_0, \ldots, v_P)$ is

$$\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] = \mathrm{Var}[\textstyle\sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i})\hat{\Theta}_\ell]$$

$$= \sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) P_\ell \Phi(\mathsf{x}_{v_i}). \tag{2.12}$$

The boolean $StopCondition$ is $true$ if $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] \leqslant \varepsilon_1(\mathbf{v}_\ell^*)$, or is $false$ otherwise. Here, $\varepsilon_1(\mathbf{v}_\ell^*)$ is a path-dependent threshold. chosen based on the grid resolution. Consider the hypothetical ideal case where a sensor can be placed at every grid point, and the measurement error variance of each sensor is $\bar{\sigma}$. Then the estimation error covariance is $P_{\mathrm{grid}} := (H_{\mathrm{grid}}^{\mathrm{T}} R^{-1} H_{\mathrm{grid}})^{-1}$, where

$$H_{\mathrm{grid}} := \begin{bmatrix} \Phi(\mathsf{x}_1) & \Phi(\mathsf{x}_2) & \dots & \Phi(\mathsf{x}_{N_{\mathrm{G}}}) \end{bmatrix}^{\mathrm{T}},$$

$R := \bar{\sigma}\mathbb{I}_{(N_{\mathrm{G}})}$. Then we define

$$\varepsilon_1(\mathbf{v}_\ell^*) := \lambda_1^2 \sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) P_{\mathrm{grid}} \Phi(\mathsf{x}_{v_i}), \tag{2.13}$$

$$\varepsilon_2(\mathbf{v}_\ell^*) := \lambda_2^2 \sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) P_{\mathrm{grid}} \Phi(\mathsf{x}_{v_i}), \tag{2.14}$$

where $\lambda_2 > \lambda_1 \geqslant 1$ are prespecified constants. Note that P_{grid} can be computed a priori, i.e., without taking any measurements at all.

The variance computation in (2.12) is well-defined because the least-squares estimation problem in Line 5 is well-posed, per Remark 1. Due to the unbiasedness of parameter estimates, the proposed algorithm can also reduce the variance and mean square error, *both,* of the estimated path cost to arbitrarily small values.

2.5.1 Relationship with Fisher Information

Fisher information and the Fisher information matrix (FIM), is closely related to the formulation of the termination criteria as well as estimation problems in general. Formally, the Fisher information matrix has entries

$$[\mathcal{I}(\theta)]_{i,j} = \mathbb{E}\left[\left(\frac{\partial}{\partial \theta_i} log f(X, \theta) \right) \left(\frac{\partial}{\partial \theta_j} log f(X, \theta) \right) \middle| \theta \right] \tag{2.15}$$

where under certain regularity conditions can alternatively be formulated as

$$[\mathcal{I}(\theta)]_{i,j} = -\mathbb{E}\left[\left(\frac{\partial^2}{\partial\theta_i\theta_j} log f(X,\theta)\right)\middle|\theta\right] \tag{2.16}$$

where $\mathbb{E}(\cdot)$ is the expectation operator.

The fisher information, and its matrix variant, is particular useful for capturing the lower bound of the variance on an estimator. For an unbiased estimator, the famous Cramer Rao lower bound (CRLB) is given by

$$Cov(\hat{\theta}) \succeq \frac{1}{\mathcal{I}(\theta)}. \tag{2.17}$$

The above form of the CRLB states that, any unbiased estimator $\hat{\theta}$ cannot have a lower variance than the inverse of the information matrix $\mathcal{I}(\theta)$. For the estimators studied in this dissertation, namely the least squares estimator and later the Kalman filter, the estimators are both unbiased and achieve the CRLB. Therefore, with respect to the notation presented

$$P = \mathcal{I}^{-1}(\theta). \tag{2.18}$$

In other words, the error covariance is the inverse of the Fisher information matrix.

Consequently, another way to state the termination criteria is with respect to information, i.e.

$$\varepsilon_1(\mathbf{v}_\ell^*) := \lambda_1^2 \textstyle\sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) \mathcal{I}_{\mathrm{grid}}^{-1}(\theta) \Phi(\mathsf{x}_{v_i}), \tag{2.19}$$

or alternatively,

$$\varepsilon_{\mathcal{I}}(\mathbf{v}_\ell^*) := \gamma^2 \textstyle\sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) \mathcal{I}_{\mathrm{grid}}(\theta) \Phi(\mathsf{x}_{v_i}), \tag{2.20}$$

where $\gamma < 1$ as to reflect the inverse relationship to $\lambda_1 > 1$. The revised stop threshold constant γ reflects the concept that $\mathcal{I}(\theta)$ represents the maximum amount of information attainable if sampling every grid location, and γ reduces the requirement on the amount of information needed in order

to terminate. The corresponding *path cost information* would be represented as

$$\mathcal{I}[\hat{J}(\mathbf{v}_\ell^*)] = \sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) \mathcal{I}_\ell(\theta) \Phi(\mathsf{x}_{v_i}). \tag{2.21}$$

Then, of course, the boolean $StopCondition$ is flipped and $true$ if $\mathcal{I}[\hat{J}(\mathbf{v}_\ell^*)] \succcurlyeq \varepsilon_{\mathcal{I}}(\mathbf{v}_\ell^*)$, or is $false$ otherwise. Rather than reducing variance below the threshold, the aim is to increase the information until the information threshold is met.

For the problem explored in this dissertation, finding and reducing the uncertainty of the optimal path, the variance formulation is more intuitive. However, other tasks such as target tracking may find that the information form of the termination criteria better aligns with the problem formulation.

2.6 Convergence Conditions and Optimality

In this section, we show that the proposed IPAS algorithm terminates in a finite number of iterations. We then provide the main result of this paper: upon termination the algorithm provides a near-optimal path.

Proposition 1. *The IPAS algorithm terminates in a finite number of iterations whenever the number of parameters N_{P} is finite, for any number of sensors $N_{\mathrm{S}} > 0$.*

Proof. Because the number of parameters N_{P} is finite, one of two cases must occur: (1) either all parameters are identified, i.e., $\mathcal{I} = [N_{\mathrm{P}}]$, or (2) a proper subset of parameters is identified, i.e., $\mathcal{I} \subset [N_{\mathrm{P}}]$. In either case, there is a finite $L \in \mathbb{N}$ such that the set $\mathcal{I}$ becomes invariant after the L^{th} iteration. Therefore, for each $\ell > L$, $\mathcal{K}_\ell \cap \mathcal{I} = \varnothing$, and $\mathcal{L} = \mathcal{K}$ becomes invariant. To see this, note that otherwise a sensor is placed in the region of estimability of at least one parameter with index in $\mathcal{I}^{\mathrm{C}}$ by (2.11), and consequently this parameter is identified by (2.5), and $\mathcal{I}$ changes in the next iteration.

By definition of the minimal cover $\mathcal{K}$, for the path $\mathbf{v}_\ell^* = (v_0, \ldots, v_P)$, the values of $\phi_n(\mathsf{x}_{v_i}) \approx 0$

for each $i = 0, \ldots, P$ and $n \notin \mathcal{K}$. Therefore, by (2.12),

$$\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] = \textstyle\sum_{i=0}^{P} \Phi_{[\mathcal{K}]}^{\mathrm{T}}(\mathsf{x}_{v_i})\; P_\ell|_{[\mathcal{K}]}\; \Phi_{[\mathcal{K}]}(\mathsf{x}_{v_i}),$$

where $P_\ell|_{[\mathcal{K}]}$ is the principal submatrix obtained by deleting rows and columns of P_ℓ that are not in the set $[\mathcal{K}]$, and $\Phi_{[\mathcal{K}]}(\mathsf{x}) := [\ \phi_{m_1}(\mathsf{x}) \quad \phi_{m_2}(\mathsf{x}) \quad \ldots\ ]^{\mathrm{T}}$, for $m_1, m_2, \ldots \in [\mathcal{K}.]$ In (2.6)–(2.9), P_{wp} is then the same as $P_\ell|_{[\mathcal{K}]}$, and only the elements of this principal submatrix are updated in P_ℓ at each iteration. Because the parameters to be estimated are constant, it follows from standard recursive least squares theory (Stengel, 1994) that $P_\ell|_{[\mathcal{K}]}$, reduces monotonically at each iteration as long as at least one new measurement is taken, which is ensured by the proposed sensor reconfiguration policy.

Therefore, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ also decreases monotonically, and $\inf\{\mathrm{Var}[\hat{J}(\mathbf{v}^*)]\}_{\ell\in\mathbb{N}} = 0$. After a finite number of iterations, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ reduces below $\varepsilon_1(\mathbf{v}_\ell^*)$, and $StopCondition$ becomes true. □

In what follows, we denote by $\bar{\ell}$ the number of iterations required for the IPAS algorithm to terminate and converge to the solution of Problem 1.

Note that the preceding proof involves an approximation, namely, that the values of basis functions outside of their regions of support are zero, whereas these are actually small non-zero values. However, arbitrary precision can be achieved by redefining the regions of significant support to include larger regions. This redefinition will have the effect of making the minimal cover larger for each path, and therefore the algorithm will take additional iterations to terminate.

The next result follows immediately due to Prop. 1, due to the definition of $StopCondition$, and due to the fact that the path cost estimate is a Gaussian r.v.

Corollary 1. $\mathbb{P}\left[\mathcal{J}(\mathbf{v}_{\bar{\ell}}^*) - \hat{J}(\mathbf{v}_{\bar{\ell}}^*) \leqslant 3\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)}\ \right] \geqslant 0.9987.$

Line 6 of the proposed algorithm prevents the algorithm from discarding potentially low-cost paths before a high confidence is reached in the estimated costs of such paths. Furthermore, the aforesaid optimistic estimate of the threat field ensures that the algorithm "explores" all relevant regions of the workspace. These features of the algorithm are formally described in the following

result.

Lemma 1. *Let $\mathcal{I}$ be the set of identified indices at the termination of the IPAS algorithm after $\bar{\ell}$ iterations. Let $\mathbf{v}_{\mathcal{I}}^*$ be a path of minimum true cost, such that the grid points of all vertices in this path lie within the region $\cup_{k\in\mathcal{I}}\mathcal{R}_k^{\sup}$. Then*

$$\mathbb{P}\left[\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}_{\mathcal{I}}^*) \leqslant 3\sqrt{\varepsilon_2(\mathbf{v}_n^*)}\right] \geqslant 0.9974. \tag{2.22}$$

Proof. If there is an iteration $n \leqslant \bar{\ell}$ such that $\mathbf{v}_n^*$ is identical to $\mathbf{v}_{\mathcal{I}}^*$, then (2.22) is true due to Line 6 of the proposed algorithm and due to the fact that the estimated cost is a Gaussian r.v.

Suppose there is no iteration where the path found in Line 7 is identical to $\mathbf{v}_{\mathcal{I}}^*$, and consider some iteration $n \leqslant \bar{\ell}$. By Line 6, $\mathrm{Var}[\hat{J}(\mathbf{v}_n^*)] \leqslant \varepsilon_2(\mathbf{v}_n^*)$ before the algorithm terminates. Consider the set of indices $\mathcal{K}$ defined in (2.10). By the sensor configuration policy (2.11), measurements are taken in the regions of estimability of all basis functions with indices in $\mathcal{K}$. Because $\varepsilon_2(\mathbf{v}_n^*)$ is path-dependent as defined in (2.14), it follows that for *every* path $\mathbf{v}$ whose minimal cover is $\mathcal{K}$, $\mathrm{Var}[\hat{J}(\mathbf{v})] \leqslant \varepsilon_2(\mathbf{v})$. Furthermore, by (2.12), for each $k \in \mathcal{K}$, the k^{th} diagonal element of P_n is small enough such that $\mathrm{Var}[\hat{J}(\mathbf{v})] \leqslant \varepsilon_2(\mathbf{v})$.

By the definition (2.5) of $\mathcal{I}$, and by the sensor configuration policy (2.11), an index $k \in [N_{\mathrm{P}}]$ is included in $\mathcal{I}$ only if k is in the minimal cover of the optimal path found in Line 7 at some iteration of the proposed algorithm before termination. It follows that, at termination, for each $k \in \mathcal{K}$, the k^{th} diagonal element of $P_{\bar{\ell}}$ is small enough such that $\mathrm{Var}[\hat{J}(\mathbf{v}^*)] \leqslant \varepsilon_2(\mathbf{v}^*)$ for *any* path $\mathbf{v}$ whose minimal cover is a subset of $\mathcal{I}$.

In particular, $\mathrm{Var}[\hat{J}(\mathbf{v}_{\mathcal{I}}^*)] \leqslant \varepsilon_2(\mathbf{v}_{\mathcal{I}}^*)$. It follows that

$$\mathbb{P}\left[\hat{J}(\mathbf{v}_{\mathcal{I}}^*) - \mathcal{J}(\mathbf{v}_{\mathcal{I}}^*) \leqslant 3\sqrt{\varepsilon_2(\mathbf{v}_n^*)}\right] \geqslant 0.9974.$$

Because Dijkstra's algorithm is optimal in Line 7, $\hat{J}(\mathbf{v}_{\mathcal{I}}^*) \geqslant \hat{J}(\mathbf{v}_{\bar{\ell}}^*)$, and the result (2.22) follows. □

Next, we state the main result of the paper as follows.

Theorem 1. *Let $\bar{\ell} \in \mathbb{N}$ be the iteration at which the IPAS algorithm terminates, and let $\mathcal{I}$ be the*

set of identified indices at termination. The path $\mathbf{v}_{\bar{\ell}}^*$ *satisfies either*

$$\mathbb{P}\left[\mathcal{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \leqslant 3(\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)} + \sqrt{\varepsilon_2(\mathbf{v}^*)})\right] \geqslant 0.9948,$$

or, if $\mathcal{I} \neq [N_{\mathrm{P}}]$, $\mathbb{P}\left[\mathcal{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \sim O(\sqrt{\frac{N_{\mathrm{G}}}{10}})\right] \geqslant 0.9974.$

Proof. As discussed in the proof of Prop. 1, after a finite number of iterations, the set $\mathcal{I}$ becomes invariant. First, consider the case that either $\mathcal{I} = [N_{\mathrm{P}}]$, and/or $\mathbf{v}^*$ is covered by $\cup_{k\in\mathcal{I}}\mathcal{R}_k^{\sup}$. By Lemma 1, $\mathbb{P}\left[\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \leqslant 3\sqrt{\varepsilon_2(\mathbf{v}^*)}\right] \geqslant 0.9974$. By Corl. 1,

$$\begin{aligned}\mathbb{P}\Big[\Big[\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \leqslant 3\sqrt{\varepsilon_2(\mathbf{v}^*)}\Big] \text{ and} \\ \Big[\mathcal{J}(\mathbf{v}_{\bar{\ell}}^*) - \hat{J}(\mathbf{v}_{\bar{\ell}}^*) \leqslant 3\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)}\,\Big]\Big] \geqslant 0.9974^2 = 0.9948,\end{aligned}$$

and it follows that

$$\mathbb{P}\left[\mathcal{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \leqslant 3(\sqrt{\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)} + \sqrt{\varepsilon_2(\mathbf{v}^*)})\right] \geqslant 0.9948.$$

Second, consider the case that $\mathcal{I} \neq [N_{\mathrm{P}}]$, and $\mathbf{v}^*$ does not entirely lie within $\cup_{k\in\mathcal{I}}\mathcal{R}_k^{\sup}$, i.e., there is at least one vertex $i \in V$ that belongs to the path $\mathbf{v}^*$ but lies outside the regions of significant support of bases with indices in $\mathcal{I}$. Recall that the threat field estimate is optimistic, i.e., for grid points x_i such that $\mathsf{x}_i \notin \cup_{k\in\mathcal{I}}\mathcal{R}_k^{\sup}$, $\hat{c}(\mathsf{x}_i) \leqslant 1 + 0.011\sum_{k\in\mathcal{I}}\bar{\theta}_k$ because the maximum value of any basis function outside its region of significant support is 0.011, by (2.2).

The true value of the threat field at this grid point is $c(\mathsf{x}_i) \geqslant 1 + 0.011\sum_{k\in\mathcal{I}}\underline{\theta}_k$. Therefore, the regret associated with not including i is at most $\hat{c}(\mathsf{x}_i) - c(\mathsf{x}_i) \leqslant 0.011\sum_{k\in\mathcal{I}}(\bar{\theta}_k - \underline{\theta}_k)$. The number of such vertices is at most $O(\sqrt{N_{\mathrm{G}}})$ because threat values are strictly positive everywhere and optimal paths will have minimal numbers of vertices. It follows that $\hat{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \sim 0.022\sum_{k\in\mathcal{I}}(\bar{\theta}_k - \underline{\theta}_k)O(\sqrt{N_{\mathrm{G}}}) \sim O(10^{-2}\sqrt{N_{\mathrm{G}}})$ and by consequence,

$$\mathbb{P}\left[\mathcal{J}(\mathbf{v}_{\bar{\ell}}^*) - \mathcal{J}(\mathbf{v}^*) \sim O(10^{-2}\sqrt{N_{\mathrm{G}}})\right] \geqslant 0.9974.$$

□

Remark 2. Note that the preceding proof relies on the unbiasedness (per Remark 1) of estimates of identified parameters. Otherwise, the estimate of the regret of not including a grid point i such that $\mathsf{x}_i \notin \cup_{k\in\mathcal{I}}\mathcal{R}_k^{\sup}$, will involve the bias of estimates of identified parameters.

Informally, the optimistic threat estimate causes the proposed algorithm to search for an optimal path through regions of significant support of bases with non-identified parameters. If the estimates of identified parameters have a negative bias, then, in turn, the search for optimal paths is biased to regions within the regions of significant support of bases with already identified parameters.

Chapter 3

Path-planning with Waiting in Time-varying Environments

3.1 Problem Formulation

In what follows, $\mathbb{R}$ and $\mathbb{Z}$ represent the sets of reals and integers, respectively.

3.1.1 Threat Field Parametrization

We consider a threat field constructed as the weighted sum of a finite number of 2D Gaussian basis functions $c(\mathsf{x}, t) = \sum_{n=1}^{N_\mathrm{P}} w_n(t)\phi_n(\mathsf{x}, t)$. Here, the spatial basis function ϕ_n is defined for each $n = 1, \ldots, N_\mathrm{P}$ by

$$\phi_n(\mathsf{x}, t) := \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-(\mathsf{x} - \mu_n(t))^\mathrm{T}(\mathsf{x} - \mu_n(t))}{2\|\Sigma_n(t)\|^2}\right), \tag{3.1}$$

where $\mu_n(t) = (\mu_{nx}(t), \mu_{ny}(t))$ defines the spatial mean of ϕ_n and $\Sigma_n(t) = (\sigma_{nx}(t), \sigma_{ny}(t))$ defines the spatial spread of ϕ_n. In this work, we consider affine functions μ_n and Σ_n of the form $\mu_n(t) = \mu_{n0} + \mu_{n1}t$, and $\Sigma_n(t) = \Sigma_{n0} + \Sigma_{n1}t$, where $\mu_{n0}, \mu_{n1}, \Sigma_{n0}, \Sigma_{n1} \in \mathbb{R}^2$ are prespecified constants.

The finite parameterization of the threat field c using Gaussian functions is justified by the fact that a large class of functions on $\mathbb{R}$, namely, square integrable functions, can be approximated with arbitrary precision by linear combinations of Gaussian functions (Calcaterra, 2008). Furthermore, 2D Gaussian functions of varying width can be used to represent varying levels of resolution by including narrower Gaussian functions into higher resolution images or video (Goshtasby and Oneill, 1994). Finally, Gaussian function appear in series solutions to several partial differential equations such as the diffusion equation (Crank, 1979), which enables the application of the pro-

posed work to path- and motion-planning with threat field modeled by physical phenomena such as advection-diffusion of gases or radiation in the atmosphere (Demetriou et al., 2013).

3.1.2 Path-planning with Uniformly High Resolution Field Map

Let $\mathcal{W} \subset \mathbb{R}^2$ be a closed square region, called the *workspace*, in which the vehicle moves. We consider a strictly positive spatiotemporal scalar field $c : \mathcal{W} \times [0, \infty) \to \mathbb{R}_{>0}$, called the *threat field*, which represents unfavorable regions with higher intensity. For path-planning, we restrict time to a compact interval $T = [t_0, t_\mathrm{f}] \subset \mathbb{R}_+$. The threat field may represent, for instance, terrain elevation (Pai and Reissell, 1998), a risk measure (Tsiotras and Bakolas, 2007), a probabilistic occupancy grid (Elfes, 1989), or the atmospheric concentration of a gas (Demetriou et al., 2013).

First, we consider path-planning on a grid consisting of N_G points uniformly placed in N_R rows and N_R columns. The coordinates in a prespecified Cartesian coordinate axis system of the j^{th} grid point are denoted by x^j, for each $j = 1, ..., N_\mathrm{G}$. The vehicle is assumed to traverse grid points according to a "$4-$connectivity" rule, and the time taken to traverse between adjacent grid points is a prespecified constant t_step. In this paper, we neglect vehicle kinematic and dynamic constraints that can restrict this motion, while noting that such constraints can in the future be incorporated in the proposed grid-world problem setup (Cowlagi and Tsiotras, 2012a).

We define a graph $\mathcal{G} = (V, E)$, where each vertex in V is uniquely associated with a grid point, and labeled with superscripts as $v^1, v^2, \ldots, v^{N_\mathrm{G}}$. The edge set E is defined as the set of pairs of vertices associated with adjacent grid points. A *path between vertices* $v^{i_\mathrm{s}}, v^{i_\mathrm{g}} \in V$, denoted $\mathbf{v}(i_\mathrm{s}, i_\mathrm{g})$, is a finite sequence of vertices $(v_0, v_1, \ldots, v_P)$, with $v_0 = v^{i_\mathrm{s}}$, $v_P = v^{i_\mathrm{g}}$, and either $(v_{j-1}, v_j) \in E$ or $v_{j-1} = v_j$, for each $i \in \{1, \ldots, P\}$. Note that *sub*scripts are used to denote indices of vertices. The *cost* of the path $\mathbf{v}(i_\mathrm{s}, i_\mathrm{g})$ is defined by $\mathcal{J}(\mathbf{v}) := \sum_{j=1}^{P} g((v_{j-1}, v_j), jt_\mathrm{step})$, where $g : E \times [0, \infty) \to \mathbb{R}_{>0}$ is a strictly positive function that assigns time-varying edge transition costs. Specifically,

$$g((v^k, v^\ell), t) = \begin{cases} c(\mathsf{x}^\ell, t) + \alpha_\mathrm{m}, & \text{if } v^k \neq v^\ell, \\ c(\mathsf{x}^\ell, t) + \alpha_\mathrm{w}, & \text{if } v^k = v^\ell. \end{cases} \tag{3.2}$$

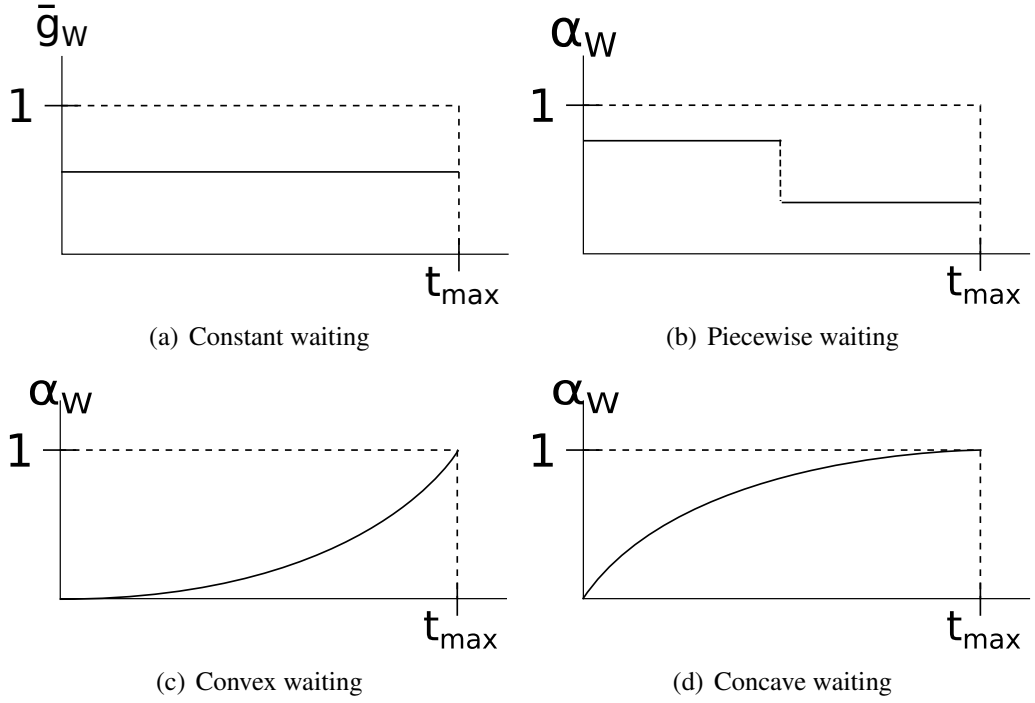
where α_{m}, and α_{w} are prespecified strictly positive constants. Note that the preceding definition of a path and its cost implicitly allows for *waiting* at any grid point, namely, instances where $\bar{v}_{j-1} = \bar{v}_j$ for any $j \in \{1, \ldots, P\}$. The constants α_{w} and α_{m} are costs of waiting and of moving, respectively.

Problem 2. *Find a path* $\mathbf{v}^*(i_{\mathrm{s}}, i_{\mathrm{g}})$ *such that* $\mathcal{J}\left(\mathbf{v}^*(i_{\mathrm{s}}, i_{\mathrm{g}})\right) \leqslant \mathcal{J}\left(\mathbf{v}(i_{\mathrm{s}}, i_{\mathrm{g}})\right)$ *for every other path* $\mathbf{v}(i_{\mathrm{s}}, i_{\mathrm{g}})$ *in* $\mathcal{G}$.

3.2 Waiting Policies and Considerations

A brief discussion of the waiting constant α_{w} will help illustrate the application of waiting and potential complexity involved. In this work, we restrict the cost of waiting to be a constant value regardless of the vehicle's location in space or time, see Fig. 3.1(a). However, in more advanced applications α_{w} can be a function of location, $\alpha_{\mathrm{w}}(\mathsf{x}_i)$, a function of time, $\alpha_{\mathrm{w}}(t)$, or a combination, $\alpha_{\mathrm{w}}(\mathsf{x}_i, t)$. Additionally, the waiting cost can be depend on the accumulation of time at a location, $\alpha_{\mathrm{w}}(\mathsf{x}_i, t, \tau)$, where τ is the duration of time after arriving at (x_i, t).

Waiting functions fall into two primary categories, location dependent and duration dependent. Location dependent waiting assigns a cost based on the current position both in space and time. For example, a parking garage in the downtown area likely has a higher cost than a garage on the edge of town. The same garage may have time dependent costs such as different parking costs for business and nighttime hours, Fig. 3.1(b). Duration dependent waiting costs penalize waiting in a location for an extended period of time, such as parking garages charging increasing rates for one, four, and eight hours. These duration dependent rates often have concave and convex shapes to the waiting cost function, Fig. 3.1(c) and Fig. 3.1(d). Parking garages rates often level off in a concave manner, whereas the exponential health risk of exposure to radiation is convex (Thomas and Scotto, 1983).



(a) Constant waiting (b) Piecewise waiting

(c) Convex waiting (d) Concave waiting

Figure 3.1: Different waiting policies based on applications.

3.3 General Solution to Uniform High Resolution Path-planning Problem

Problem 2 is similar to a standard path-planning problem with the exception that the edge transition costs are time-varying. This exception, however, significantly complicates the solution of this problem. Specifically, to solve Problem 2, a search must be performed not on the graph $\mathcal{G}$ but instead on a much larger graph $\mathcal{GT}$ obtained as follows. Let $\mathcal{T}$ be a line graph whose vertices are the time instants $t_0, t_1, \ldots$, and whose edges are the pairs (t_{j-1}, t_j), where $t_j - t_{j-1} = t_{\text{step}}$ for each $j \in \mathbb{N}$. The graph $\mathcal{GT}$ is then defined as the *product* of graphs $\mathcal{G}$ and $\mathcal{T}$, i.e., each vertex in $\mathcal{GT}$ is a pair (v^k, t_j), and the pairs (v^k, t_j) and (v^ℓ, t_m) form an edge of $\mathcal{GT}$ if and only $t_m - t_j = t_{\text{step}}$ and either $v^k = v^\ell$ or $(v^k, v^\ell) \in E$, with $k, \ell \in \{1, \ldots, N_{\text{G}}\}$ and $j, m \in \mathbb{N}$. The worst-case complexity of Dijkstra's algorithm for solving Problem 2 is $\mathcal{O}(|\mathcal{GT}|(1 + \log|\mathcal{GT}|))$.

In typical path-planning algorithms, Dijkstra's algorithm is sped up using a *search heuristic*: for example, the A* algorithm uses Euclidean distance to the goal as a search heuristic (Nilsson,

1998). For Problem 2, we can define for each vertex $v^k \in V$, $k = 1, 2, \ldots, |V|$, we define the search heuristic

$$h(v^k) := \alpha_{\mathrm{m}} \|\mathsf{x}^{i_{\mathrm{g}}} - \mathsf{x}^k\|, \tag{3.3}$$

which is Euclidean distance to the goal scaled by the positive constant α_{m}. The scale factor ensures that the heuristic h underestimates the true cost to the goal, and therefore ensures that the search algorithm returns an optimal path (Nilsson, 1998). Note that the *worst-case* computational complexity of the search algorithm does not change despite using the search heuristic (Nilsson, 1998), although a significant speed-up is achieved for most $(v^{i_{\mathrm{s}}}, v^{i_{\mathrm{g}}})$ pairs of initial and goal vertices.

3.3.1 No-Wait Path-planning

An approximate solution to Problem 2 can be obtained by ignoring the possibility of waiting. To this end, we define a *no-wait path* as a path $\mathbf{v} = (v_0, v_1, \ldots, v_P)$ where $v_{j-1} \neq v_j$ for each $i \in \{1, \ldots, P\}$. We define an edge transition cost function:

$$g_{\mathrm{nw}}((v^k, v^\ell), t) := c(\mathsf{x}^\ell, t) + \alpha_{\mathrm{m}}. \tag{3.4}$$

The cost of a no-wait path is then defined as $\mathcal{J}_{\mathrm{nw}}(\mathbf{v}) := \sum_{j=1}^{P} g_{\mathrm{nw}}((v_{j-1}, v_j), jt_{\mathrm{step}})$.

Compared to solving Problem 2 *exactly* as in Section 3.3, it is significantly easier to find a path $\mathbf{v}^*_{\mathrm{nw}}(i_{\mathrm{s}}, i_{\mathrm{g}})$ with minimum no-wait cost $\mathcal{J}_{\mathrm{nw}}(\mathbf{v}^*_{\mathrm{nw}})$. The worst-case complexity of Dijkstra's algorithm to find this path is $\mathcal{O}(|\mathcal{G}|(1 + \log|\mathcal{G}|))$, and $|\mathcal{G}|$ is smaller than $|\mathcal{GT}|$ by the order of magnitude of $|\mathcal{T}|$.

It is therefore natural to question: (1) whether the computational effort in *exactly* solving Problem 2 is worthwhile: i.e., whether the suboptimality of the $\mathbf{v}^*_{\mathrm{nw}}$, measured as $\mathcal{J}(\mathbf{v}^*_{\mathrm{nw}}) - \mathcal{J}(\mathbf{v}^*)$, is large enough to justify spending the significantly higher computational resources to find the true solution $\mathbf{v}^*$ to Problem 2, and (2) whether the exact solution $\mathbf{v}^*$ can be computed faster. In the sequel we address both of these questions.

To address the first question (on computational effort), we characterize the difference $\mathcal{J}(\mathbf{v}^*_{\mathrm{nw}}) -$

$\mathcal{J}(\mathbf{v}^*)$ based on a study of a large number of numerical simulations (discussed in Section B.3). The results of this study identify the threat fields where the difference $\mathcal{J}(\mathbf{v}^*_{\mathrm{nw}}) - \mathcal{J}(\mathbf{v}^*)$ is significant. This study consisted of 2000 simulations of different instances of Problem 2. In each simulation, the threat field c was constructed with an arbitrary N_{P} chosen from the set $\{3, 4, \ldots, 40\}$, followed by arbitrary choices of the constants w_{n0}, w_{n1}, μ_{n0}, μ_{n1}, Σ_{n0}, and Σ_{n1} for each $n = 1, \ldots, N_{\mathrm{P}}$. The weights w_{n0}, w_{n1} were fixed at 1 so that all peaks in the field are of uniform height. The parameters μ_{n0}, μ_{n1} were chosen by random sampling on a uniform distribution over the workspace $\mathcal{W}$. The parameters Σ_{n0}, Σ_{n1} were chosen by random sampling from a uniform distribution a quarter the size of the workspace $\mathcal{W}$. The simulations were run using $N_{\mathrm{G}} = 1600$, $t = [0, 100]$, and $t_{\mathrm{step}} = \frac{\mathcal{W}_{\mathrm{wd}}}{N_{\mathrm{R}}-1}$ where $\mathcal{W}_{\mathrm{wd}}$ is the width of the (square) workspace. The product graph $\mathcal{GT}$ in each simulation has approximately 1.2×10^6 vertices. The simulations were designed using MATLAB® and executed on a Windows 7 Enterprise® computer using an Intel® Core™ i7-4770 3.4 GHz CPU and 16 GB RAM . The results of this study are discussed in Section B.3, and MATLAB® source code is available at http://users.wpi.edu/~rvcowlagi/software.html.

The issue of interest here is whether it is beneficial to search in the larger product graph $\mathcal{GT}$ or the smaller topological graph $\mathcal{G}$. This issue depends on the properties of the threat field itself, and not on the specific software implementation of the search algorithm. Therefore, simulation results in MATLAB® suffice to study this issue. Real-time implementations of the search algorithms (e.g. in a lower-level language such as C/C++) are not necessary because they shed no further light.

To address the second question (whether the exact solution can be computed faster), we develop a method to prune search trees during the execution of Dijkstra's algorithm as discussed next.

3.4 Local Test for No-Wait Suboptimality

The number of computations in solving Problem 2 can be reduced by pruning edges of $\mathcal{GT}$ during the execution of Dijkstra's algorithm. Specifically, an edge of $\mathcal{GT}$ between the pairs (v^k, t_j) and (v^k, t_{j+1}), $k \in \{1, \ldots, N_{\mathrm{G}}\}$, $j \in \mathbb{N}$, can be pruned if a condition, as identified next, precludes waiting at x^k at time t_{j-1} from being optimal.

Consider the cost of moving from x^k to an adjacent point x^ℓ, with $k, \ell \in \{1, \ldots, N_{\mathrm{G}}\}$. Accordingly, consider the vertex (v^k, t_j) of $\mathcal{GT}$, and two paths for traveling to x^ℓ. The first path is the single edge in $\mathcal{GT}$ from vertex (v^k, t_j) to vertex (v^ℓ, t_{j+1}) achieves this travel. The second path consists of two successive edges from vertex (v^k, t_j) to vertex (v^k, t_{j+1}) to vertex (v^ℓ, t_{j+2}) also achieves this travel, but includes waiting at the point x^k for one time step (see Fig. 3.2). Waiting at x^k is beneficial if the cost of the first path is greater than the cost of the second path, i.e., if

$$g((v^k, v^k), t_{j+1}) + g((v^k, v^\ell), t_{j+2}) < g((v^k, v^\ell), t_{j+1}), \tag{3.5}$$

$$\Rightarrow c(\mathsf{x}^k, t_{j+1}) + \alpha_{\mathrm{w}} + c(\mathsf{x}^\ell, t_{j+2}) < c(\mathsf{x}^\ell, t_{j+1}). \tag{3.6}$$

For further insight into this condition, consider a two vertex system with current position x^k, and goal position x^ℓ. Figure 3.2 illustrates the possible paths and costs. For brevity, define $F_{k,j+1} := c(\mathsf{x}^k, t_{j+1})$, $F_{\ell,j+1} := c(\mathsf{x}^\ell, t_{j+1})$, and $F_{\ell,j+2} := c(\mathsf{x}^\ell, t_{j+2})$. Waiting is beneficial if

$$\begin{aligned} \alpha_{\mathrm{w}} + F_{k,j+1} + \alpha_{\mathrm{m}} + F_{\ell,j+2} &< \alpha_{\mathrm{m}} + F_{\ell,j+1} \\ \Rightarrow \alpha_{\mathrm{w}} + F_{k,j+1} &< F_{\ell,j+1} - F_{\ell,j+2}, \end{aligned} \tag{3.7}$$

and it is easy to see that,

$$\alpha_{\mathrm{w}} + c(\mathsf{x}^k, t_{j+1}) < -\Delta c(\mathsf{x}^k + \Delta\mathsf{x}, t_{j+1})\Delta t \tag{3.8}$$

The terms in (3.8) suggest that an optimal path can involve waiting when the sum of α_{w} and the threat in the next time step $c(\mathsf{x}^k, t_{j+1})$ is less than the difference between the field costs at the next vertex $-\Delta c(\mathsf{x}^k + \Delta\mathsf{x}, t_{j+1})\Delta t$, where $\mathsf{x}^\ell = \mathsf{x}^k + \Delta\mathsf{x}$. This occurs when the waiting costs and field values are low relative to large decreasing changes in the field threat. In other words, it is beneficial to wait if the threat along the optimal path $\mathbf{v}^*$ is rapidly diminishing and the cost to remain at current position x^k is low. This observation is supported via numerical experiments. We identify a local no-wait condition as follows.

$$\alpha_{\mathrm{w}} + c(\mathsf{x}^k, t_{j+1}) + c(\mathsf{x}^\ell, t_{j+2}) - c(\mathsf{x}^\ell, t_{j+1}) \geqslant 0 \tag{3.9}$$

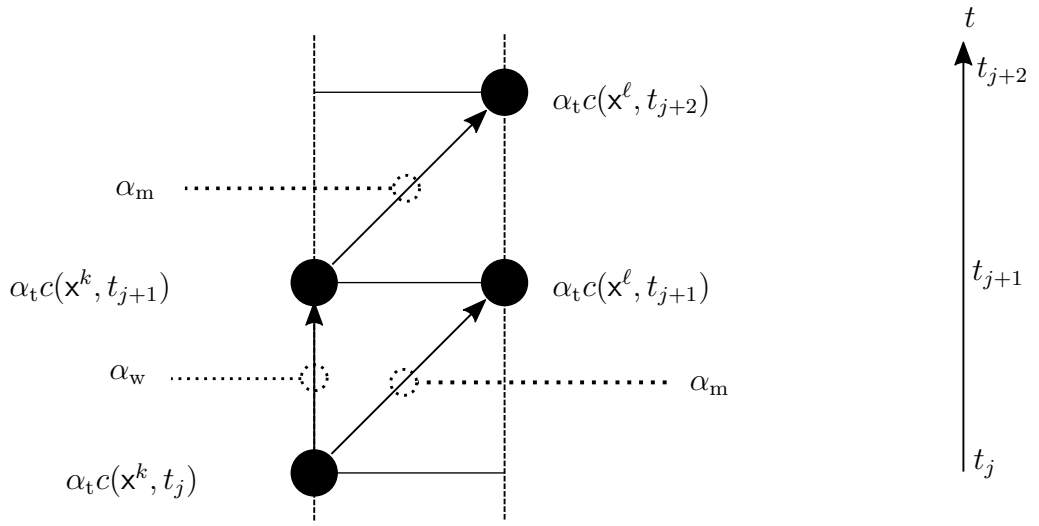


Figure 3.2: Illustration of the local no-wait condition. The decision to wait at vertex v^k or immediately move to vertex v^ℓ is based on the future threat field values $c(\mathsf{x}^k, t_{j+1})$, $c(\mathsf{x}^\ell, t_{j+1})$, $c(\mathsf{x}^\ell, t_{j+2})$, and waiting and movement costs α_{w} and α_{m}.

When executing Dijkstra's algorithm, edges can be pruned using (3.9). If (3.9) holds true, then the two-vertex local condition is violated. To decrease the number of edges explored in the graph $\mathcal{GT}$, the neighbors of v^k are evaluated under (3.9) at each iteration. If any neighbor violates the waiting condition, then v^k is marked as a non-waiting node.

3.5 Traditional Heuristics for Reducing Computational Burden

In this approach, we improved the performance by optimizing the path-planning algorithm. We applied the Manhattan distance heuristic in A* to reduce the run time of both the algorithm that considers waiting and the algorithm that ignores waiting.

Because Manhattan distance from current node to the goal node is an optimistic estimate of the total cost, it is admissible and guaranteed to find the optimal path while usually considering fewer nodes. As we discussed in Section II, the *cost* of the path $\mathbf{v}(i_{\mathrm{s}}, i_{\mathrm{g}})$ is defined by $\mathcal{J}(\mathbf{v}) := \sum_{j=1}^{P} g((v_{j-1}, v_j), jt_{\mathrm{step}})$, where $g : E \times [0, \infty) \to \mathbb{R}_{>0}$ is a strictly positive function that assigns time-varying edge transition costs. We can estimate the cost better by also adding

threat exposure cost. More specifically, Instead of using $h = d(node, goal)$, we update h with $h_{new} = d(node, goal) + minimum\_threat\_value * d(node, goal)$ where $d(node, goal)$ is the Manhattan distance from current node to the goal node.

Datasets are the same as we described in Section B.1. To assess the algorithms, we generated various but identical environments for path-planning with "Wait without heuristic", "Wait with heuristic", "No-wait without heuristic" and "No-wait with heuristic". We kept track of the number of nodes visited and the run time to assess and compare each of these algorithms.

3.6 Machine Learning Classification for Path-planning Algorithm Selection

The fundamental trade-off being explored with the waiting concept is between path optimality and computational efficiency. The waiting algorithm will provide the optimal solution at the cost of additional computational burden, and the non-waiting algorithm will be faster but provide a suboptimal solution. In Section 6.2.1, we present results of 2000 simulations that demonstrate there is some relationship between characteristics of the field (minimum/average field value, min and max field gradients, ...) and the benefits of waiting (reduction in path cost). A waiting algorithm used on some fields may give the same path solution as the non-waiting algorithm, thus there was no benefit to using the computationally expensive waiting algorithm. The question arises, "Can the appropriate algorithm be selected a priori given a field and characteristics of that field?" In other words if, for a given field, waiting will provide some benefit then use the expensive algorithm, otherwise use the cheaper algorithm, as seen in Fig. 3.3.

This strategy amounts to a classification problem in standard machine learning terminology, more specifically a supervised learning problem. In short, we can gather many examples of different threat field topologies and behaviors and in what amounts to a large complicated regression problem the machine learning process will develop a function that determines if a given field is beneficial for waiting or not. The advantage is that this machine learning problem can be done off-line, then the classification can be done on-line and provide the most appropriate algorithm

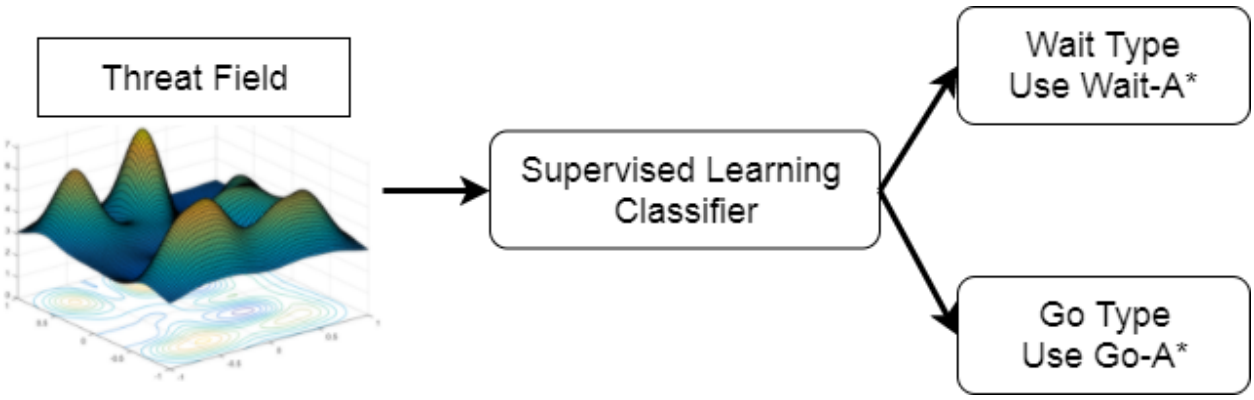


Figure 3.3: A potential outcome is the use of a classifier to select the appropriate path-planner given a particular instance of the Gaussian field. The classifier will be trained on occurrences of the field labeled as Go *or* Wait

choice to the path planning agent.

This machine learning classification endeavor began as a curiosity, and while it provide incredibly valuable from a learning and exploration perspective the results were not satisfactory. Rather than interrupt the flow of this thesis from noiseless time-varying path planning problems to uncertain time-varying environment path planning, the machine learning content is available in a supplemental chapter in Appendix B. A brief conclusion states that for this problem it is difficult to "hand-engineer" features for a traditional machine learning approach such as support vector machines (SVM), k-Nearest Neighbors (KNN), or random forests. When it is difficult to hand engineer features, deep learning techniques can automatically discern features and learn classification simultaneously. However, the deep learning approach using a CNN-LSTM (convolutional neural network with recurrent neural network LSTM layers) also failed to produce satisfactory classification. This may be due to hyperparameter optimization, however, CNN architectures are used because they are invariant to the spatial position of a feature and the path planning problem is inherently spatially variant. A different architecture which directly considers the spatial position of the path may be more appropriate. See Appendix B for full details and additional issues dealing with classification accuracy.

Chapter 4

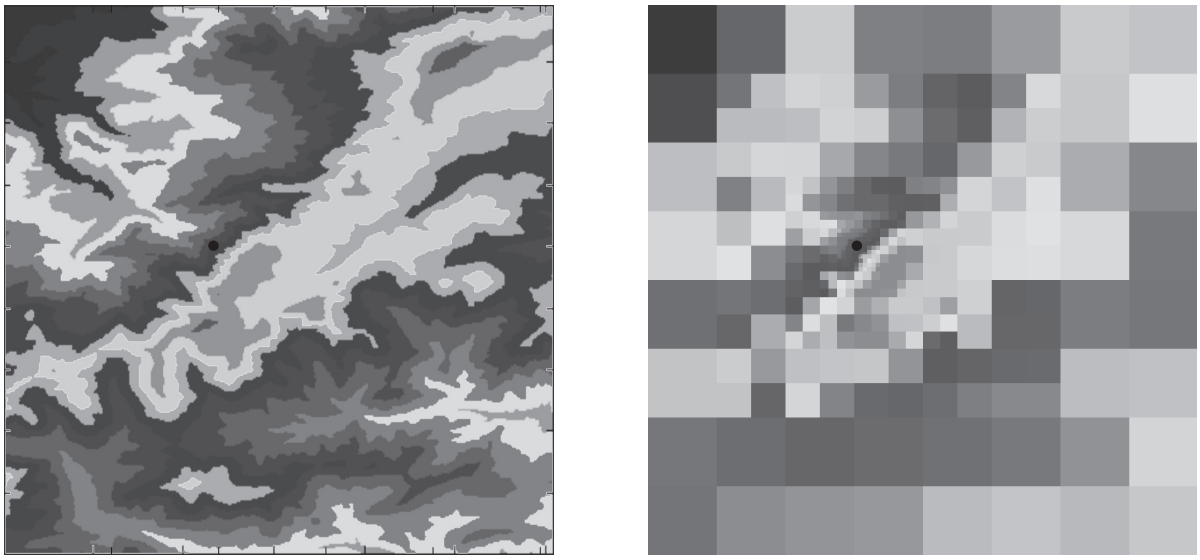
Multiresolution Path-planning with Waiting in Time-varying Spatial Fields

4.1 Problem Formulation

We consider next a multiresolution discretization of the workspace based on the discrete wavelet transform (DWT). This multiresolution discretization algorithm is adopted from the second author's previous works (Cowlagi, 2014; Cowlagi and Tsiotras, 2012b), and provides a so-called *vehicle-centric multiresolution approximation* to the threat field intensity map. The motivation for considering multiresolution discretization is twofold. First, such a discretization may be necessary for practical onboard computations. Second, such a discretization reflects the nature of typical maps available to vehicles in many applications: namely, well-known in the immediate vicinity of the vehicle, partially known in regions farther away, as illustrated in Fig. 4.1. We investigate the potential cost reductions by considering waiting in path-planning. Waiting is allowed only at vertices in the highest resolution region. We provide a skeletal overview of this discretization here, and refer the reader to (Cowlagi and Tsiotras, 2012b) for details.

The DWT represents a scalar field using so-called *approximation and detail coefficients,* which multiply spatial basis functions called *scaling functions and wavelets.* If appropriate, it is possible to use these spatial basis functions to also represent the threat field, without affecting the results presented in this paper. The reasons for using wavelets in this section are the convenient dyadic structure and orthogonality of the basis functions (Rao and Bopardikar, 1998).

Without loss of generality, we assume that $\mathcal{W} = [0,1] \times [0,1]$. For the following discussion, we choose a parameter $D \in \mathbb{Z}_+$ indicating the highest resolution considered in the multiresolution approximation. Specifically, the smallest grid separation (i.e. highest resolution) is 2^{-D}. For con-



(a) (b)

Figure 4.1: Example of an intensity map and its vehicle-centric multiresolution approximation according to Eqn. (4.6). The vehicle's location is indicated by the black dot near the center. (a) Original field map. (b) Vehicle-centric multiresolution approximation.

text, the grid separation in the uniformly high resolution map considered in the previous subsection is 2^{-7} units.

Assumption 2. The field c is sufficiently smooth such that

$$c(x, y, t_0) + \sum_{n=1}^{\infty} \frac{(t-t_0)^n}{n!} \left. \frac{\partial^n c(x,y,t)}{\partial t^n} \right|_{t=t_0} \tag{4.1}$$

converges to $c(\mathsf{x}, t)$ for all $\mathsf{x} \in \mathcal{W}$, and $t \in [0, \infty)$. □

Assumption 3. The values taken by c are known at a finite resolution $m_{\mathrm{f}} > -D$, for $k, \ell = 0, 1, \ldots, 2^{D+m_{\mathrm{f}}} - 1$. Without loss of generality, $m_{\mathrm{f}} = 0$. □

Assumption 4. The values taken by the temporal derivatives of c at time $t = t_0$ are known at the same finite resolution as described in Assumption 3. □

Relying on Assumption 2, we consider Taylor series expansions of *time-varying* approximation and detail coefficients of the DWT of the threat field c. The coefficients of these Taylor series

expansions are given by

$$\alpha^n_{m_0,k,\ell} := \left\langle \Phi_{m_0,k,\ell}(x,y),\, \frac{\partial^n c(x,y,t)}{\partial t^n}\bigg|_{t=t_0} \right\rangle, \tag{4.2}$$

$$\beta^{p,n}_{m,k,\ell} := \left\langle \Psi^p_{m,k,\ell}(x,y),\, \frac{\partial^n c(x,y,t)}{\partial t^n}\bigg|_{t=t_0} \right\rangle, \tag{4.3}$$

for $p = 1,2,3$, $k,\ell \in \mathbb{Z}$, $m_{\mathrm{f}} \geqslant m \geqslant m_0 = -D$, and $n \in \mathbb{Z}_{\geqslant 0}$. Here Φ and Ψ denote families of scaling functions and wavelets, respectively (see (Cowlagi and Tsiotras, 2012b) for details). The threat field is reconstructed from these coefficients as follows:

$$\begin{aligned}\overline{c}(x,y,t) = &\sum_{k,\ell=0}^{1} \sum_{n\in\mathbb{Z}_{\geqslant 0}} \frac{(t-t_0)^n}{n!} \alpha^n_{m_0,k,\ell} \Phi_{m_0,k,\ell}(x,y) \\ &+ \sum_{p=1}^{3} \sum_{m=m_0}^{m_{\mathrm{f}}} \sum_{k,\ell=0}^{2^{m-m_0}} \sum_{n\in\mathbb{Z}_{\geqslant 0}} \frac{(t-t_0)^n}{n!} \beta^{p,n}_{m,k,\ell} \Psi^p_{m,k,\ell}(x,y),\end{aligned} \tag{4.4}$$

where $\overline{c}$ denotes the reconstructed field. To construct a vehicle-centric multiresolution approximation of the threat field, let $\mathcal{A} \subset \{(m,k,\ell) \in \mathbb{Z}^3 : m_0 \leqslant m < 0,\ 0 \leqslant k,\ell \leqslant 2^{D+m}\}$. We define for all $p = 1,2,3$ and $n \in \mathbb{Z}_+$,

$$\hat{\beta}^{p,n}_{m,k,\ell} := \begin{cases} \beta^{p,n}_{m,k,\ell} & (m,k,\ell) \in \mathcal{A}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

The reconstruction of the threat field is then performed using $\hat{\beta}^{p,n}_{m,k,\ell}$ in Eqn. (4.4) instead of $\beta^{p,n}_{m,k,\ell}$. The set $\mathcal{A}$ contains the indices of detail coefficients that are considered "significant". Following (Cowlagi and Tsiotras, 2012b), we choose $\mathcal{A}$ such that high resolution information is retained in the immediate vicinity of the vehicle's current location $(x_0,y_0) \in \mathcal{W}$ and it is gradually discarded in regions farther away. To this end, let $\varrho : \mathbb{Z} \to \mathbb{N}$ be a "window" function that specifies, for each level of resolution, the distance from the vehicle's location up to which the detail coefficients at that level are significant. The set $\mathcal{A} = \mathcal{A}^{\mathrm{win}}(x_0,y_0)$ of indices is then defined by

$$\mathcal{A}^{\mathrm{win}}(x_0,y_0) := \big\{(m,k,\ell) : m_0 \leqslant m < 0, \tag{4.6}$$

$$\lfloor 2^m x_0 \rfloor - \varrho(m) \leqslant k \leqslant \lfloor 2^m x_0 \rfloor + \varrho(m),\ \lfloor 2^m y_0 \rfloor - \varrho(m) \leqslant \ell \leqslant \lfloor 2^m y_0 \rfloor + \varrho(m) \big\}.$$

The preceding description of the vehicle-centric multiresolution approximation is minimal; the interested reader is referred to (Cowlagi, 2014; Cowlagi and Tsiotras, 2012b) for further details. In what follows, we focus on the following problem.

Problem 3. *Solve Problem 2 using a vehicle-centric multiresolution approximation $\overline{c}$ of the threat field.*

To formulate Problem 2 precisely, we define a multiresolution cell decomposition Ω^{mr}, which is a partition of $\mathcal{W}$ into square cells of different sizes, such that $\overline{c}$ is constant in the spatial variables over each of the cells. We denote by $\mathsf{cell}(v^k;\Omega)$ the coordinates of the center of the cell associated with a vertex $v^k \in V$, and by $\mathsf{vert}(\mathcal{C};\mathcal{G})$ the vertex in V associated with a cell $\mathcal{C} \in \Omega$. We attach with the cell decomposition Ω^{mr} a graph $\bar{\mathcal{G}} = (\bar{V}, \bar{E})$ such that each cell in Ω^{mr} corresponds to a unique vertex in $\bar{V}$. Each vertex $j \in \bar{V}$ corresponds to a set $W(\bar{v}^k, \bar{V}) \subset V$, and the collection $\{W(\bar{v}^k, \bar{V})\}_{\bar{v}^k \in \bar{V}}$ is a partition of V. Specifically:

$$W(\bar{v}^k, \bar{V}) := \big\{ v^k \in V : \mathsf{cell}(v^k;\Omega) \subseteq \mathsf{cell}(\bar{v}^k;\Omega^{\mathrm{mr}}) \big\}. \tag{4.7}$$

Vertices $\bar{v}^k, \bar{v}^\ell \in \bar{V}$ are adjacent in $\bar{\mathcal{G}}$ if there exist $v^k \in W(\bar{v}^k, \bar{V})$ and $v^\ell \in W(\bar{v}^\ell, \bar{V})$ such that $\{v^k, v^\ell\} \in E$. We define edge costs $\bar{g} : \bar{E} \to \mathbb{R}_+$ in $\bar{\mathcal{G}}$ as

$$\bar{g}((\bar{v}^k, \bar{v}^\ell), t) := \begin{cases} c(\mathsf{x}^\ell, t) + \alpha_{\mathrm{w}}, & \text{if } |W(\bar{v}^k, \bar{V})| = 1, \bar{v}^k = \bar{v}^\ell, \\ (c_{\bar{v}^\ell}(\bar{v}^\ell, t) + \alpha_{\mathrm{m}})|W(\bar{v}^\ell, \bar{V})|, & \text{otherwise,} \end{cases} \tag{4.8}$$

where c_{v^ℓ} are time-averaged cell intensities. Owing to the use of the DWT, the computation of these time-averaged intensities involves simple algebraic computations (see (Cowlagi, 2014) for details). This edge cost function considers approximate periods of time and distance required to traverse these cells, which are in turn related to the cell sizes $|W(\bar{v}^\ell, \bar{V})|$. We restrict waiting to only the uniform resolution window of Ω^{mr}, precisely, when $|W(\bar{v}^k, \bar{V})| = 1$ and $\bar{v}^k = \bar{v}^\ell$ in Eqn. (4.8). The cost $\bar{\mathcal{J}}(\bar{\mathrm{v}})$ of a path in $\bar{\mathcal{G}}$ is the sum of all edge costs.

4.1.1 Solution to Multiresolution Path-planning Problem

The multiresolution path-planning algorithm utilizes Dijkstra's algorithm to search the product $\bar{\mathcal{G}}\mathcal{T}$ of graph $\bar{\mathcal{G}}$ and $\mathcal{T}$.

Figure 4.2 shows in pseudo-code form the proposed path-planning algorithm based on the vehicle-centric multiresolution approximation of the spatial field c. The algorithm iterates Lines 3–10 until the goal is reached. At each iteration, the algorithm computes a vehicle-centric multiresolution approximation and the corresponding cell decomposition graph (Lines 1–4). In Line 5, the optimal path in $\bar{\mathcal{G}}_n$ is computed by a label-correcting algorithm. The vehicle is assumed to traverse the first cell in the path $\bar{\pi}_n^*$, and the process is repeated for the new vehicle location.

A procedure to determine the locations and the sizes of cells in Ω^{mr} in the vehicle-centric multiresolution approximation, including fast updates of these cell locations and sizes with the changing vehicle location, is provided in (Cowlagi and Tsiotras, 2012b). Furthermore, a procedure denoted MR-GRAPH to determine the edges in the graph $\bar{\mathcal{G}}$ associated with Ω^{mr}, including fast updates to the sets of vertices and edges of this graph with the changing vehicle location, is also provided in (Cowlagi and Tsiotras, 2012b).

One detail not explicitly stated in the pseudo-code in Figure 4.2 is the ability to bypass the optimization problem in Line 5 when the multiresolution decomposition in Lines 1–4 is unchanged. Because of the vehicle-centric decomposition, if the vehicle has found a waiting beneficial path it will remain stationary for several iterations while it steps through the optimal space-time path, $\bar{\mathbf{v}}^*$. In this case, Line 5 can be bypassed until the vehicle moves and the decomposition updates. For the multiresolution planning, this significantly speeds up the planning and allows the planner which considers waiting to compute in the nearly the same time as the no-wait planner as will be seen in the results section.

The *topological properties* (i.e. cell locations, sizes, and adjacency relations) of the vehicle-centric multiresolution approximation are the same as in (Cowlagi and Tsiotras, 2012b) and therefore, the proof of completeness of this path-planning algorithm is the same as that provided in detail in (Cowlagi and Tsiotras, 2012b).

Multiresolution Path-planning with Time-varying Costs

procedure MR-APPROX(j)

1: $\mathcal{A} := \mathcal{A}^{\text{win}}(\mathsf{cell}(v^k; \mathcal{G}))$ using Eqn. (4.6)

procedure MAIN

1: $\mathbf{v} := v^{i_\text{s}}$, $v_0 := v^{i_\text{s}}$, $n := 0$, *AtGoal* $:= 0$, $\mathcal{J}(\mathbf{v}) := 0$
2: **while** $\neg$*AtGoal* **do**
3: $\quad \mathcal{A}_n := \text{MR-APPROX}(v_n^k)$
4: $\quad \mathcal{G}_n := \text{MR-GRAPH}(\mathcal{A}_n)$
5: $\quad \bar{\mathbf{v}}_n^* := \arg\min \left\{ \bar{\mathcal{J}}(\bar{\mathbf{v}}) : \bar{\mathbf{v}} \text{ is a path in } \bar{\mathcal{G}}_n \right\}$
6: $\quad v_{n+1}^k := \mathsf{vert}(\mathsf{cell}(\bar{v}_1^\ell; \bar{\mathcal{G}}_n); \mathcal{G})$, where $\bar{v}_1^\ell$ is the second vertex in the path $\bar{\mathbf{v}}_n^*$
7: $\quad$ *AtGoal* $:= (v_{n+1}^k = v^{i_\text{g}})$,
8: $\quad \mathbf{v} := (\mathbf{v}, v_n^k)$
9: $\quad \mathcal{J}(\mathbf{v}) := \mathcal{J}(\mathbf{v}) + g(v_n^k, v_{n+1}^k, n\delta t)$
10: $\quad n := n + 1$

Figure 4.2: Pseudo-code for the proposed path-planning algorithm.

Chapter 5

Interactive Planning and Sensing for Time-varying Systems

We address planar path-planning for a mobile vehicle, which we call the *actor* vehicle, to traverse a planar workspace $\mathcal{W}$ with minimum exposure to a spatially and temporally varying scalar field, called the *threat field.* The threat field is unknown, time-variant, and strictly positive everywhere on $\mathcal{W}$. The values taken by the threat field over $\mathcal{W}$ are estimated by a finite number of mobile sensors that take pointwise measurements. All measurements are noisy. Future applications of this problem setup include, for example, delivery (by an actor) of emergency supplies to a remote location that lies within/beyond a region afflicted by wildfire or atmospheric contaminants (the threat field).

We study the problem of sensor scheduling over a finite time horizon to optimize the actor's performance. This explicit relationship between the problem of sensor placement and the actor's path-planning problem removes the traditional separation between the sensing and planning.

To this end, we formulate this problem on a grid defined on $\mathcal{W}$, which in turn defines a topological graph $\mathcal{G}$. The threat field is assumed to be finitely parameterized by coefficients of spatial basis functions. These coefficients are time-varying and described by a set of differential equations as outlined in Section 5.3. Estimates of these parameters are obtained from the sensor measurements through a Kalman filter. Whereas edge transitions in the graph $\mathcal{G}$ are deterministic and known, the transition *costs* depend on the threat field estimates, and are deterministic but unknown. We propose an iterative sensor placement and path-planning algorithm. At each iteration, Dijkstra's algorithm is used to find a path with minimum threat exposure in the graph $\mathcal{G}$ for the actor. A set of grid points "near" this path are identified as points of interest. The next set of sensor locations is determined to improve the confidence of threat field estimates on these points of interest. The threat field estimate is accordingly updated, and the iteration repeats. The algorithm either returns

a path of sufficient confidence or reports a failure.

Our contributions in this paper are as follows. First, we formulate a problem in which an actor vehicle with access to a sensor network must traverse an unknown and time-varying environment (the *threat field*) and minimize its exposure to the threat field. We solve this problem using an iterative bidirectional interaction between planning and sensing phases by the actor and sensor network, respectively. We refer to this interaction as *interactive planning and sensing for time-varying fields*. The formulations and results of this paper complement and extend the work in (Cooper and Cowlagi, 2018) and (Cooper and Cowlagi), which studied the problem of static environments and proved convergence and bounds on optimality for the iterative algorithm.

5.1 Problem Overview

In what follows, we denote by $\mathbb{R}$ and $\mathbb{N}$ the sets of real and natural numbers, respectively; by $[N]$ the set $\{1, \dots, N\}$ for any $N \in \mathbb{N}$; and by $\mathbb{I}_{(N)}$ the identity matrix of size N.

Let $\mathcal{W} \subset \mathbb{R}^2$ be a closed square region, called the *workspace*, in which the actor and the sensors move. In this workspace, we formulate a grid consisting of N_{G} points uniformly placed in the workspace. The coordinates in a prespecified Cartesian coordinate axis system of the i^{th} grid point are denoted by x_i, for each $i \in [N_{\mathrm{G}}]$. We consider a strictly positive spatiotemporal scalar field $c : \mathcal{W} \times [0, \infty) \to \mathbb{R}_{>0}$, called the *threat field*, which represents unfavorable regions with higher intensity. The actor is assumed to traverse grid points according to a "$4-$connectivity rule." Let δ denote the distance between adjacent grid points. We neglect vehicle kinematic and dynamic constraints, while noting that such constraints can in the future be easily incorporated in the proposed grid-world problem setup (Cowlagi and Tsiotras, 2012a), and that multiresolution grids can also be considered (Cowlagi and Tsiotras, 2012b). We also assume that the actor vehicle has no uncertainties in localization or in motion on the grid: i.e., the current grid-point location of the actor is known, and the effect of moving to an adjacent grid-point is deterministic and known.

The actor's motion-planning problem is formulated as a graph search problem on a graph $\mathcal{G} = (V, E)$, where each vertex in V is uniquely associated with a grid point, and labeled by integers $1, 2, \dots N_{\mathrm{G}}$. The edge set E is the set of pairs of vertices associated with adjacent grid

points. For path-planning, we assume a compact interval $T = [t_0, t_f] \subset \mathbb{R}_+$. For planning purpose, $t := \{t_k : k\Delta t_s, k \in \mathbb{N}\}$, which indicates that a pair of adjacent vertices are traversed in Δt_s time, encoding the vehicle's speed. Edge transition costs are assigned by a scalar function $g : E \times T \to \mathbb{R}_{>0}$ defined as

$$g((i,j),t) = c(\mathsf{x}_j, t), \text{ for } i,j \in [N_G],\ (i,j) \in E. \tag{5.1}$$

A *path* in the graph $\mathcal{G}$ between two prespecified vertices i_s and i_g is a sequence $\mathbf{v} = (v_0, v_1, \ldots, v_P)$ of successively adjacent vertices with $v_0 = i_s$ and $v_P = i_g$. The *cost* $\mathcal{J}(\mathbf{v}) \in \mathbb{R}_{>0}$ of this path is the sum of edge transition costs, i.e., $\mathcal{J}(\mathbf{v}) := \sum_{k=1}^{P} g((v_{k-1}, v_k), kt_{P-step})$. The actor's motion-planning problem is the problem of finding a path with minimum cost between initial and goal grid points $i_s, i_g \in [N_G]$. We will refer to this path as the *true optimal* path, denoted by $\mathbf{v}^*$.

Note that the costs are time dependent, and a path-planning algorithm which considers time-varying edge cost is used. For time-varying environments the *true optimal* path may include instances of *waiting* (vehicle remains at the same location for multiple consecutive time steps) which increases the computational complexity (Dean, 2004b) by an order of T^2. General algorithms for path-planning with waiting to minimize time-varying edge costs appear in (Chabini, 2013; Dean, 2004b; Orda and Rom, 1991), and the trade off between optimality and complexity is discussed in (**?**) for environments similar to those in this paper. In order to focus on the sensor placement and iterative process of the proposed algorithm, we restrict planning to *non-waiting* solutions and subsequent references to path optimality are with respect to the optimal non-waiting path.

Uncertainty in the actor's motion-planning problem arises from uncertainty in the knowledge of the threat field. The spatiotemporal threat field may model some physical phenomena such as advection-diffusion of gases or radiation in the atmosphere (Demetriou et al., 2013). These various phenomena modeled as partial differential equations can be approximated as series solutions using Gaussian basis functions (Crank, 1979). Therefore, we assume that the threat field is finitely parametrized as $c(\mathsf{x}, t) = \sum_{n=1}^{N_P} \theta_n(t)\phi_n(\mathsf{x}) = \Phi(\mathsf{x})\Theta(t)$, where $\phi_n : \mathcal{W} \to \mathbb{R}$ are prespecified spatial basis functions, $\Phi(\mathsf{x}) := [\ \phi_1(\mathsf{x}) \ \ \ldots \ \ \phi_{N_P}(\mathsf{x})\ ]$, and $\Theta(t) := [\ \theta_1(t) \ \ \ldots \ \ \theta_{N_P}(t)\ ]^T$. The basis functions and the number of parameters N_P are prespecified. Specifically, we assume

$$\phi_n(\mathsf{x}) := \exp(-\frac{1}{2\nu_n} \cdot (\mathsf{x} - \bar{\mathsf{x}}_n)^T(\mathsf{x} - \bar{\mathsf{x}}_n)), \tag{5.2}$$

for each $n \in [N_{\mathrm{P}}]$, where $\nu_n \in \mathbb{R}_{>0}$ and $\bar{\mathsf{x}}_n \in \mathcal{W}$ are prespecified constants.

Definition 3 (Region of significant support)**.** The region $\mathcal{R}_n^{\sup} := \{\mathsf{x} : \|\mathsf{x} - \bar{\mathsf{x}}_n\| \leqslant 3\sqrt{\nu_n}\} \cap \mathcal{W}$ is defined as the *region of significant support* for the basis function ϕ_n.

Whereas the functions ϕ_n do not have compact support in $\mathbb{R}^2$, 99.74% of the volume enclosed under each ϕ_n is enclosed by the restriction of ϕ_n to $\mathcal{R}_n^{\sup}$.

Assumption 5. The constants ν_n and $\bar{\mathsf{x}}_n$ are chosen such that the union of the interiors of the regions of significant support cover the entire workspace.

When N_{P} is a perfect square, the workspace coverage in Assumption 5 is achieved by choosing $\bar{\mathsf{x}}_n$ on a uniform $\sqrt{N_{\mathrm{P}}} \times \sqrt{N_{\mathrm{P}}}$ grid on the workspace $\mathcal{W}$. To ensure coverage and to minimize overlap of the basis functions, ν_n is chosen such that the $\mathcal{R}_n^{\sup}$ extends to half the diagonal distance between basis functions. Specifically, for adjacent basis functions with indices n and p, we choose $\nu_n := (\frac{\sqrt{2}}{6}(\bar{\mathsf{x}}_n - \bar{\mathsf{x}}_p))^2$.

This choice of Gaussian basis functions is justified by the fact that square integrable functions can be approximated with arbitrary precision by linear combinations of Gaussian functions (Calcaterra and Boldt, 2008). The proposed algorithm does not specifically depend on this choice, and other basis functions such as orthogonal wavelets (Rao and Bopardikar, 1998) can be used.

We assume that a finite number N_{S} of sensors take pointwise measurements. These sensors are assumed to be located at grid points, and the set of these grid points is denoted by $\mathbf{s} = \{s_1, s_2, \ldots, s_{N_{\mathrm{S}}}\} \subset \{1, \ldots, N_{\mathrm{G}}\}$. The measurement taken by each sensor is $z_k := c(\mathsf{x}_{s_k}, t) + \eta_k$, where $\eta_k \sim \mathcal{N}(0, \sigma_k^2)$, for each $k = 1, \ldots, N_{\mathrm{S}}$. Finally, we assume that the number of sensors is "small," i.e., $N_{\mathrm{S}} \ll N_{\mathrm{G}}$.

We define a *region of estimability* for each basis ϕ_n, which is a region in which a sensor must be located to confidently estimate the parameter θ_n. To this end, let $\bar{\sigma} := \max\{\sigma_k\}_{k=1}^{N_{\mathrm{S}}}$. For Gaussian measurement noise, if the signal-to-noise ratio (SNR) is greater than three, $\frac{\theta_n \phi_n}{\bar{\sigma}} \geqslant 3$, the probability of detecting a signal is at least 0.9974 (Miller et al., 2005). Therefore, placing sensors outside of the region where $\theta_n \phi_n \geqslant 3\bar{\sigma}$ is not advisable[1]. By (5.2), it follows that this region is defined by the ball $\left\{\mathsf{x} \in \mathcal{W} : \|\mathsf{x} - \bar{\mathsf{x}}_n\| \leqslant \sqrt{2\nu_n \log(\theta_n / 3\bar{\sigma})}\right\}$.

[1]Note that the maximum SNR is obtained when the sensor is placed at $\mathsf{x} = \bar{\mathsf{x}}_n$.

Whereas the parameter θ_n is unknown, the radius of this ball scales logarithmically[2] with θ_n. Therefore, this region can be determined using a rough order of magnitude estimate of θ_n, which is typically available in practice. We assume that upper and lower bounds $\bar{\theta}_n$ and $\underline{\theta}_n$ are known, and that these bounds are within two orders of magnitude of each other.

It is always possible to place a sensor at a grid point inside $\mathcal{R}^{\mathrm{est}}$ if the distance between grid points $\delta < 2\sqrt{2\nu_n \log(\bar{\theta}_n/3\bar{\sigma})}$. We tighten this region further to ensure that the regions of estimability of adjacent basis functions do not significantly overlap. By doing so, estimates of parameters made from measurements taken within this region are independent from estimates of other parameters. That is, if z_n, z_p are measurements taken by sensors placed in $\mathcal{R}_n^{\mathrm{est}}$ and $\mathcal{R}_p^{\mathrm{est}}$, respectively, with $n \neq p$ and $n, p \in [N_{\mathrm{P}}]$, then $\mathrm{Covar}(\hat{\theta}_n, \hat{\theta}_p | z_n, z_p) \approx 0$. It is always possible to place a sensor at a grid point inside $\mathcal{R}_n^{\mathrm{est}}$ if $\delta < 2(1 - \frac{1}{\sqrt{2}})\Delta\bar{\mathsf{x}}$, where $\Delta\bar{\mathsf{x}} := \min\{\|\bar{\mathsf{x}}_n - \bar{\mathsf{x}}_p\| : n, p \in [N_{\mathrm{P}}]\}$.

Definition 4 (Region of estimability)**.** The *region of estimability* for the basis function ϕ_n is the set

$$\mathcal{R}_n^{\mathrm{est}} := \left\{ \mathsf{x} : \|\mathsf{x} - \bar{\mathsf{x}}_n\| \leqslant (1 - 1/\sqrt{2})\Delta\bar{\mathsf{x}} \right\}.$$

The actor can avail of estimates of the threat field parameters generated using measurements taken by the sensors over a sequence of placements $\{\mathbf{s}_\ell\}_{\ell \in \mathbb{N}}$. Therefore, the actor's path-planning problem is reformulated by considering deterministic but unknown edge transition costs based on the *estimated* threat field.

Problem 4. *Find a sequence of sensor placements $\{\mathbf{s}_\ell\}_{\ell \in \mathbb{N}}$, and a path $\hat{\mathbf{v}}^*$ in $\mathcal{G}$ with minimum estimated cost.*

This problem involves an explicit dependence between the sensor locations and the actor's motion-planning problem.

[2]To be precise, the radius scales slower: i.e., with the square root of the logarithm of θ_n.

5.2 Estimation Formulation using Kalman Filter

In the static environment case, the IPAS algorithm continually updates the parameter estimates using a recursive least squares formulation. In order to extend IPAS to time-varying fields, it is straightforward to now estimate the parameters using a Kalman filter by incorporating a predictor step. We formulate the filter as a discrete Kalman filter. The input vector in the system model in Eq. 5.13 is assumed zero. The filter is initialized with $\hat{\Theta}_0 := \mathbf{0}$, and $P_0 := \lambda_0 \mathbb{I}_{(N_\mathrm{P})}$, where λ_0 is an arbitrary large constant.

The prediction or time update step is given by the discrete equations,

$$\Theta_k^- = A\Theta_{k-1} \tag{5.3}$$

$$P_k^- = AP_{k-1}A^\mathrm{T} + Q. \tag{5.4}$$

A modification to the measurement update step equations ensures each estimate update is a well-posed problem. The algorithm also maintains a set of indices $\mathcal{L} = \{m_1, \ldots, m_{N_\mathrm{L}}\} \subset [N_\mathrm{P}]$. The sensor reconfiguration policy determines $\mathcal{L}$ as discussed in Section 5.3.3. The size of this set $\mathcal{L}$ is $N_\mathrm{L} \leqslant N_\mathrm{S}$. Informally, these indices are such that the k^th sensor is placed in the region of estimability $\mathcal{R}^\mathrm{est}_{m_k}$ of the m_k^th basis function. From a practical perspective, the placement $\mathbf{s}_\ell$ is interpreted as *new* locations for sensors to be placed, and that N_L out of N_S sensors move at each iteration. The matrix $T \in \mathbb{R}^{N_\mathrm{L} \times N_\mathrm{P}}$ is defined as

$$T_{ij} := \begin{cases} 1 & \text{if } j = m_k, \\ 0 & \text{otherwise.} \end{cases}$$

for $k \in [N_\mathrm{L}], j \in [N_\mathrm{P}]$.

The observation matrix H_wp is constructed with respect to the number of measurements used and the basis ϕ_{m_k} such that H_wp has full rank. Then the parameter estimates and error covariance can be computed with the standard recursive equations modified with the transformation matrix T

at time t_k:

$$P_{\mathrm{wp}}^{-} := T P_k^{-} T^{\mathrm{T}} \tag{5.5}$$

$$G_k := P_{\mathrm{wp}}^{-} H_{\mathrm{wp}}^{\mathrm{T}} (H_{\mathrm{wp}} P_{\mathrm{wp}}^{-} H_{\mathrm{wp}}^{\mathrm{T}} + R)^{-1}, \tag{5.6}$$

$$\hat{\Theta}_k := \hat{\Theta}_k^{-} + T^{\mathrm{T}} G_k (\mathbf{z}_k - H_{\mathrm{wp}} T \hat{\Theta}_k^{-}), \tag{5.7}$$

$$P_k := (\mathbb{I}_{(N_{\mathrm{P}})} - T^{\mathrm{T}} G_k H_{\mathrm{wp}} T) P_k^{-}. \tag{5.8}$$

5.3 Interactive Planning and Sensing for Time-varying Systems Algorithm

The problem of sensor placement to optimize an actor's performance was studied for a static environment (Cooper and Cowlagi, 2018). The proposed solution in that work, *Interactive Planning and Sensing* (IPAS), determined an optimal path through an iterative sensor placement, estimation, and planning loop. The IPAS algorithm was shown to always converge upon a solution as well as ensure optimal performance to an arbitrary small value (Cooper and Cowlagi).

The IPAS algorithm is extended to environments which vary both spatially and temporally. In turn, we now refer to the algorithm as *Interactive Planning and Sensing for Time-varying fields* (IPAST). Consider the linear system model with additive Gaussian noise:

$$\dot{\Theta}(t) = A^c \Theta(t) + w \tag{5.9}$$

where $w \sim \mathcal{N}(0, Q^c)$ is the white noise process of the continuous system where $Q^c = \sigma_p \mathbb{I}_{(N_P)}$. A^c is the transition matrix that describes the evolution of the threat parameters Θ. The construction of A^c is based on the assumption that the threat field is governed by a partial differential equation with a series solution composed of Gaussian basis functions as mentioned in 5.1. For example, consider that the field is governed by the heat diffusion equation

$$\frac{\partial c}{\partial t} = \alpha \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) \tag{5.10}$$

which has a series solution as a sum of Gaussian functions,

$$c(\mathsf{x},t) = \sum_{n=1}^{N_{\mathrm{P}}} \theta_n(t)\phi_n(\mathsf{x}) = \Phi(\mathsf{x})\Theta(t). \tag{5.11}$$

The parameters Θ are the unknown quantities, therefore we are interested in tracking the evolution of these *threat parameters* through some differential equation $\dot{\Theta}(t) = f(\Theta(t)), t \in [0,\infty)$. We can relate the rate of change in the field approximation to the rate of change of the parameter:

$$\frac{\partial c}{\partial t} = \Phi(x,y)\dot{\Theta}(t). \tag{5.12}$$

Along with the two spatial derivatives, the threat field equation, Eq. 7.2, can be arranged to give

$$\dot{\Theta}(t) = \alpha\frac{\Phi^T}{|\Phi|^2}\nabla^2\Phi\Theta(t). \tag{5.13}$$

The ∇^2 is the Laplace operator which arises to describe the spatial derivatives of the Gaussian bases Φ.

Remark. This particular representation of the threat parameter evolution $\dot{\Theta}(t)$ uses a partial differential equation, specifically the diffusion equation, as a model of the physical process under investigation. Therefore, it is necessary to characterize the boundary and initial conditions of this process. While the planning and sensing spatial domain is defined on the workspace $\mathcal{W} = [-1,1]^2 \subset \mathbb{R}^2$, the physical process is defined through and beyond the boundary of $\mathcal{W}$. The spatial domain Ω of the physical process is defined on some multiple of the planning and sensing workspace, $\Omega = \ell_\Omega\mathcal{W} = [-\ell_\Omega, \ell_\Omega] \subset \mathbb{R}^2$. The physical process has the Dirichlet boundary conditions $c(t,-\ell_\Omega,y) = c(t,\ell_\Omega,y) = c(t,x,-\ell_\Omega) = c(t,x,\ell_\Omega) = c_{bd},\ x,y \in [-\ell_\Omega,\ell_\Omega]$ where c_{bd} is some small number. The initial condition is $c(0,x,y) = c_0(x,y) \in \Omega$ for $x,y \in [-\ell_\Omega,\ell_\Omega]$, where c_0 is the distribution of the threat or physical process at time $t = 0$. With regard to regularity, the smoothing associated with the diffusion equation ensures the well-posedness of the problem.

However, the dynamics of the threat parameter may be modeled in a number of ways. For example, data may be collected about some phenomenon such as wind speed and direction, and

the representative model is simply a curve fitting to the available data. In such a case, the choice of model (polynomial, local basis functions, splines, wavelets) determines the smoothness and properties such as differentiability of the threat process of interest. There is no inherent requirement on the smoothness, continuity or other 'well-behaved' properties for the IPAS approach, however properties of the threat model may dictate the type of estimation scheme implemented.

5.3.1 Discretization of threat parameter system

The target implementation is discrete Kalman filter running on an on-board microprocessor, therefore the system must first be discretized. The Kalman filter time update runs every Δt_{kf} seconds. We follow the procedure outlined in (Xie et al., 2007)(Sec. 2.4) for discretization of a continuous stochastic system. Given the continuous state transition matrix A^c and noise covariance matrix Q^c, the infinite series expansion of the sampled matrices are

$$A = \mathbb{I}_{(N_{\mathrm{P}})} + A^c \Delta t_{\mathrm{kf}} + \frac{(A^c)^2 (\Delta t_{\mathrm{kf}})^2}{2!} + ... \tag{5.14}$$

$$Q = Q^c \Delta t_{\mathrm{kf}} + \frac{(A^c Q + Q^c A^{\mathrm{T}})(\Delta t_{\mathrm{kf}})^2}{2!} + ... \tag{5.15}$$

We consider the case that Δt_{kf} is sufficiently small that terms with $(\Delta t_{\mathrm{kf}})^2$ are disregard.

The resulting discretized dynamics are now written as

$$\Theta_{k+1} = A\Theta_k + w_k \tag{5.16}$$

where $w_k \sim \mathcal{N}(0, Q)$ is a zero-mean process noise, and A and Q are the matrices from Eqns. (5.14) and (5.15) where terms of order $(\Delta t_{\mathrm{kf}})^2$ are disregard.

5.3.2 The IPAST Algorithm

The interactive planning and sensing for time-varying fields (IPAST) algorithm is described in Fig. 5.2 and the sensor reconfiguration policy is described in Fig. 5.3. The sequence and relative scaling of the phases of the IPAST algorithm are also illustrated as a time-line in Fig. 5.1.

The IPAST algorithm is initialized in Lines 1–4 with an arbitrary sensor placement $\mathbf{s}_0$, a zero vector of estimates $\hat{\Theta}_0 := \mathbf{0}$, a diagonal error covariance matrix with arbitrary large elements $P_0 := \lambda_0 \mathbb{I}_{(N_\mathrm{P})}$, and a reconfiguration counter $\ell := 0$. In addition, four flags are initialized that indicate different phases of the IPAST algorithm. The $SRflag$ indicates a new set of sensor locations should be obtained from the sensor reconfiguration procedure, while the $PlanFlag$ indicates a new path should be obtained. The $PlanFinal$ and $PlanFail$ flags describe the final resolution of the algorithm.

A Kalman filter loop drives the entire process. Upon entering the for loop, in Lines 6–15, the algorithm determines if the path should be updated and what the next phase of the algorithm will be. The planning phase is entered if the planning flag $PlanFlag$ has been set (in Line 6 of sensor reconfiguration procedure), plan failure hasn't occurred, and the current time is at least the reconfiguration time plus time to gather some measurements, $t_k \geqslant t_\mathrm{rd} + \Delta t_\mathrm{c}$. The time t_rd is the time at which sensor reconfiguration is done, and Δt_c is a window of time for measurements that allows the Kalman filter to converge on an estimate.

In Line 7, the starting time for the search window is determined as the time at which the planning will be concluded, the current time plus the time to compute a path solution, $t_\mathrm{start} := t_\mathrm{pd} = t_k + \Delta t_\mathrm{p}$. The path planning algorithm is a version of Dijkstra's algorithm which considers time-varying cost functions with a prescribed time window (see (Akgüna et al., 2007; Cai et al., 1997; Orda and Rom, 1991; Philpott and Mees, 1993) for further details and implications.) The path planning algorithm uses the current estimate of the evolution of the threat field as well as the remaining prescribed time window $t_\mathrm{rem} = [t_\mathrm{pd}, t_f]$ and returns a path or reports failure.

As the last part of the planning phase, if the path planner has not reported a failure, the next stage of the algorithm will be determined. The conditional statement in Line 10 compares the variance of the current estimated path cost, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$, with the a stop threshold, $\varepsilon_1(\mathbf{v}_\ell^*)$, based on the lower bound for the estimated path cost of the current path described in detail in Section 2.5. If the stopping condition is not met, the algorithm will continue repositioning sensors by setting the sensor reconfiguration flag. If the condition is met, the $PlanFinal$ flag is set to true. Informally, the stopping condition characterizes the confidence in the current path.

Next, at Line 16, the algorithm determines if a sensor reconfiguration is needed. If the sensor

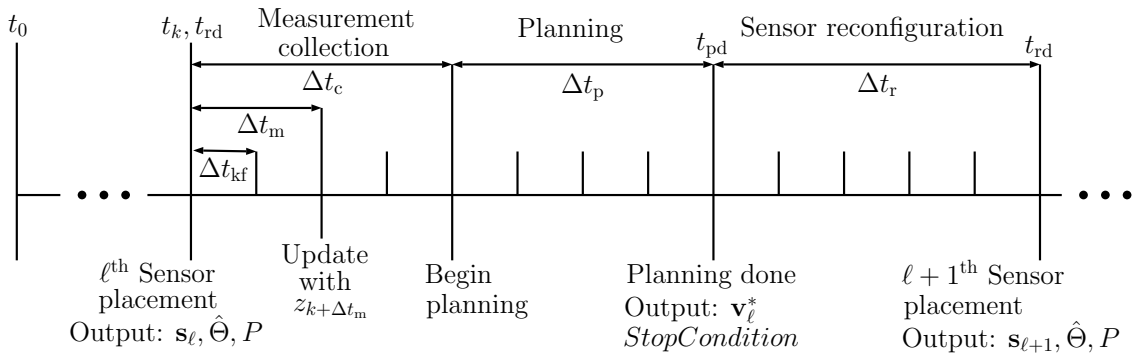


Figure 5.1: Illustration of the various time steps in the IPAST algorithm.

reconfiguration flag $SRflag$ has been set, plan failure hasn't occurred, and the current time is at least the time in which planning is done, $t_k \geqslant t_{\rm pd}$, then the SENSOR RECONFIGURATION procedure is called. The time $t_{\rm pd}$ represents the time at with we expect planning is done and the latest path is available.

After the sensor reconfiguration phase, Lines 19–21 follow the standard continuous-discrete Kalman filter formulation, see (Xie et al., 2007). The measurement update is trigged with respect to the notion of *data availability*. Data can be collected with a measurement frequency, $\frac{1}{\Delta t_{\rm m}}$ indicated by the modulus operation $\mod (t_k, \Delta t_{\rm m}) = 0$, and is only collected after a sensor reconfiguration has been completed, indicated by $t_k \geqslant t_{\rm rd}$. The other modification embedded in the measurement update is a procedure to ensure the well-posedness of the estimation problem as discussed in Section 5.2. The last instruction in Line 23 is triggered if a path has successfully met the stopping condition from Line 10, in which case at each time step of the for loop the appropriate step of the path is executed until the goal is reached.

Note that the pseudocode in Fig. 5.2 represents a compressed description of the algorithm used in simulation. Specifically, extra "book keeping" flags are needed to handle the statements which occur after the intervals in which planning or sensor reconfiguration occur. In Fig. 5.2 this is simplified with the conditionals when $t_k \geqslant t_{\rm pd}$ in Line 9 and when $t_k \geqslant t_{\rm rd}$ in Line 18. A detailed algorithm including all extra book keeping flags can be found in Appendix C.

Interactive Planning and Sensing, Time-varying

1: Set initial sensor placement $\mathbf{s}_0 \subset \{1, \ldots, N_{\mathrm{G}}\}$.
2: $\hat{\Theta}_0 := \mathbf{0},\ P_0 := \lambda_0 \mathbb{I}_{(N_{\mathrm{P}})},\ \ell := 1,\ t_{\mathrm{rd}} = 0$
3: Set $SRflag :=$ false, $PlanFlag :=$ true.
4: Set $PlanFinal :=$ false, $PlanFail :=$ false.
5: **for** $t_k = t_0$ **to** t_f **do**
6: **if** $PlanFlag$ **and** $\neg PlanFail$ **and** $t_k \geqslant t_{\mathrm{rd}} + \Delta t_{\mathrm{c}}$ **then**
7: $t_{\mathrm{pd}} = t_k + \Delta t_{\mathrm{p}}$
8: $\{\mathbf{v}_\ell^*, PlanFail\} \leftarrow$ Dijkstra's algorithm.
9: **if** $\neg PlanFail$, when $t_k \geqslant t_{\mathrm{pd}}$ **then**
10: **if** $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] > \varepsilon_1(\mathbf{v}_\ell^*)$ **then**
11: $SRflag :=$ true
12: **else**
13: $PlanFinal :=$ true
14: **else**
15: **return** failure.
16: **if** $SRflag$ **and** $\neg PlanFail$ **and** $t_k \geqslant t_{\mathrm{pd}}$ **then**
17: SENSOR RECONFIGURATION
18: Set $\ell := \ell + 1$ when $t_k \geqslant t_{\mathrm{rd}}$.
19: Predict:$\{\hat{\Theta}_k^-, P_k^-\}$
20: **if** $t_k \geqslant t_{\mathrm{rd}}$ **and** $\mod(t_k, \Delta t_{\mathrm{m}}) = 0$ **then**
21: Meas. Update: $\{\hat{\Theta}_k, P_k\} \leftarrow$ by (5.5)–(5.8).
22: **if** $PlanFinal$ **then**
23: Execute $\mathbf{v}_\ell^*$ for time step t_k.

Figure 5.2: Pseudocode for Kalman filter based IPAST algorithm to solve Problem 4.

Sensor Reconfiguration

1: Find the set $\mathcal{K}$ by Eqn. (5.17).
2: Sort $\mathcal{K}$ by largest $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$ for $n \in \mathcal{K}$. Eqn. (5.19)
3: Find the set $\mathcal{L}$ according to Eqn. (5.18).
4: For each $m \in \mathcal{L}$, place a sensor in $\mathcal{R}_m^{\mathrm{est}}$ to get $\mathbf{s}_{\ell+1}$.
5: $t_{\mathrm{rd}} = t_k + \Delta t_{\mathrm{r}}$
6: Set $PlanFlag :=$ true.

Figure 5.3: Pseudocode for SENSOR RECONFIGURATION *procedure called at Line 17 of main algorithm in Fig 5.2.*

5.3.3 Task-Driven Sensor Reconfiguration

Central to the novelty of *interactive planning and sensing* is the identification of regions to place sensors which are relevant to the path-planning problem. The sensor reconfiguration procedure, Fig. 5.3, of the proposed IPAST algorithm is designed to collect "task"-relevant measurements. This procedure first identifies (Line 1) a minimal set of indices $\mathcal{K} \subseteq [N_{\mathrm{P}}]$ such that the current path lies within the total region of significant support defined by the corresponding basis functions. To be precise, let $\mathcal{V}_\ell \subset \mathcal{W}$ be the set of grid point locations in the workspace associated with each of the vertices in the path $\mathbf{v}_\ell^*$. Then

$$\mathcal{K} := \{p \in [N_{\mathrm{P}}] : \mathcal{V}_\ell \cap \mathcal{R}_p^{\sup} \neq \varnothing\}. \tag{5.17}$$

We call the set $\mathcal{K}$ the *minimal cover* of the path $\mathbf{v}_\ell^*$.

Next, in Line 3, we compute a set of indices $\mathcal{L}$ defined as

$$\mathcal{L} := \{\mathcal{K}_{sorted} : |\mathcal{L}| = N_{\mathrm{S}}\}. \tag{5.18}$$

In Line 2, the set $\mathcal{K}$ is sorted by descending $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$, for each basis $\phi_n \in \mathcal{K}$. The value $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$ can be thought of as the influence of the threat n ,corresponding to threat coefficient θ_n and basis ϕ_n, on the variance of the estimated path cost of the current path $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$, or more simply, the *threat influence*. The threat influence of threat n is calculated as

$$\begin{aligned}\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n} &:= \textstyle\sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i})\big|_{\phi_n} P_\ell \big|_{\phi_n} \Phi(\mathsf{x}_{v_i})\big|_{\phi_n} \\ &= \textstyle\sum_{i=0}^{P} Var[\hat{\theta}_n]\phi_n^2(\mathsf{x}_{v_i}).\end{aligned} \tag{5.19}$$

Then, in Line 3, the first N_{S} bases of $\mathcal{K}_{sorted}$, with largest variance, are assigned to set $\mathcal{L}$. The sensor placement $\mathbf{s}_{\ell+1}$ in Line 4 consists of locations within the region of estimability of basis function ϕ_m, for each $m \in \mathcal{L}$. Ideally, sensors should be placed at grid points closest to the locations $\bar{\mathsf{x}}_m$ for each $m \in \mathcal{L}$, i.e., the maxima of the basis functions ϕ_m.

Remark 3 (Task-driven identification)**.** This *threat influence* metric $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$ explicitly con-

siders the mission task of finding an optimal path by direct inclusion of $\mathbf{v}_\ell^*$ in the calculation of Eqn. (5.19). Consider the scenario in which the set $\mathcal{K}$ contains two basis, ϕ_a and ϕ_b, and that the variance of corresponding coefficient estimates is such that $Var[\hat{\theta}_a] > Var[\hat{\theta}_b]$. The initial intuition is to prioritize ϕ_a for measurement by relocating a sensor to the estimable region $\mathcal{R}_a^{\text{est}}$. However, the mean location $\bar{\mathsf{x}}_a$ of threat basis ϕ_a may be collectively farther from the path than threat basis ϕ_b, i.e. $\sum_{i=0}^{P} \|\mathsf{x}_{v_i} - \bar{\mathsf{x}}_a\| > \sum_{i=0}^{P} \|\mathsf{x}_{v_i} - \bar{\mathsf{x}}_b\|$. The result may be that measuring at basis ϕ_a may have little effect on reducing $\text{Var}[\hat{J}(\mathbf{v}_\ell^*)]$.

Remark 3 emphasizes the importance of the *task-driven* approach which is core to interactive planning and sensing. First, the identification of bases for set $\mathcal{K}$ reduces the space of bases to only those relevant to the planning problem. Then, sorting of basis by threat influence metric $\text{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$ prioritizes the basis which have a more dominant effect on the variance of the estimated path cost $\text{Var}[\hat{J}(\mathbf{v}_\ell^*)]$. Note that we could calculate $\text{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$ for every $n \in N_{\text{P}}$ and achieve the same result. However, if N_{P} is very large, then $|\mathcal{K}| \ll N_{\text{P}}$, and calculating $\text{Var}[\hat{J}(\mathbf{v}_\ell^*)]\big|_{\phi_n}$ for every $n \in N_{\text{P}}$ would be computationally expensive and unnecessary.

5.4 Termination and Convergence Conditions

The termination criterion for the IPAST algorithm given in Line 10 is based on the variance of the estimated path cost which is derived from the parameter estimation error covariance, as in Eqn. (5.20). Because the path cost depends linearly on the parameters, it is a Gaussian r.v. and retains the same properties as the parameter estimates including unbiasedness. Note that the sensor reconfiguration policy *ensures* that parameters with indices in $\mathcal{K}$ are measured as the algorithm iterates.

After the ℓ^{th} reconfiguration and planning phase of the IPAST algorithm, the variance of the estimated cost of path $\mathbf{v}_\ell^* = (v_0, \ldots, v_P)$ is

$$\begin{aligned} \text{Var}[\hat{J}(\mathbf{v}_\ell^*)] &= \text{Var}[\textstyle\sum_{i=0}^{P} \Phi^{\text{T}}(\mathsf{x}_{v_i}) \hat{\Theta}_\ell] \\ &= \textstyle\sum_{i=0}^{P} \Phi^{\text{T}}(\mathsf{x}_{v_i}) P_\ell \Phi(\mathsf{x}_{v_i}). \end{aligned} \tag{5.20}$$

The termination condition, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] > \varepsilon_1(\mathbf{v}_\ell^*)$, in Line 10 will confirm the current path exceeds the prescribed level of confidence as given by $\varepsilon_1(\mathbf{v}_\ell^*)$. Here, $\varepsilon_1(\mathbf{v}_\ell^*)$ is a path-dependent threshold. chosen based on the grid resolution. Consider the hypothetical ideal case where a sensor can be placed at every grid point, and the measurement error variance of each sensor is $\bar{\sigma}$. Then the estimation error covariance is $P_{\mathrm{grid}} := (H_{\mathrm{grid}}^{\mathrm{T}} R^{-1} H_{\mathrm{grid}})^{-1}$, where

$$H_{\mathrm{grid}} := \begin{bmatrix} \Phi(\mathsf{x}_1) & \Phi(\mathsf{x}_2) & \dots & \Phi(\mathsf{x}_{N_{\mathrm{G}}}) \end{bmatrix}^{\mathrm{T}},$$

$R := \bar{\sigma}\mathbb{I}_{(N_{\mathrm{G}})}$. Then we define

$$\varepsilon_1(\mathbf{v}_\ell^*) := \lambda_1^2 \textstyle\sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i})(P_{\mathrm{grid}}/N_{\mathrm{P}})\Phi(\mathsf{x}_{v_i}), \tag{5.21}$$

Note that P_{grid} can be computed a priori, i.e., without taking any measurements at all. Additionally, scaling P_{grid} by N_{P} gives a consistent threshold $\varepsilon_1(\mathbf{v}^*)$ for environments of increasing N_{P}.

The following proposition identifies the conditions under which the proposed IPAST algorithm will terminate in a finite number of iterations.

Proposition 2. *The IPAST algorithm terminates in a finite number of iterations $\bar{\ell}$ if there is sufficient time and the following inequality holds:*

$$\begin{aligned} &\underbrace{\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)|(P_k - P_{k-n_R})]}_{\textit{change of } \mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] \textit{ in reconfiguration phase}} \\ &\quad - \underbrace{\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)|T^{\mathrm{T}} G_k H_{\mathrm{wp}} T P_k^-]}_{\textit{Reduction due to measurements}} \\ &+ \underbrace{\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)|(P_{k+n_m})_{n_P} - P_{k+n_m})]}_{\textit{change during planning phase}} < 0 \end{aligned} \tag{5.22}$$

where,

$$\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)|X] = \sum_{i=0}^{P} \Phi^{\mathrm{T}}(\mathsf{x}_{v_i}) X \Phi(\mathsf{x}_{v_i}) \tag{5.23}$$

where $(P_k)_n$ is the discrete error covariance recurrence relation after n updates,

$$(P_k)_n = A^n P_k (A^{\mathrm{T}})^n + \sum_{i=0}^{n-1} A^i Q (A^{\mathrm{T}})^i, \tag{5.24}$$

and n is the number of updates that occur during a particular phase, i.e. $n = \frac{\Delta t_{\mathrm{p}}}{\Delta t_{\mathrm{kf}}}$ for the duration of planning.

In Proposition 2, the integer n_R is the number of updates during a reconfiguration phase, i.e. $n_R = \frac{\Delta t_{\mathrm{r}}}{\Delta t_{\mathrm{kf}}}$. Therefore, P_{k-n_R} is the error covariance matrix n_R updates before the time index k. Then, $(P_k - P_{k-n_R})$ is the growth of the error covariance since time index $k - n_R$, in other words, the amount of error covariance that is accumulated over a reconfiguration sequence. The term $\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)|T^{\mathrm{T}} G_k H_{\mathrm{wp}} T P_k]$ represents the amount of reduction in the path cost variance due to measurements from the current set of sensors. The term P_{k+n_m} is the state of the error covariance after the measurement updates at time index $k + n_m$ where n_m is the number of updates due to measurements. The term $(P_{k+n_m})_{n_P}$ represents the state of the error covariance after the planning phase is completed, and $n_P = \frac{\Delta t_{\mathrm{p}}}{\Delta t_{\mathrm{kf}}}$ is the number of update cycles during planning phase. Therefore, the difference $((P_{k+n_m})_{n_P} - P_{k+n_m})$ represents the amount of variance accumulated while calculating the path.

Note that $\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)]$ is evaluated against $\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)$ immediately after the planning complete. Therefore, stated simply, if the growth of the error variance over the previous planning and reconfiguration phase, minus the affect of the sensors, plus the additional growth while computing a path is less than zero, eventually $\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)]$ will decrease below the stopping threshold $\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)$.

Finally, note that all path cost variance evaluations are made with respect to the final accepted path $\mathbf{v}_{\bar{\ell}}^*$. We are interested in the path that is both the output from the path planner, e.g. a minimum threat path determined by Dijkstra's algorithm, and has a path cost variance $\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)]$ below the specified threshold $\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)$. Therefore, the quantity of interest is the behavior of $\mathrm{Var}[\hat{J}(\mathbf{v}_{\bar{\ell}}^*)]$, the variance of the final accepted path.

Proof. The argument for the convergence proposition rests on the notion that the growth of the error covariance cannot be faster than the ability of the sensor reconfiguration procedure to target

and reduce the parameter estimate variances associated with the path.

To begin, consider that the number of parameters N_{P} is finite and there at least $N_{\mathrm{S}} \geqslant 1$ sensors available. Consider the moment immediately after a sensor reconfiguration has taken place at time index k as the reference time index. Then P_k is the state of the error covariance after reconfiguration is complete, but before measurements are taken. Then consider that P_{k-n_R} is the state of the error covariance at the beginning of the sensor reconfiguration. The matrix $(P_k - P_{k-n_R})$ represents the change in the error covariance due to Eq. 5.4. Therefore, the first term in Eq. 5.22 is the change in the path cost variance during the reconfiguration phase.

The second term of the inequality describes the reduction of the variance of the estimated path cost due to the measurements from sensors assigned to the basis set $\mathcal{L}$. Inspecting the error covariance update formula in Eqn. (5.8), the term that modifies the variance of the estimates is $T^{\mathrm{T}} G_k H_{\mathrm{wp}} T P_k^-$. Therefore, by considering the matrix $T^{\mathrm{T}} G_k H_{\mathrm{wp}} T P_k^-$, we can quantify the reduction in the estimated path cost variance due to the measurements.

Lastly, because the stop condition, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] < \varepsilon_1(\mathbf{v}_\ell^*)$, cannot be evaluated until the ℓ^{th} path is available, we also must consider the growth of the error covariance over the planning duration Δt_P. In the third term of inequality 5.22, P_{k+n_m} is the error covariance after measurements have been assimilated, and $(P_{k+n_m})_{n_P}$ is the state of the error covariance after n_P update cycles, i.e. during the planning phase. Therefore, the third term $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)|(P_{k+n_m})_{n_P} - P_{k+n_m})]$ represents the change in the estimated path cost variance while the next optimal path is computed.

Therefore, if the L.H.S. of the inequality Eqn. (5.22) which represents the total change of $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ is negative then after a finite number of iterations, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ reduces below $\varepsilon_1(\mathbf{v}_\ell^*)$. □

5.5 Steady State Conditions and Extra considerations

Remark 4 (Steady state covariance)**.** The time-varying system as described in Section 5.1 is derived from the advection-diffusion equation with no sources. Therefore, the system is inherently stable and the solution of the discrete algebraic Ricatti equation will yield a unique steady state

covariance for the error covariance matrix of the threat parameters, P_∞. In the extreme case where the planning and reconfiguration phases are much slower than the system dynamics, the implications of equation 5.22 in Proposition 2 are that the reduction in $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ by the measurements need to be large enough to reduce the path cost variance given by the steady state covariance, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)|P_\infty]$, below the stop threshold $\varepsilon_1(\mathbf{v}_\ell^*)$.

Remark 5 (Time Window)**.** The termination criterion in Line 10 of the IPAST algorithm as well as the convergence condition of Proposition 2 assume a valid path $\mathbf{v}^*$ is available. For a finite horizon mission window, if a planning phase begins too late there will be insufficient time in the remaining window. This is the case when Dijkstra's algorithm returns a `true` $PlanFail$ flag in Line 8. Therefore, for Remark 4 to hold, the time window must be sufficiently long to ensure both an available path $\mathbf{v}^*$ and for the error covariance to evolve to P_∞.

In the preceding proposition, several factors can determine if the inequality holds. If the total time for reconfiguration Δt_r is increased, this allows for a great accumulation of variance that needs to be overcome by the next set of sensors. Depending on A and Q, the steady state solution P_∞ may lead to a $\mathrm{Var}[\hat{J}(\mathbf{v}^*)|P_\infty]$ much larger than the required threshold $\varepsilon_1(\mathbf{v}_\ell^*)$. In addition, depending on the transition matrix A, the error covariance can evolve toward P_∞ rapidly. Increasing the number of available sensors N_S will increase the reduction rate of variance, while concurrently reducing the ratio of unmonitored to monitored basis covering the path, $\mathcal{K}$. Conversely, a larger number of parameters N_P, in general, leads to a larger $|\mathcal{K}|$ leading to faster growth of $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$.

Chapter 6

Simulation Results and Discussion

6.1 Interactive Planning and Sensing Results and Discussion

In this section, we present results of numerical simulations to illustrate the proposed IPAS algorithm, to study its performance characteristics, and to compare its performance against a typical information-driven approach from the literature.

The numerical simulations reported in this section are all performed with a square workspace $\mathcal{W} = [-1, 1] \times [-1, 1]$ in nondimensional units. The threat field is constructed as in Section 2.1. The locations $\bar{\mathrm{x}}_n$ of maxima of the basis functions are assigned to chosen to provide uniform separation between them. As discussed in Section 2.1, the constants ν_n are chosen such that the workspace is covered by the union $\cup_{n\in[N_\mathrm{P}]}\mathcal{R}_n^{\mathrm{sup}}$. The ground truth values of the parameters θ_n are randomly generated for each simulation by sampling the uniform distribution $\mathcal{U}(0, 10)$. Therefore, $\underline{\theta}_n = 0$ and $\bar{\theta}_n = 10$ for each $n \in [N_\mathrm{P}]$.

6.1.1 Illustrative Example

Figures 6.1 and 6.2 provide results of the implementation of the algorithm on an illustrative example. In this example, the algorithm termination criterion is set with the constant $\lambda_1 = 5$ in (2.13). The numbers of grid points, parameters, and sensors are $N_\mathrm{G} = 400$, $N_\mathrm{P} = 36$, and $N_\mathrm{S} = 10$, respectively. As previously discussed, the IPAS algorithm attempts to minimize the estimated path cost. To study the algorithm's performance, Fig. 6.1 shows also the true cost of the optimal path, and the *incurred cost*, which is true cost of the path found by the algorithm.

First, note in Fig. 6.1 (top) that the IPAS algorithm terminates in 5 iterations. For each of the iterations $\ell = 2, 3, 4,$ and 5, Fig. 6.2 shows the threat estimate $\hat{c}$ by a colormap, the path $\mathbf{v}_\ell^*$ of

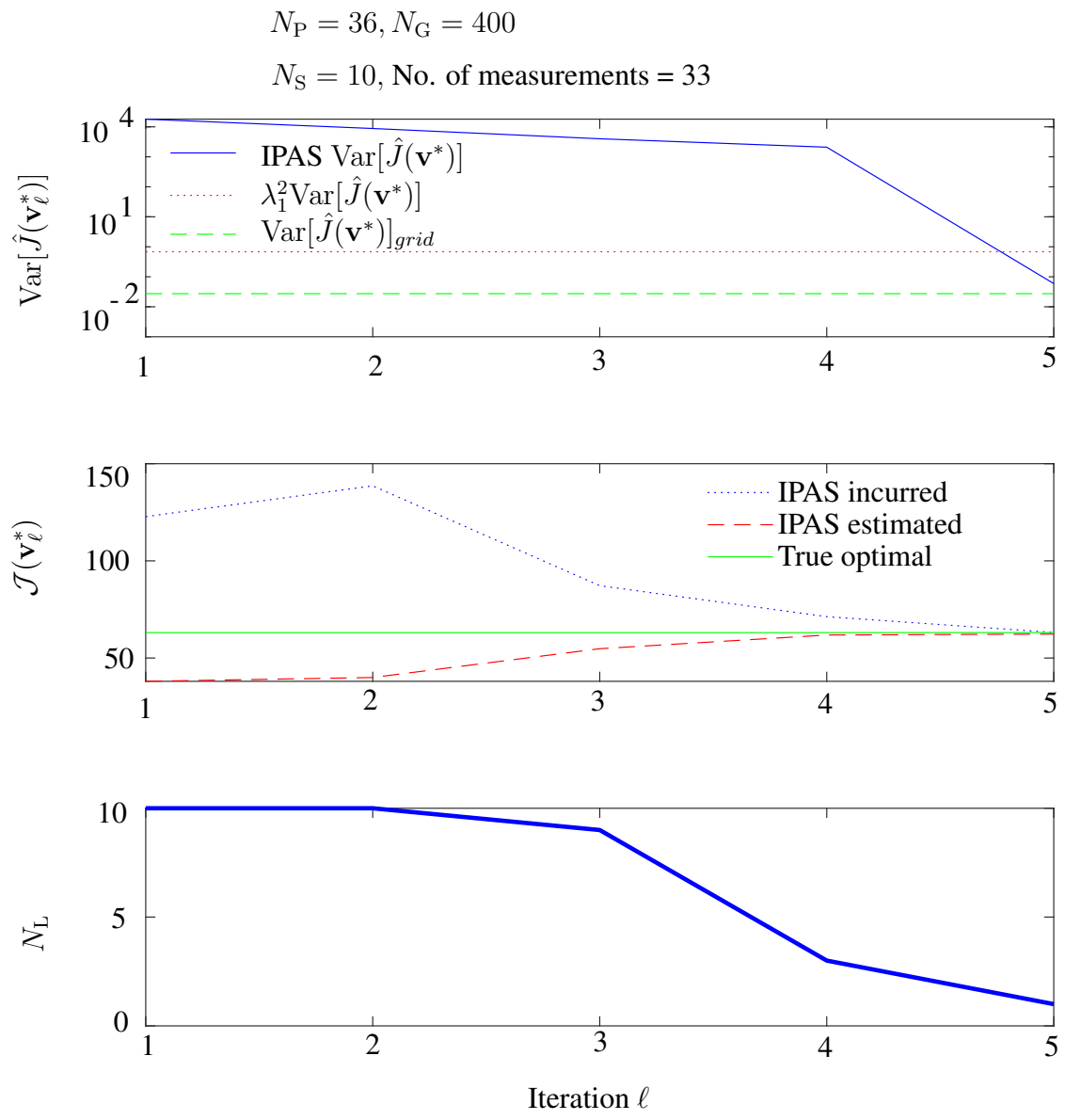


Figure 6.1: Convergence behavior of the IPAS algorithm.

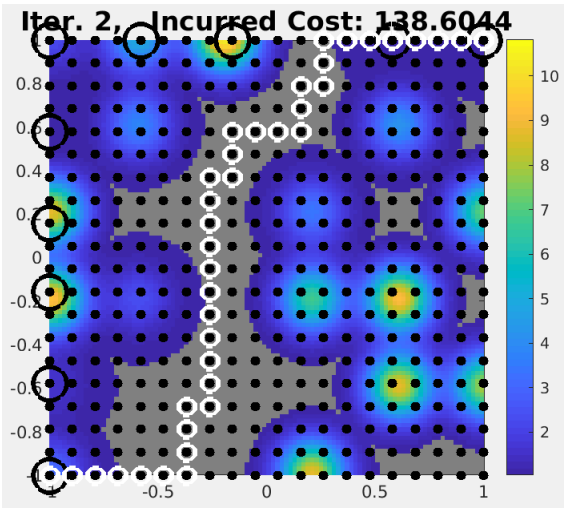
minimum estimated cost by white circles, and the sensor placement $\mathbf{s}_\ell$ by black circles. Whereas the termination criterion requires that the path cost variance be at most $\varepsilon_1(\mathbf{v}_\ell^*) = 0.683$, in this case, at termination the actual path cost variance is much lower, namely $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] = 0.059$. In Fig. 6.2, the gray regions are of significant support of bases with non-identified parameters, i.e., those ϕ_n with $n \in \mathcal{I}^{\mathrm{C}}$. Because the IPAS algorithm finds optimistic threat estimates, as discussed in 2.3, these gray regions indicate zero contribution from the basis functions to the estimated threat. Therefore, in Fig. 6.2(a) for example, the path $\mathbf{v}_2^*$ traverses through the gray regions.

Second, note in Fig. 6.1 (middle) that upon termintation, the incurred and estimated costs are nearly identical: namely $\mathcal{J}(\mathbf{v}_\ell^*) = 63.16$ and $\hat{J}(\mathbf{v}_\ell^*) = 62.33$ respectively. This observation agrees also with the low variance in the estimated cost at termination. More importantly, note that the incurred and estimated costs are nearly identical to the *true* optimal cost, which agrees with the theoretical result of Theorem 1. Before termination, the estimated cost is lower than the true optimal cost because of the optimistic threat estimates. Conversely, the incurred cost of a path at a given iteration will be greater than or equal to the true optimal cost.

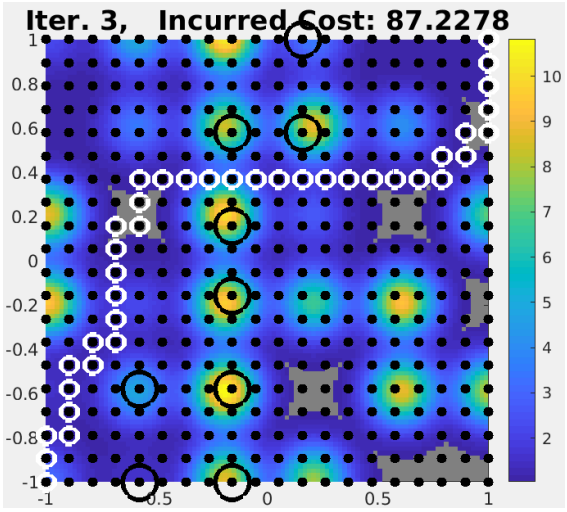
Third, note in Fig. 6.1 (bottom) that the number of sensors placed in each iteration is not always N_{S}. The sum of the numbers of sensors placed at each iteration, over all of the five iterations, is 33. Crucially, note that the total number of measurements taken by the IPAS algorithm is less than the number of parameters, and yet the path found by the algorithm is nearly identical in cost to the *true* optimal path. This example therefore illustrates the most powerful feature of the IPAS algorithm: not only can it find a near-optimal path with $N_{\mathrm{S}} \ll N_{\mathrm{P}}$, but it can find a near-optimal path with fewer *measurements* than parameters, i.e., even with $\sum_{\ell=1}^{\bar{\ell}} N_{\mathrm{L}}(\ell) < N_{\mathrm{P}}$, where $N_{\mathrm{L}}(\ell)$ is the number of sensors placed at each iteration, and $\bar{\ell}$ is the total number of iterations. In Section 6.1.4, we emphasize this feature with additional results.

6.1.2 Convergence and Optimality

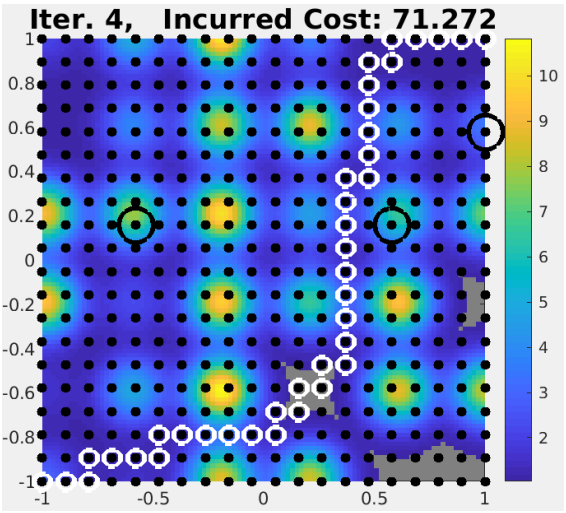
To corroborate the theoretical results Proposition 1, Theorem 1, and Proposition 3 with numerical simulation results, we performed a study with varying number of sensors and parameters. To this end, we performed numerical simulations for every combination of $N_{\mathrm{P}} \in \{4, 9, 16, 25, 36, 49, 64, 81\}$



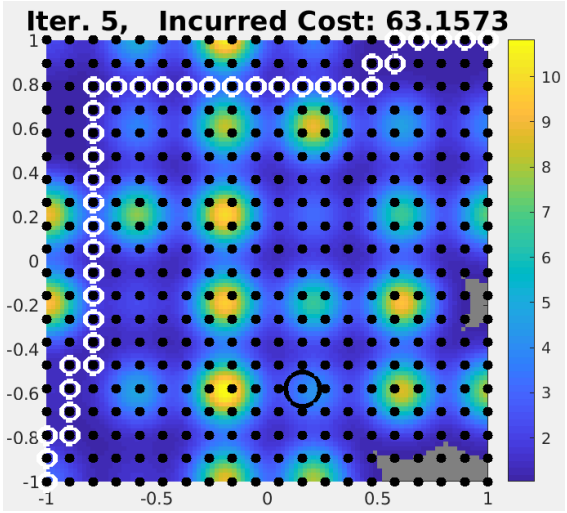
(a) Iteration 2.



(b) Iteration 3.



(c) Iteration 4.



(d) Final iteration 5.

Figure 6.2: Visualization of the iterative and interactive planning and sensing (IPAS) process for $N_P = 36$ and $N_S = 10$.

with $N_{\mathrm{S}} \in \{1, 2, \ldots, 90\}$, and with fixed $N_{\mathrm{G}} = 400$ and $\lambda_1 = 5$. The results gathered for every combination $(N_{\mathrm{S}}, N_{\mathrm{P}})$ were averaged over 100 simulations, for different threat fields with randomly generated parameter values. The algorithm was programmed to terminate after a maximum of 30 iterations unless the previously discussed $StopCondition$ was reached earlier.

Figure 6.3 shows results on the convergence of the proposed algorithm, i.e., the number of iterations $\bar{\ell}$ required to terminate "normally" by the $StopCondition$ criterion. As intuitively expected, the number of iterations decreases with increasing number of sensors, irrespective of the number of parameters. Indeed, the data plotted in Fig. 6.3 are empirically approximated by the curve $\bar{\ell} = 29.8\exp(-0.23N_{\mathrm{S}}) + 9.09\exp(-0.0056N_{\mathrm{S}})$. As was the case with the illustrative example of Section 6.1.1, the algorithm converges in a small number of iterations even in cases where $N_{\mathrm{S}} \ll N_{\mathrm{P}}$. For example, note that for $N_{\mathrm{P}} = 81$, the algorithm converges in less than 10 iterations with as few as 13 sensors.

Figure 6.4 shows the striking result that the incurred costs of paths found by the IPAS algorithm are almost always identical to the *true* optimal cost. The exceptions are cases where an extremely small number of sensors is used to estimate a field with a large number of parameters, e.g. $N_{\mathrm{S}} = 1$ and $N_{\mathrm{P}} = 81$. Even in this case, the suboptimality is due to the forcible termination of the algorithm after 30 iterations. If additional iterations are allowed, then the algorithm converges to a near-optimal path even in these cases.

As described in Section 2.5, the constant λ_1 characterizes a level of path cost variance that must be achieved before the algorithm terminates. Lower values of λ_1 provide more stringent termination criteria by requiring a higher confidence (lower variance) in the estimated path cost. In Figure 6.5, the convergence results from a set of simulations run with $\lambda_1 = 1.2$ show an increase in the iterations required. Note that for higher N_{S} the number of required iterations is similar for $\lambda_1 = 5$ and $\lambda_1 = 1.2$.

6.1.3 Comparisons with Information-Driven Approaches

We compare the proposed IPAS algorithm to two information-driven approaches: one that attempts to maximize the so-called *frame potential,* and the other that attempts to minimize mean

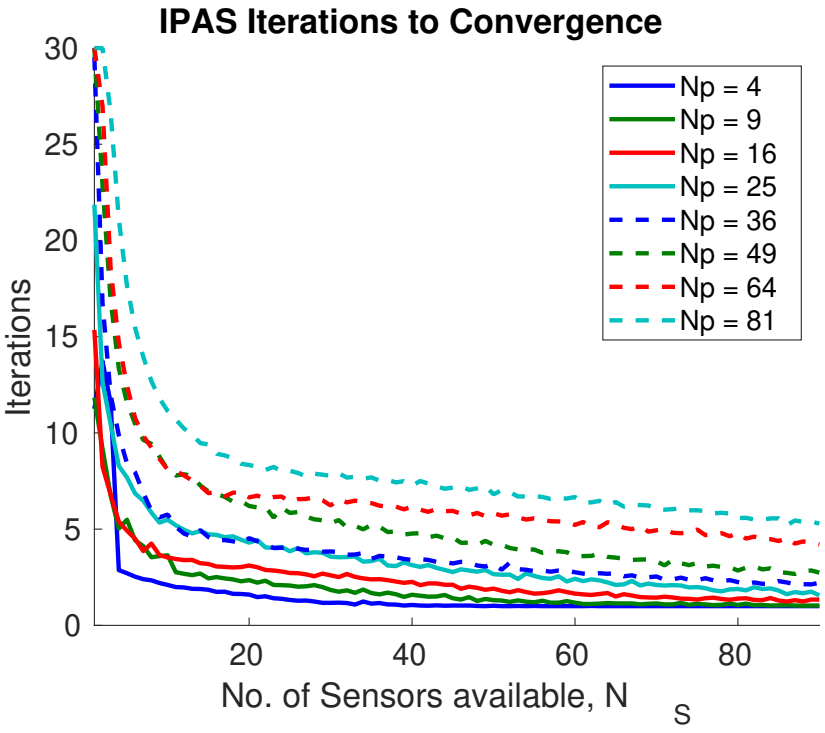


Figure 6.3: Number of iterations until convergence with $\lambda_1 = 5$.

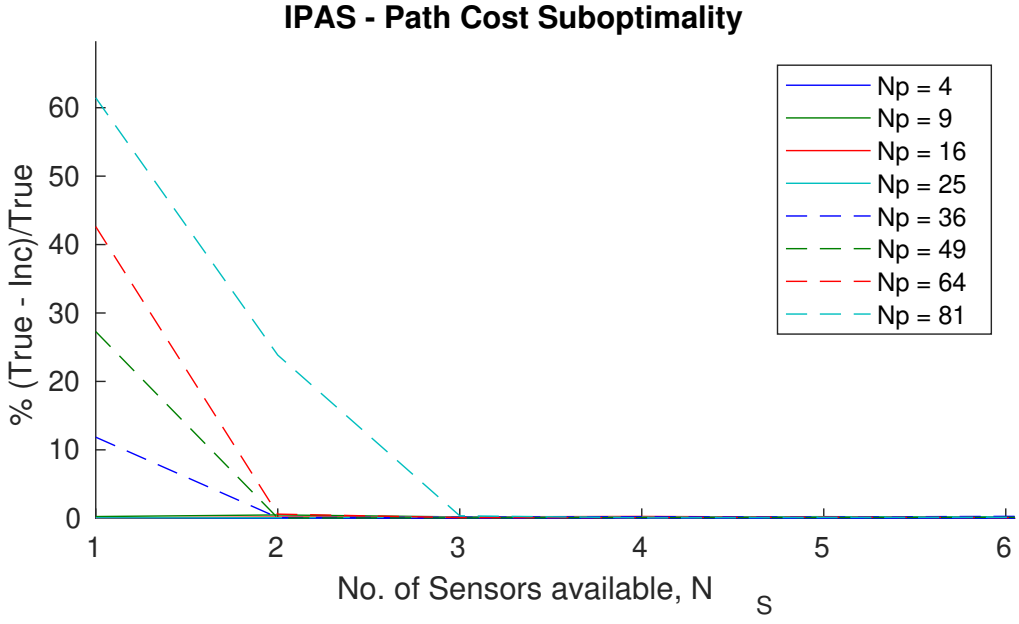


Figure 6.4: Suboptimality in pathological cases where the number of sensors is extremely small, and when the number of iterations of the IPAS algorithm are limited.

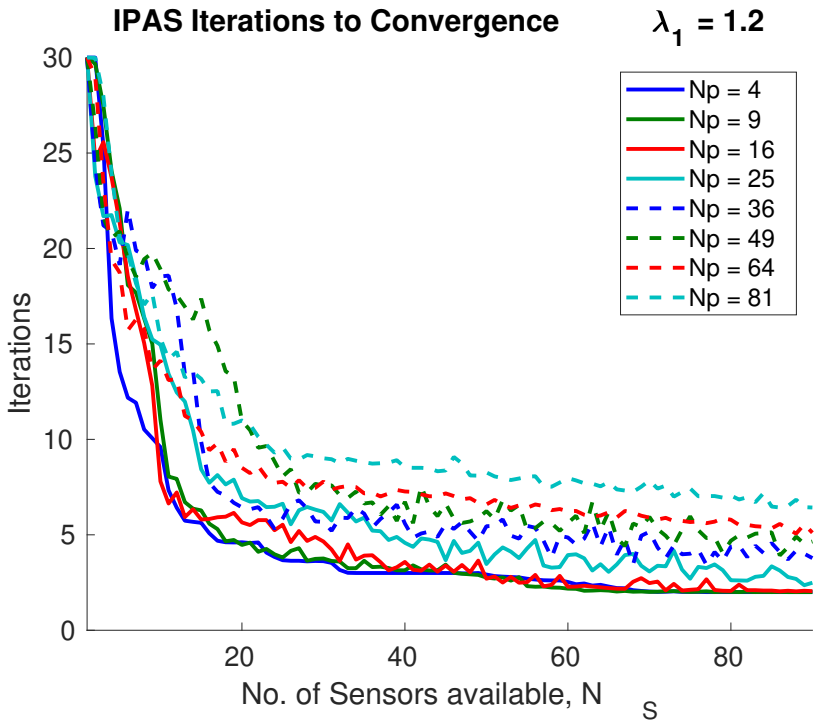
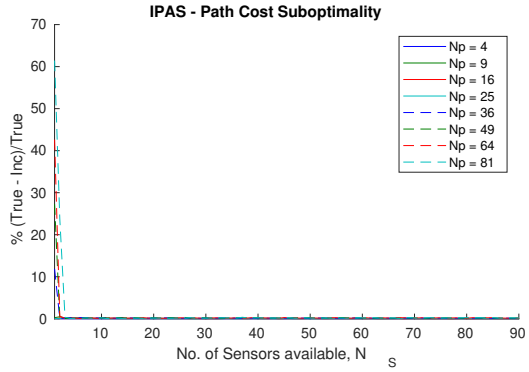


Figure 6.5: Number of iterations until convergence with $\lambda_1 = 1.2$.

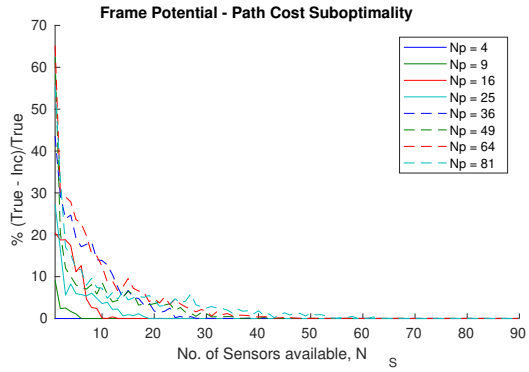
squared error. Frame potential is a scalar metric that measures the orthogonality of the rows of the measurement matrix, and a near-optimal greedy placement algorithm and its software implementation[1] are readily available in the literature (Ranieri et al., 2014). Initial simulations also considered comparisons with mutual information (Krause et al., 2008). However, for our application, mutual information performed similarly to frame potential but required up to two orders of magnitude more computation time and was omitted from the large simulation studies. Numerical simulations were performed for these two methods for the same set of combinations of N_{P} and N_{S} as discussed in Section 6.1.1. For each $(N_{\mathrm{P}}, N_{\mathrm{S}})$ combination, we performed 100 simulations with different randomly generated threat fields. In addition we use sensor placement that randomly assigns locations without repetition (no location assigned twice). Comparing against random sensor placement is a commonly used "sanity check" in the literature (Krause et al., 2008; Ranieri et al., 2014).

Figure 6.6 shows the optimality of paths resulting from these two information-driven approaches and from randomized sensor placement, compared to the path resulting from the IPAS

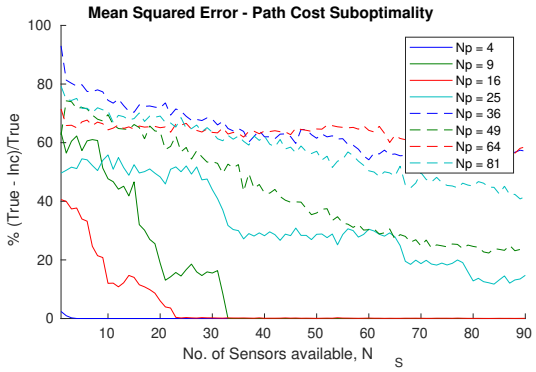
[1] https://github.com/jranieri/OptimalSensorPlacement.



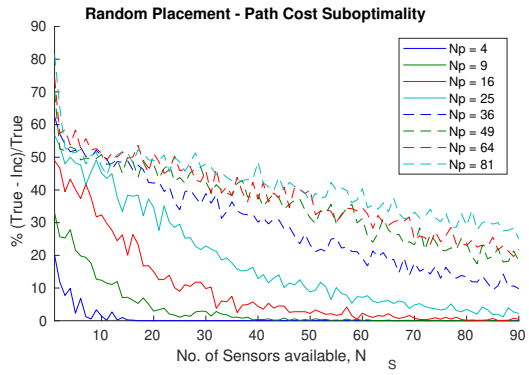
(a) IPAS algorithm.



(b) Maximum frame potential (Ranieri et al., 2014).



(c) Minimum mean squared error.



(d) Random sensor placement.

Figure 6.6: Simulations results summarizing the path cost percentage suboptimality compared to the true optimal cost.

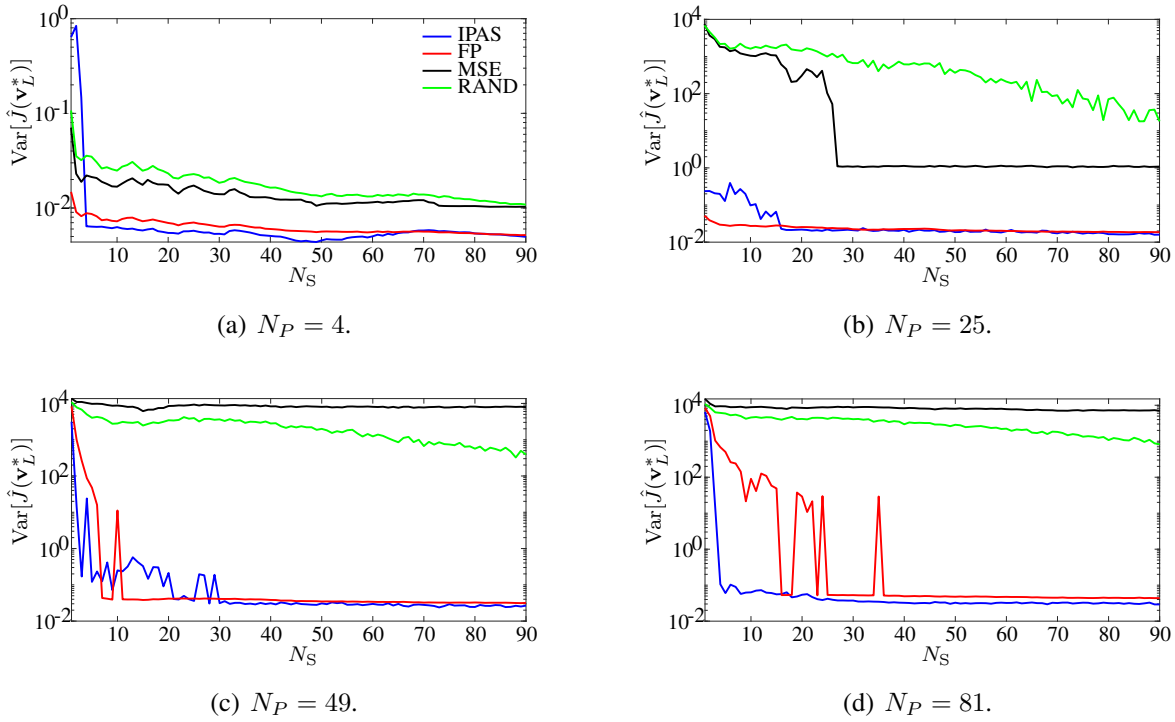
approach. The percentage difference between the costs of these path and the cost of the true optimal path is indicated. These plots show that (1) the proposed IPAS algorithm consistently finds a path with cost nearly identical to the true optimal cost, and (2) the IPAS algorithm outperforms the other two information-driven approaches especially in situations where $N_{\mathrm{S}} \ll N_{\mathrm{P}}$. Note, for example, that for $N_{\mathrm{S}} \leqslant 30$ the results of the two information-driven approaches are significantly suboptimal for several N_{P}, whereas the IPAS algorithm returns near-optimal paths. As discussed in Section 6.1.1, the instances of sub-optimal results of the IPAS algorithm in Figs. 6.4 and 6.6(a) are due to limiting the number of iterations at 30.

To ensure that these comparisons were fair, each of the three comparative methods are allowed the same number of *measurements* as the IPAS algorithm, i.e., the previously stated quantity $\sum_{\ell=1}^{\bar{\ell}} N_{\mathrm{L}}(\ell)$, which is greater than N_{S}.

Next, we compare the variance of estimated costs of the paths resulting from the aforesaid three comparative approaches to that resulting from the IPAS algorithm. Figure 6.7 indicates this variance for various values of N_{P}. Recall that the termination criterion of the IPAS algorithm is based on this variance, and by Prop. 1, this variance is guaranteed to be at most equal to the threshold $\varepsilon_1(\mathbf{v}_{\bar{\ell}}^*)$.

In each of the cases shown in Fig. 6.7, the IPAS algorithm achieves a path cost variance at least as low, and often slightly lower than that achieved by the frame potential method. The path cost variances achieved by the other two approaches is higher by at least two orders of magnitude. For cases with $N_{\mathrm{S}} \ll N_{\mathrm{P}}$, the termination limit of 30 iterations often causes the IPAS algorithm to result in a higher path cost variance. However, even in these cases, Fig. 6.7 shows that these variances are of the same order of magnitude as those achieved by the frame potential method.

To summarize, not only does the IPAS algorithm consistently find paths whose true costs are near-optimal, but it also ensures that the confidence in the estimated path cost is high (i.e., path cost variance is low).



(a) $N_P = 4$.

(b) $N_P = 25$.

(c) $N_P = 49$.

(d) $N_P = 81$.

Figure 6.7: Path cost variance behavior for each sensor placement strategy at selected N_P.

6.1.4 Performance in Parameter-Rich & Resource-Constrained Scenarios

To further emphasize the fact that the IPAS algorithm can achieve near-optimal paths with far fewer measurements than the number of parameters, we present here an example with $N_\mathrm{P} = 100$ and $N_\mathrm{S} = 10$, is an *order of magnitude* smaller than N_P.

Figure 6.8 shows the path resulting from the IPAS algorithm on this problem. Note the significant regions of the map which remain gray, corresponding to many parameters that were not identified. The estimated, incurred, and true path costs, the variance of the estimated path cost, and the numbers of sensor placements at each iteration are shown in Fig. 6.9. Note that the path resulting from the IPAS algorithm is in fact a true optimal path.

By comparison, an information-driven approach will necessarily place sensors in the gray regions, whereas those sensor measurements will provide no benefit for this particular path planning problem. In this example, the IPAS algorithm terminated normally in 8 iterations, and the total number of measurements taken over all of these iterations was 55, which is roughly half the num-

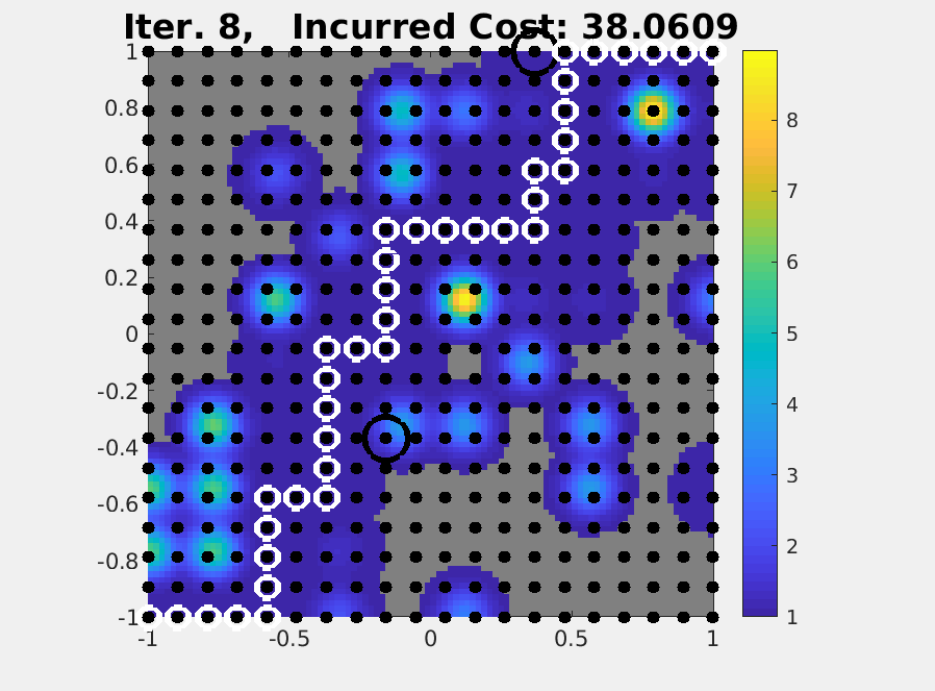


Figure 6.8: Example with $N_P = 100$ and $N_S = 10$, the requirement number of measurements for convergence is only 55.

ber of parameters N_{P}. In contrast, when the frame potential method with 55 measurements was implemented, the resultant path with minimum *estimated* cost was significantly suboptimal compared to the *true* optimal path. The reason for this behaviour is that, as shown in Fig. 6.9 (top), the variance of the estimated path cost due to the frame potential method remained four orders of magnitude higher than that resulting from the IPAS algorithm.

The maximization of information in the data collection stage can have different objective functions, such as mutual information or frame potential, but all metrics typically strive to uniformly gather data in environment or other information space. This uniformity of measurement explains why information approaches perform poorly in comparison to the IPAS approach in high parameter spaces. As the number of parameters increases, the probability that sensors will be placed with sufficient density around the true path decreases. Therefore, to maintain optimality and confidence of path cost estimates in environments characterized by a large number of parameters and/or a relatively small number of sensors, the IPAS algorithm is more suitable.

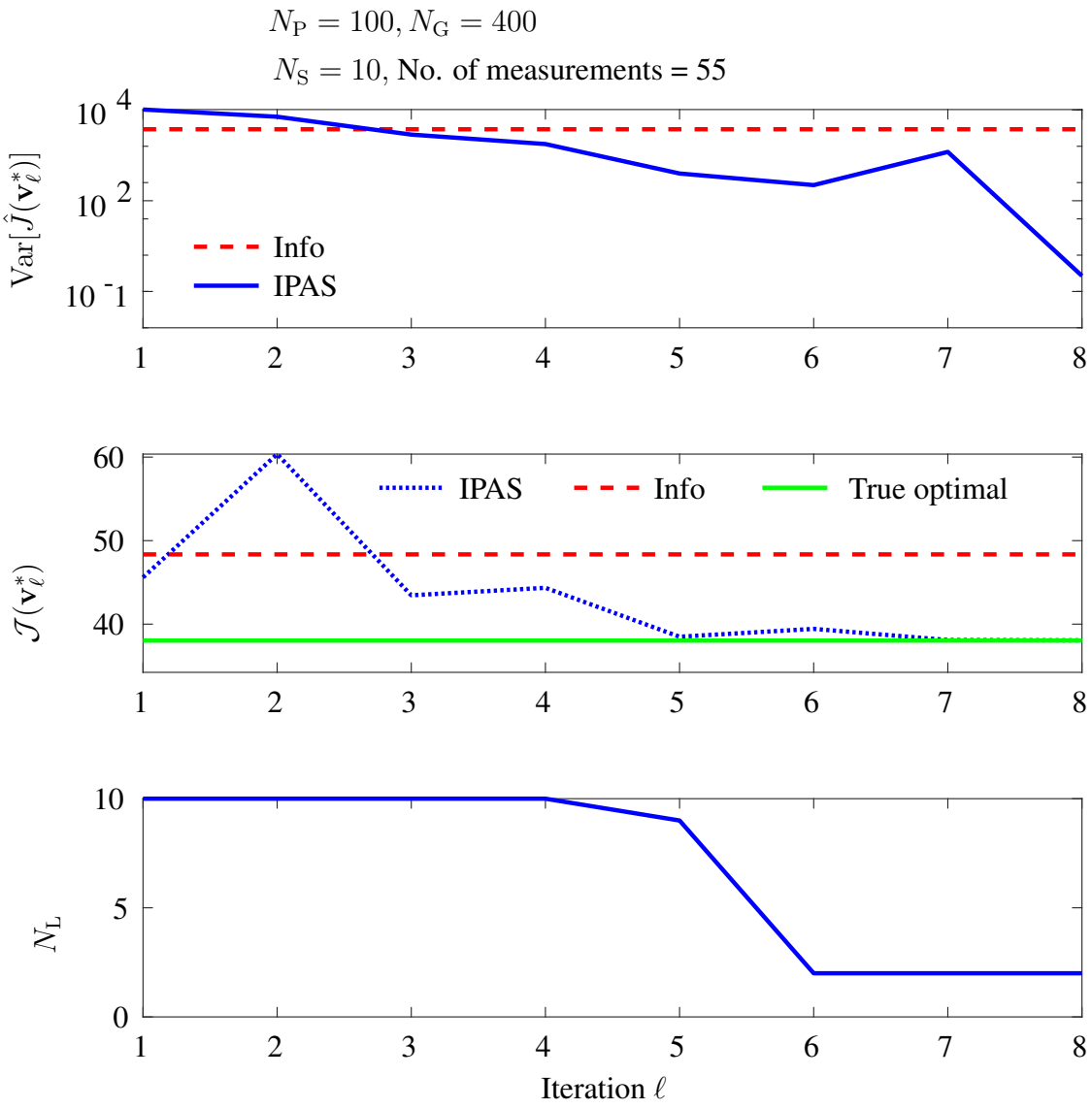


Figure 6.9: Comparison of IPAS approach and an information-driven approach (frame potential).

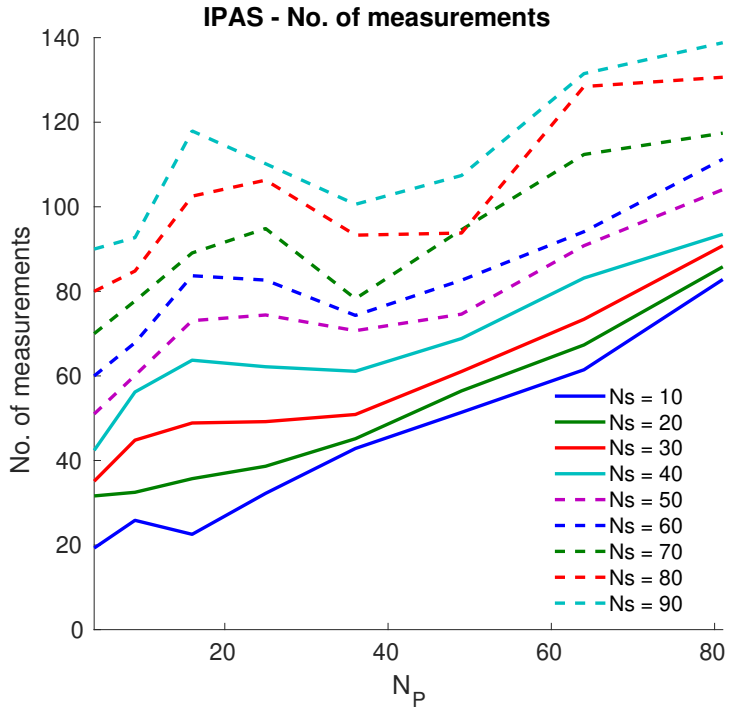


Figure 6.10: The total number of measurements required given an $N_P - N_S$ pair were tracked over all simulations. The number of required total measurements increases slowly with N_P.

6.1.5 Computational Complexity

The computational complexity of the IPAS algorithm depends on the number of calls to Dijkstra's algorithm in Line 7 in Fig. 2.1. Note that it is trivial to replace Dijsktra's algorithm in Line 7 with the faster A* algorithm (Nilsson, 1998), but the worst-case computational complexity of both algorithms is identical. Specifically, the worst-case complexity of either algorithm is $O(N_{\mathrm{G}} + N_{\mathrm{G}} \log(N_{\mathrm{G}}))$, when the algorithm is implemented using a Fibonacci heap data structure (Nilsson, 1998). The total number of iterations of the algorithm scales linearly with the number of parameters N_{P}. Therefore, the worst-case computational complexity of the IPAS algorithm is $O(N_{\mathrm{P}}(N_{\mathrm{G}} + N_{\mathrm{G}} \log(N_{\mathrm{G}})))$.

Figure 6.10 shows the number of measurements required for termination. An approximate linear increase in th number of measurements is observed with an increasing number of parameters N_{P}.

6.1.6 Discussion of IPAS Algorithm and Alternative Comparisons

The traditional approach to the solution of Problem 1 is to first deploy sensors to optimize some metric about the quality of the threat field estimate, and to then plan a path using the estimated field. We refer to such traditional approaches as *information-driven,* to highlight the fact that the objective of sensor placement is to obtain as much information about the threat field as possible without regard to the path-planning problem at hand.

A key observation here, which was also used in the proof of Theorem 1, is that the solution of Problem 1 improves as the variance of the estimated path cost decreases. In other words, for a path $\hat{\mathbf{v}}^*$ with minimum estimated cost, if $\mathrm{Var}[\hat{J}(\hat{\mathbf{v}}^*)]$ is small, then the estimated path cost $|\hat{J}(\hat{\mathbf{v}}^*) - \mathcal{J}(\hat{\mathbf{v}}^*)|$ is small with a high probability. Consequently, $\hat{\mathbf{v}}^*$ is a near-optimal path with high probability. It follows from (2.12) that the sensor placement problem in Problem 1 must attempt to minimize $\mathrm{tr}[P_{\bar{\ell}}|_{[\mathcal{K}]}]$. Recall from (2.10) that $\mathcal{K}$ is the minimal cover of the path $\hat{\mathbf{v}}^*$.

In an information-approach, regardless of which specific method is used for sensor placement, $\mathrm{tr}[P_{\bar{\ell}}|_{[\mathcal{K}]}]$ is at least as large as that obtained from the IPAS algorithm. This claim is true because the information-driven approach is not concerned with $\mathcal{K}$ at all, whereas the IPAS algorithm specifically attempts to reduce $\mathrm{Var}[\hat{J}(\hat{\mathbf{v}}^*)]$, and therefore implicitly reduces $\mathrm{tr}[P_{\bar{\ell}}|_{[\mathcal{K}]}]$. We assume that the total number of measurements made is the same in either case, and that the threshold $\varepsilon_1(\mathbf{v}^*_{\bar{\ell}})$ is sufficiently small to prevent premature termination of the IPAS algorithm.

Based on these observations, the following result is true.

Proposition 3. *Let $\hat{\mathbf{v}}^*_{\mathrm{IPAS}}$ be the solution to Problem 1 returned by the IPAS algorithm, and let $\hat{\mathbf{v}}^*_{\mathrm{ID}}$ be the solution to Problem 1 returned by an information-driven approach. Then $\mathcal{J}(\hat{\mathbf{v}}^*_{\mathrm{IPAS}}) \leqslant \mathcal{J}(\hat{\mathbf{v}}^*_{\mathrm{ID}})$.*

6.1.7 Comparison with Blackbox Optimization

It is clear that Problem 1 can be formulated as an optimization problem where the set of all possible sensor placements $\mathbf{s}$ is treated as the design space, and the cost $\hat{J}(\mathbf{v})$ is the objective function to be minimized. The least squares estimates of parameters using the sensor measurements,

and the execution of Dijkstra's algorithm for computing the path $\mathbf{v}$ with minimum estimated cost can be wrapped inside a "blackbox" with input $\mathbf{s}$ and output $\hat{J}(\mathbf{v})$. The inner operations of this blackbox need not be visible or available to an optimization algorithm based on, say, gradient descent or randomized sampling. This practice is common in the optimal design of experiments, and arguably it can be applied to the solution of Problem 1.

One obvious issue with this approach is that the size of the design space expands combinatorially with increasing number of sensors N_{S}. A more serious issue is that $\mathbf{s}$ is in fact an inappropriate design vector. As is clear from the proposed IPAS algorithm, when the number of sensors is small, multiple iterative sensor placements are needed, and the number of *measurements* exceeds the number of sensors. (see Sections 6.1.1 and 6.1.4 for illustrative examples). Therefore, the appropriate design vector for the preceding blackbox optimization problem is the set of *measurements,* possibly achieved via multiple iterative sensor placements. This design space is, of course, much larger than that of the sensor placements. What is worse, is that we do not *a priori* know how many measurements are needed to reduce the variance of the estimated path cost below a desired threshold. In other words, the size of the design space is variable, which renders the blackbox approach impractical for this particular problem, irrespective of the method used for optimization.

6.2 Waiting in Spatiotemporal Fields Illustrative Example and Discussion

The results in this section are summarized as follows. The allowance for waiting in the path-planning algorithm can reduce paths costs by 25% compared to no-wait path-planning, but this cost reduction incurs an added computational expense of a 30- to 300-fold increase in execution time. Using the local no-wait condition of Section 3.4 reduces this added computation expense to only a 10-fold increase in execution time, while discovering a majority of optimal paths.

6.2.1 Uniformly High Resolution Waiting vs. Non-waiting Solution

In what follows, we present the results of the numerical study described in Section 3.3.1, specifically, the characteristics of the difference in costs between the path solving Problem 2 and the no-wait path: $\mathcal{J}\left(\mathbf{v}^*_{\mathrm{nw}}\right) - \mathcal{J}\left(\mathbf{v}^*\right)$ normalized by $\mathcal{J}\left(\mathbf{v}^*\right)$.

Computational effort for allowance of waiting. Over the 2000 simulations conducted, the greatest suboptimality of the no-wait path was 25% . In 156 of these simulations, i.e., nearly 8% of the simulated cases, $\mathbf{v}^*_{\mathrm{nw}}$ was suboptimal compared to $\mathbf{v}^*$, but this suboptimality was significant (greater than 5%) in only in less than 1% of cases of the 2000 simulations. The computation times for finding the no-wait path in each simulation were all under 0.4 s, while those for finding the solution to Problem 2 were in the range of 11–120 s as shown in the histogram in Fig. 6.11. In other words, the computation time to find the solution $\mathbf{v}^*$ to Problem 2 was between 30–300 times slower than the time to find the no-wait approximate solution $\mathbf{v}^*_{\mathrm{nw}}$, whereas the suboptimality $\mathcal{J}\left(\mathbf{v}^*_{\mathrm{nw}}\right) - \mathcal{J}\left(\mathbf{v}^*\right)$ was greater than zero in slightly less than 8% of the cases simulated. In certain applications, however, this extra computational effort may be justified, say in the transportation of hazardous waste through adverse weather (threat) (Akgüna et al., 2007).

Local no-wait condition – Cost reduction and computational efficiency. The discussion in Section 3.4 indicates that waiting is locally beneficial when the waiting cost α_{w} and the threat field value are low compared to the local temporal gradient of the threat field. It was noted that the movement cost α_{m} did not affect this local test, and therefore the ratio $\alpha_{\mathrm{m}}/\alpha_{\mathrm{w}}$ is not relevant to the waiting decision. In Fig. 6.12(a), we note that the cost reduction of waiting occurs more frequently with a lower minimum values of the field c. Figures 6.12(b)-(d) summarize the average value across time for the field and its gradients.

When the field temporal gradient is small, i.e., $\nabla c(\mathsf{x}, t) \to 0$, the field is approximately static and an optimal path does not involve waiting. For further insight into the lack of waiting paths for fields with large temporal gradients, consider a vehicle at location x^k at time t_j and the threat value $c(\mathsf{x}^\ell, t_j)$ at location x^ℓ along the optimal path $\mathbf{v}^*$. Due to the large gradient, the threat will move or dissipate at time $t_{j+m\Delta t}$. If the threat at position x^ℓ dissipates such that $\nabla c(\mathsf{x}^\ell) \geqslant \frac{d\mathsf{x}}{dt}$, where $\frac{d\mathsf{x}}{dt}$

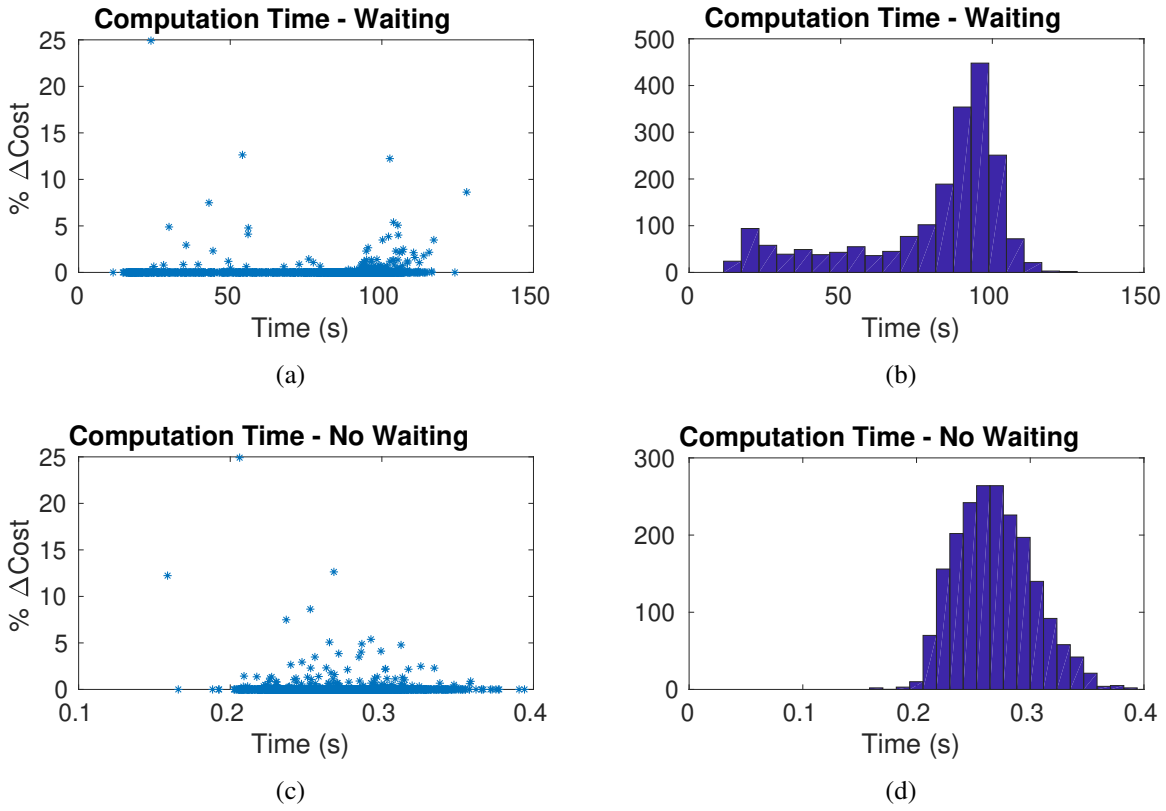


Figure 6.11: Execution time to solve Problem 1. (a),(b) The computational expense with waiting included is bimodal with the dominant mode at higher computation times. Bimodality indicates calculation of waiting paths falls into two populations: optimal waiting paths found in a narrow search of $\mathcal{GT}$, and optimal waiting paths found in a broader search of $\mathcal{GT}$. This observation identifies two characteristically different threat field families. (c),(d) Computational expense (execution time) when waiting is not allowed.

is the speed of the vehicle, then the vehicle arrives after the threat is dissipated and no waiting is required. However, if the vehicle arrives at position x^ℓ before time $t_{j+m\Delta t}$, it will incur a threat exposure cost and should wait. These arguments indicate that optimal paths may involve waiting when the field changes moderately over time, i.e., $\nabla c(\mathsf{x}, t)$ is neither too high nor too low.

As discussed in Section 3.4, we can modify Dijkstra's algorithm with the local no-wait condition (3.8), which leads to a significant computational advantage. Using this local no-wait condition check, the average calculation time is reduced from 78.7s to 5.47s, which is a 140% reduction (see Fig. 6.13). With the local no-wait condition check, 83% of paths are determined in under 6 seconds while still discovering optimal paths that involve waiting. Because this no-wait condition is approximate, potentially optimal waiting paths can be discarded. Among the 2000 simulations conducted, 156 cases involved waiting in the optimal path, but with the local no-wait condition

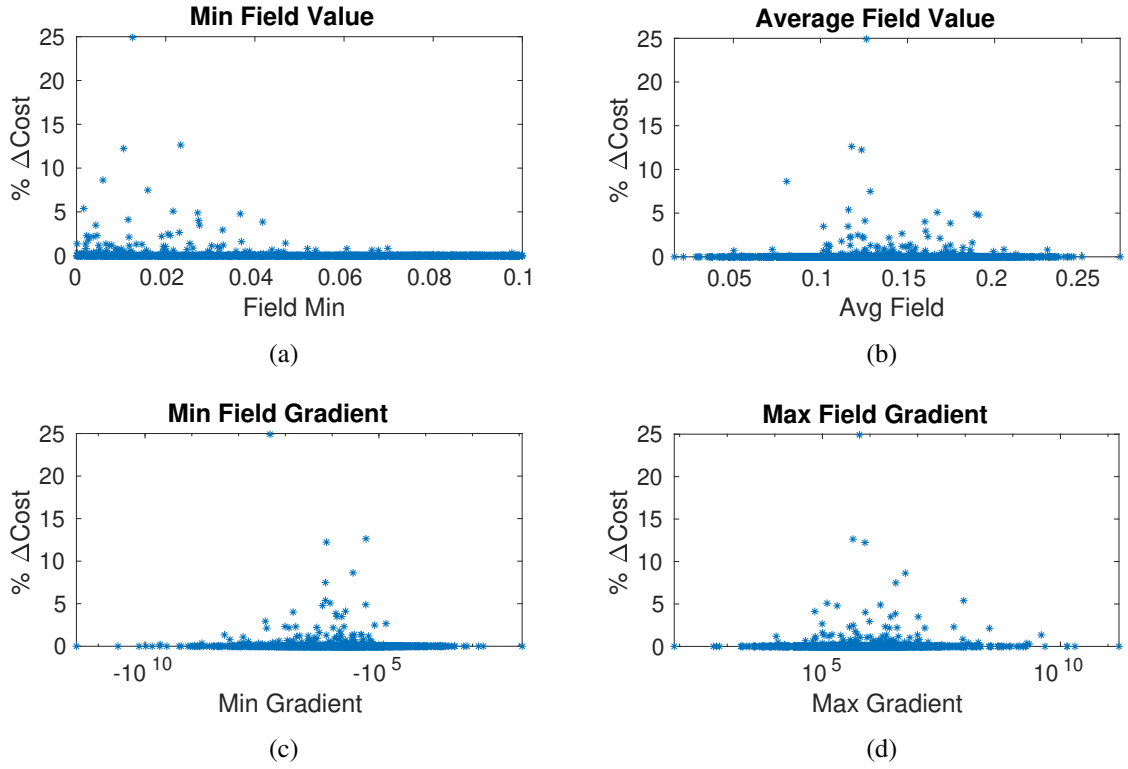
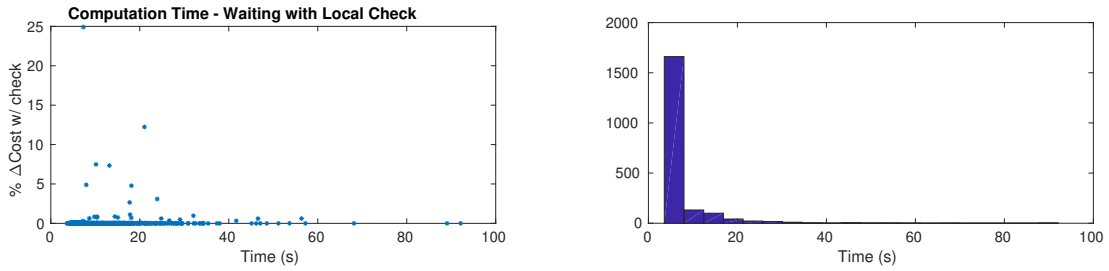
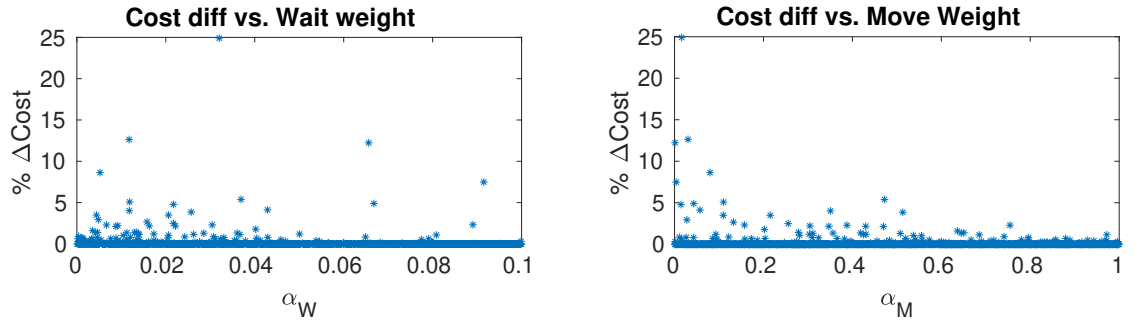


Figure 6.12: Difference in costs of waiting-allowed and no-wait optimal paths, for different field characteristics. When the minimum field value is low, optimal paths more frequently include waiting. The greatest differences between paths with and without waiting cluster near a median value suggesting that large and small temporal gradients of the field favor movement over waiting.



(a) (b)

Figure 6.13: Computational efficiency due to local no-wait condition. When the local no-wait condition (3.9) is used to prune search trees the calculation time for the majority of paths is significantly reduced while still finding optimal paths that include waiting.



(a) (b)

Figure 6.14: Difference in costs of waiting-allowed and no-wait optimal paths, for different values of constants in edge transition cost. For lower values of the constants α_{w} and α_{m}, optimal paths more frequently involve waiting.

check included, only 13 of these 156 waiting paths were found by the algorithm. However, these included paths with the greatest reduction in cost: specifically, the reduction of cost of the waiting paths (compared to an optimal no-wait path) with a cost difference greater than 5%, 57% were found by the algorithm including the local no-wait check condition. In summary, the local no-wait condition was observed to significantly reduce computational efforts while finding optimal paths in a majority of the simulated cases.

Other dependencies. Figure 6.14 indicates that a low value of the movement cost constant α_{m} in the edge transition cost (3.2), the optimal path more frequently involves waiting. Such cases may occur when the optimal no-wait path $\mathbf{v}^*_{\mathrm{nw}}$ is shorter (i.e., has fewer vertices) but has higher path cost $\mathcal{J}\left(\mathbf{v}^*_{\mathrm{nw}}\right)$ compared to when that $\mathbf{v}^*_{\mathrm{w}}$ is longer than $\mathbf{v}^*_{\mathrm{nw}}$, but has lower cost path due to

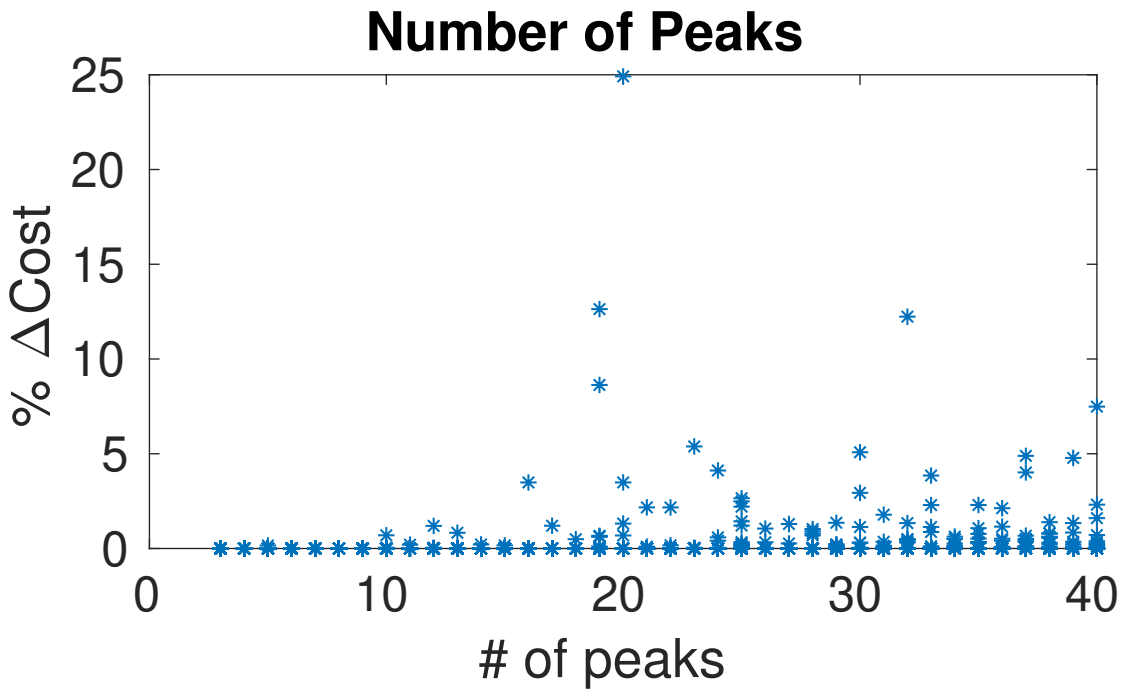


Figure 6.15: Difference in costs of waiting-allowed and no-wait optimal paths, for different numbers of field parameters. For fields with a larger number of parameters (i.e. peaks) N_{P}, the optimal path more frequently involves waiting.

threat exposure. For these simulations the waiting cost constant α_{w} was restricted to the interval $(0, 0.1]$.

The number of parameters N_{P} used to define the threat field also demonstrated a pattern with respect to cost-reduced waiting paths. In Fig. 6.15, an increase in N_{P} is associated with a greater number of optimal paths involving waiting. An increase in the density of peaks in the field may lead to lower spatial gradients of the field (min and max), which influences optimal waiting-allowed paths in a manner similar to the aforesaid influence of the temporal gradient.

6.2.2 A* Path-Planning Algorithm using Traditional Heuristics

We simulated over 800 environments with spatial size 10x10 and temporal resolution of 40 (10x10x40) and tested each of the four algorithms. A strong positive correlation was found between the number of visited nodes and the computation time. As seen in Figures 6.16 and 6.17, without using the new heuristic, computation time for the A* algorithm with waiting often took

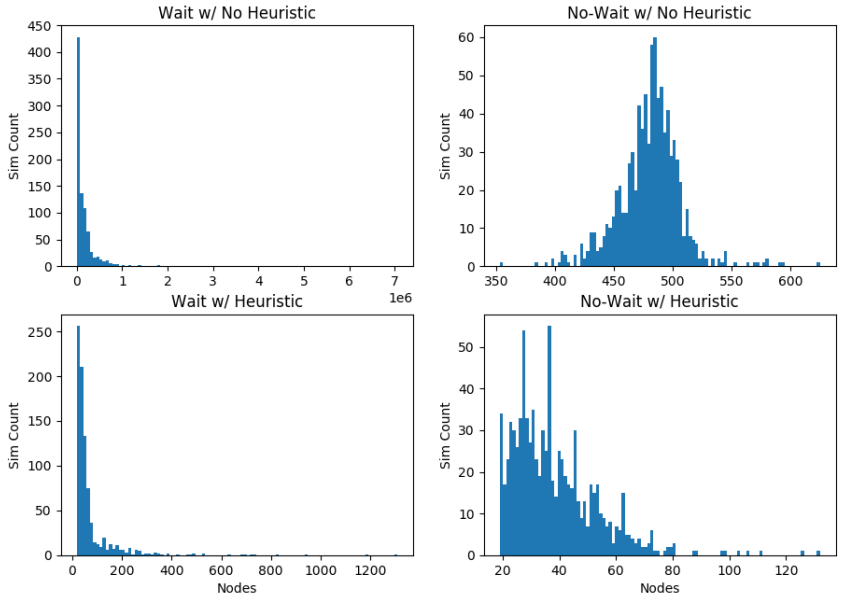


Figure 6.16: Results of Heuristic Speed up. Heuristic drastically changed the number of visited nodes, therefore improving the efficiency of searching.

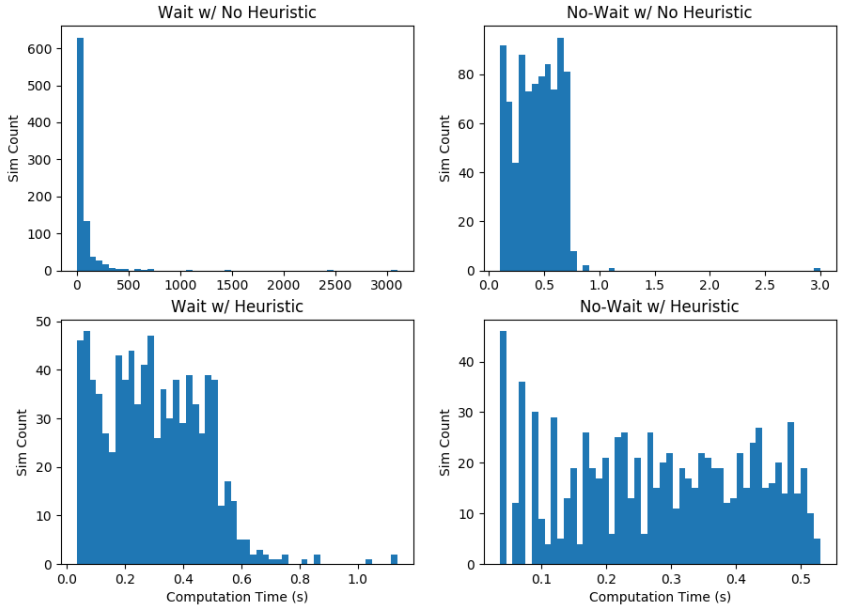


Figure 6.17: Results of Heuristic Speed up. There was a strong reduction in the computation time for A with waiting.*

multiple minutes, exploring hundreds of thousands of nodes. In extreme cases, computation time was almost an hour for a single path-planning problem. Meanwhile, when the heuristic was added to the A* algorithm with waiting, the vast majority of computation times fell to a fraction of a second with extreme cases taking little over a second to compute.

Meanwhile, the A* algorithm without waiting achieved computation times ranging from tenths of a second to just over a second even without the heuristic, as the number of nodes to be explored numbered only in the hundreds rather than the hundreds of thousands. Even so, the algorithm displayed significant improvement when the heuristic was added. The number of nodes explored was reduced drastically, such that even outlying cases explored just over a hundred nodes. This reduction in explored nodes resulted in a corresponding reduction in computation time, roughly halving the computation time of the algorithm on average.

In order to understand the impact of these encouraging results, we scaled up the environment size and performed the same set of simulations. The results of a 30x30x80 environment set are presented in the histogram of Figure 6.18. The trend of strong reductions in computation burden are maintained, and we see a reduction of nearly two orders of magnitude in the waiting search case. In Figure 6.19, the mean value of nodes generated for each environment and search case highlight the relative reductions for waiting and non-waiting search. Although there is strong reduction in the space searched for waiting, the nodes generated are still an order of magnitude above the non-waiting algorithm case.

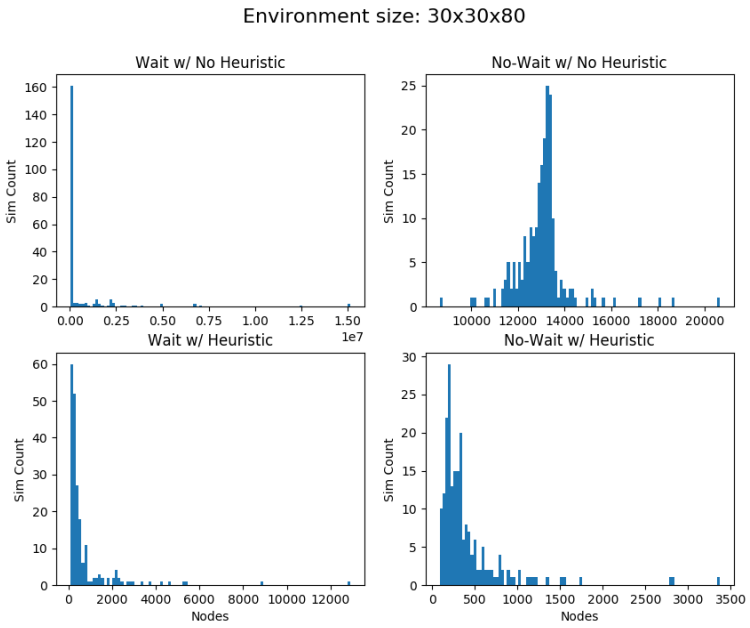


Figure 6.18: Expanding the environment from 10x10x40 (4000 space-time locations) to 30x30x80 (72000 possible space-time locations) maintains strong computation reductions for the search with heuristic. The number of nodes generated with the heuristic is orders of magnitude smaller.

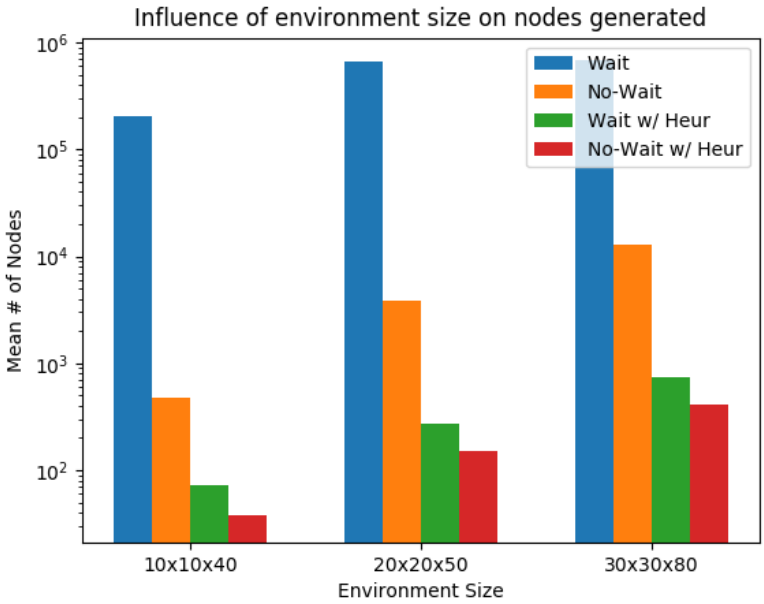


Figure 6.19: The scaling of computational reduction was explored with three increasing environment sizes which give increasing number of possible space-time locations: 4000, 20000, and 72000. In this graph, we compare the mean number of nodes generated for each environment size. Computation reduction is strongest when applying the heuristic to the Wait version of A. However, the heuristic still does not reduce the search complexity of waiting to the level of a No-Wait search.*

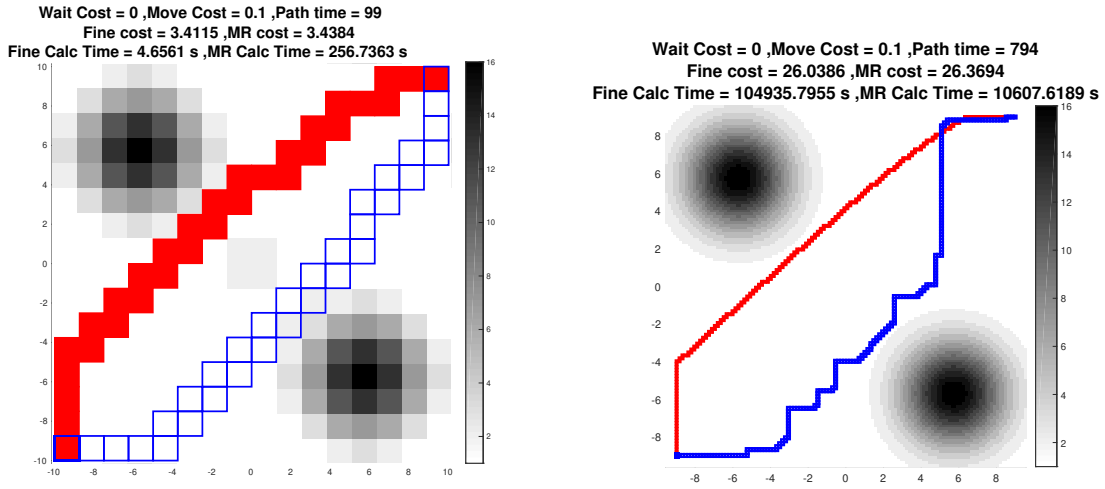
The prior full resolution examples also serve as a benchmark for the multiresolution with waiting examples.

6.3 Multiresolution Waiting vs. Non-waiting Solution

An investigation with the multiresolution representation was initially performed using a threat field with three parameters (i.e., $N_{\mathrm{P}} = 3$) as shown in Fig. 6.20, corresponding to three "peaks" of the threat field. The center peak decreases in magnitude over time, therefore the optimal path should wait as long as possible then traverse through the center of the field. This scenario was run at four different resolutions, $N_{\mathrm{R}} = \{16, 32, 64, 128\}$, corresponding to $D = \{4, 5, 6, 7\}$ and the results of these simulations are summarized in Table 6.1. Recall that the parameter D indicates the highest resolution considered in the multiresolution approximation. With $D = 6$ the total computation time for finding the optimal path with the multiresolution map (i.e. solving Problem 3) is of the same order of magnitude as that for finding the optimal path in the uniform resolution map ((i.e. solving Problem 2)). With $D = 7$, computation with the multiresolution map is an order of magnitude faster. This computational speed is achieved with suboptimality, i.e. the cost of the solution of Problem 3 is always greater than or equal to the cost of solution of Problem 2. However, with $D = 7$, the maximum such suboptimality observed across all cases was approximately 1%. Next, we discuss this suboptimality with lower values of D.

Table 6.1: Summary of cost of and calculation times

		Allow wait		No wait	
D		Uniform	Multires.	Uniform	Multires.
4	Path cost	3.41	3.43	3.57	3.88
	Calc. time	4.66 s	256.7 s	0.79 s	92.1 s
5	Path cost	6.57	6.64	7.39	7.42
	Calc. time	59.9 s	743.9 s	13.0 s	358.0 s
6	Path cost	12.91	13.04	14.72	14.72
	Calc. time	2305 s	2489 s	375.8 s	1284.2 s
7	Path cost	26.04	26.37	29.62	29.76
	Calc. time	104,940 s	10,600 s	21,835 s	5,978 s



(a) (b)

Figure 6.20: Waiting-allowed path with a multiresolution map of a three peak Gaussian field. Here, the center peak fades over time for $D = 4$ and 7 (a and b). The multiresolution path is in blue boxes and the uniform resolution path in solid red. Note the tenfold reduction in calculation time for the $D = 7$ multiresolution case.

In addition to the three peak field, the numerical study reported in the Section 6.2.1 was repeated for the multiresolution cases $D = \{4, 5, 6\}$. All parameters of the threat fields remain the same. A set of 100 simulations (in addition to the previous set of 2000 simulations) was run for each $D = \{4, 5, 6\}$, and each simulation included four calculations: waiting with uniform and multiresolution and non-waiting with uniform and multiresolution. The major results are summarized in Figs. 6.21 and 6.22.

Figure 6.21 describes the cost reduction of considering a path with waiting under the multiresolution approximation of the field. A significant difference from the uniform resolution results is the phenomenon of cases when the allowance for waiting is leads to paths with *higher* costs compared to no-wait paths. An explanation of this phenomenon is as follows. Based on the current position, the path planner expects that waiting is beneficial but ends up anticipating incorrectly due to the partial knowledge available at that location. However, increasing the D parameter reduces the number of such occurrences.

Fig. 6.22 shows the difference between uniform- and multiresolution cases when both path planners allow waiting, i.e., the difference in the cost of the solution of Problem 3 to that of Problem 2. Therefore, this result is an indication of the suboptimality incurred due to the vehicle-

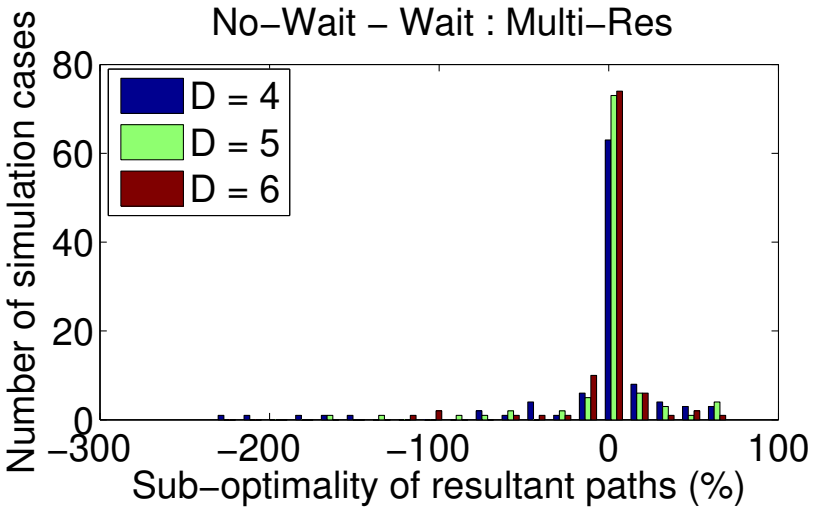


Figure 6.21: Percentage cost reduction when waiting is considered using the multiresolution representation at different resolution levels. Negative cost reductions (or sub-optimality of no-wait paths) indicate that the algorithm anticipated a benefit from waiting but was mistaken.

centric multiresolution approximation of the field. This suboptimality increases with increasing values of D. This observation is explained as follows. Allowance for waiting is made inside of the high-resolution window of the multiresolution approximation. As D increases, the area of this window is a smaller portion of the overall workspace, and therefore the multiresolution path planner is "myopic" and suboptimal.

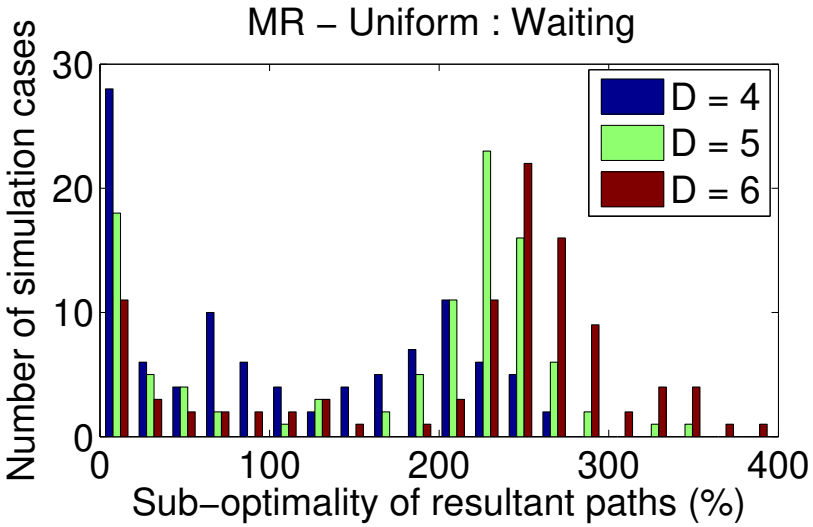
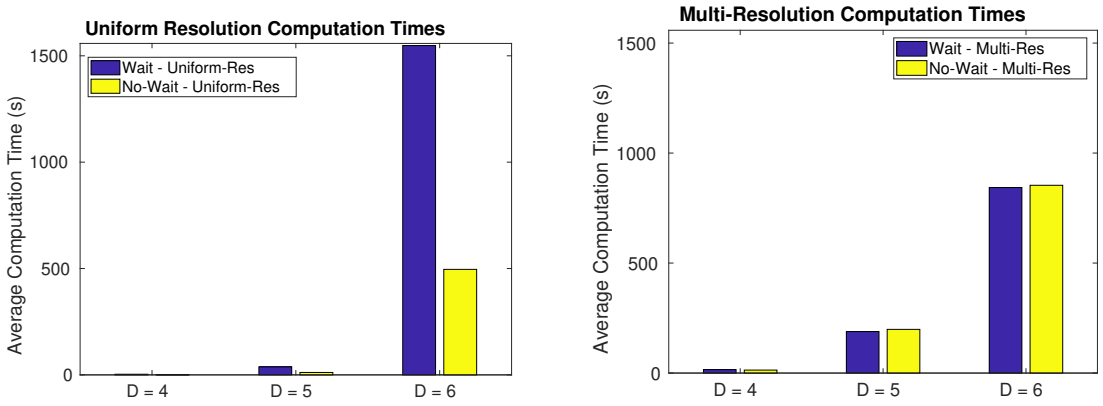


Figure 6.22: Suboptimality of using the multiresolution map, with different values of the parameter D.

Finally, we observe the average computation times for the four cases of waiting-allowed vs. no-wait paths and uniform vs. multiresolution maps, as summarized in Fig. 6.23. It is immediately



(a) (b)

Figure 6.23: Comparison of computation times. The computation time was measured and averaged for each case of waiting-allowed vs. no-wait paths, and uniform vs. multiresolution maps. Waiting-allowed and no-wait computations with multiresolution maps require equal computational time.

clear that the multiresolution map bears computational advantages as the resolution parameter D increases. Specifically, *the average computation time between waiting and non-waiting for the multiresolution cases is nearly equal.* This result was previously alluded to when describing the path planning algorithm in Fig. 4.2. Whenever the search algorithm explores waiting, the *spatial* multiresolution decomposition does not change. Therefore, in the multiresolution path-planning problem, the allowance of waiting does not incur significant additional computational expense.

6.4 Interactive Planning and Sensing for Time-varying Systems Results and Discussion

We demonstrate the IPAST algorithm with an illustrative example, describe key points during the progression of the algorithm through time, and discuss the convergence of the algorithm as a function of the variance of the estimated path cost $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$.

The numerical simulations reported in this section are all performed with a square workspace $\mathcal{W} = [-1, 1] \times [-1, 1]$ in nondimensional units. The threat field is constructed as in Section 5.1. The locations $\bar{\mathrm{x}}_n$ of maxima of the basis functions are assigned to chosen to provide uniform separation between them. As discussed in Section 5.1, the constants ν_n are chosen such that the

workspace is covered by the union $\cup_{n\in[N_{\rm P}]}\mathcal{R}_n^{\rm sup}$.

6.4.1 Illustrative Example

Figures 6.24 and 6.25 demonstrates the implementation of the algorithm on an illustrative example. In this example, the algorithm termination criterion, $\varepsilon_1(\mathbf{v}^*)$ is set with the constant $\lambda_1 = 25$ in (5.21). The numbers of grid points, parameters, and sensors are $N_{\rm G} = 400$, $N_{\rm P} = 9$, and $N_{\rm S} = 2$, respectively. The IPAST algorithm attempts to minimize the estimated path cost. To study the algorithm's typical behavior, Fig. 6.24 displays the evolution of the variance of the path costs,$\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ (top) and the path costs, $\mathcal{J}(\mathbf{v}_\ell^*)$, at each planning and reconfiguration iteration ℓ. To see the performance and convergence characteristics of IPAST, Fig. 6.24 shows also the true cost of the optimal path (both when planning starts at $t = 0$, and when replanning occurs at the current starting time of each iteration ℓ), and the *incurred cost*, which is true cost of the path found by the algorithm given $\mathbf{v}_\ell^*$.

Note the decreasing true optimal cost (with $t_{\rm start}(\ell)$) versus the true optimal with planning beginning at $t_{\rm start} = 0$. This is an indication that the optimal path includes waiting, and the replanned paths inadvertently benefit from the waiting required to reduce path uncertainty. As mentioned earlier, we restrict the current planning problem to non-waiting solutions and reserve the complication of waiting for future work.

For selected time instants $t = 1.2, 3.2, 4.45, 10.85, 12.80$ and 22.05, Fig. 6.25 shows the threat estimate $\hat{c}$ by a colormap, the candidate path $\mathbf{v}_\ell^*$ of minimum estimated cost by gray circles, the final path $\mathbf{v}_{\bar{\ell}}^*$ by white circles, and the sensor placement $\mathbf{s}_\ell$ by red rings. During execution of the final path, as in Fig. 6.25(f), the current position of the vehicle is shown as a small red circle. In Fig. 6.25, the gray regions are of significant support of bases with non-identified parameters, i.e., those ϕ_n where no sensor has been placed within $\mathcal{R}_n^{\rm est}$. Because the IPAS algorithm finds optimistic threat estimates, as discussed in 5.3, these gray regions indicate zero contribution from the basis functions to the estimated threat. Therefore, in Fig. 6.25(a) for example, the path $\mathbf{v}_1^*$ traverses through the gray regions.

The IPAST algorithm begins at time $t_k = 0$ with $N_{\rm S}$ initial sensor placements. The resulting

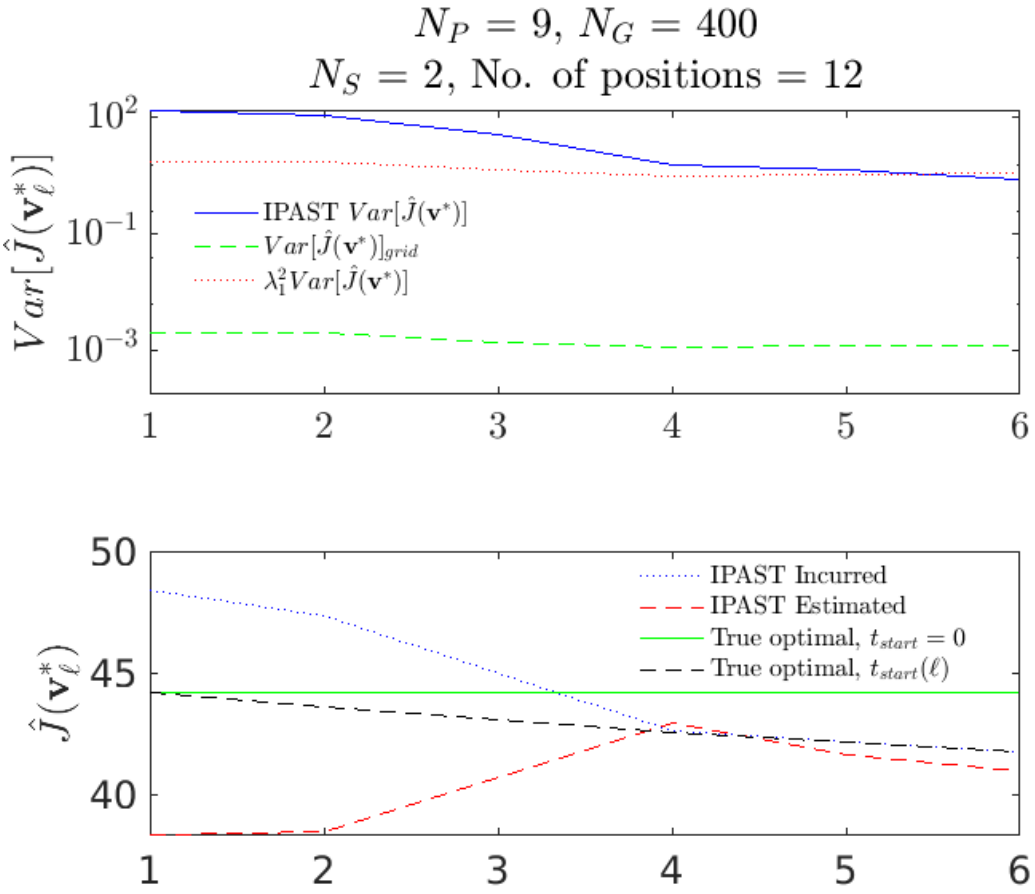


Figure 6.24: Path Cost and Variance of Path Cost behavior of IPAST algorithm.

estimated field from the initial measurements is shown in Fig. 6.25(a) as well as the initial path $\mathbf{v}_1^*$ given the current estimated field. In this example, the measurement collection time Δt_{c} is 0.2 seconds and the planning duration is $\Delta t_{\mathrm{p}} = 1$ second, therefore a planning phase is triggered at $t = 0.2$ and the plan is ready at $t = 1.2$. At this stage, a significant portion of the path lies in unknown regions and the path cost variance $\mathrm{Var}[\hat{J}(\mathbf{v}_1^*)]$ will likely be higher than the stopping criterion $\varepsilon_1(\mathbf{v}^*)$ triggering a sensor reconfiguration. The time required for sensor reconfiguration is set to $\Delta t_{\mathrm{r}} = 2$, therefore at $t = 3.2$ seconds, in Fig. 6.25(b), the next set of sensors and the updated field estimate are shown. The planning and reconfiguring phases continue in this fashion in Figs. 6.25(c)–6.25(e). Note that in Fig. 6.25(d) a true optimal path is found, however the stopping criterion has not been met. In this case, the previously measured regions have accumulated additional variance through the error covariance update step, in particular through the process noise. Consequently, previously sensed regions will be targeted again by the sensor reconfiguration pro-

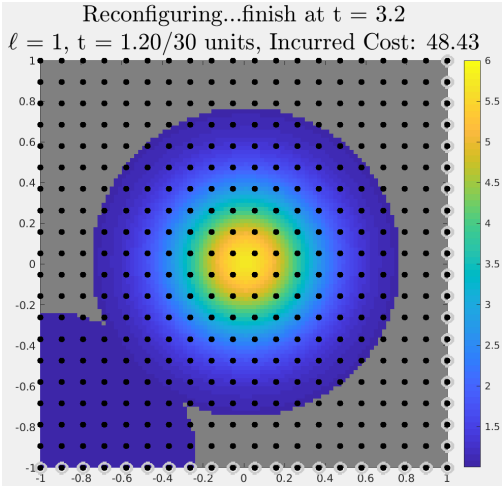
cedure as in Fig. 6.25(e). Finally, at $t = 17.2$, seconds a path meets the path cost variance threshold $\varepsilon_1(\mathbf{v}_\ell^*)$, is accepted as the final plan, and the vehicle executes the path as in Fig. 6.25(f).

6.4.2 Path Cost Variance Convergence Behavior

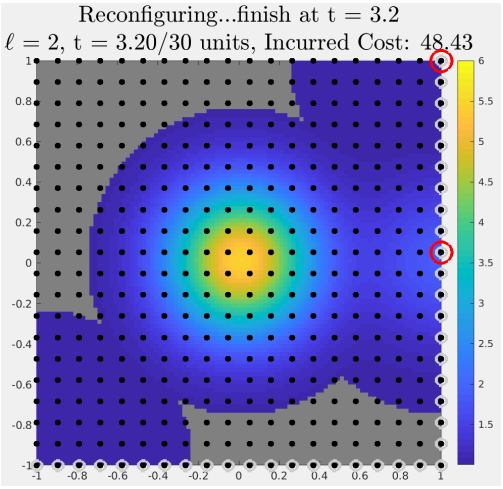
As a demonstration of the algorithm termination criterion and convergence properties and Proposition 2 discussed in Section 2.5, we investigate the state of the path cost variance $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ and stopping criterion $\varepsilon_1(\mathbf{v}_\ell^*)$ at every time instant t_k during the IPAST operation. Although the IPAST algorithm only checks the stopping condition after obtaining a new path, we can observe the full behavior of $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ for every loop through the Kalman filter. In Fig. 6.26 the path cost variance and stop threshold from the previous example are shown. For the first 10 seconds the path cost variance stays relatively large compared to the threshold value. During this time the candidate optimal path is typically moving from one unknown region to another, and significant variance remains in the parameter estimates and resulting path costs. After approximately $t = 10$, the regions in which the path lies have been measured and reaching the threshold $\varepsilon_1(\mathbf{v}_\ell^*)$ is contingent upon how quickly the measurements can counter the growth of the variance, as in approximately $t = 10 \ldots 17$. This is the scenario we refer to in Proposition 2. Note that the threshold is evaluated based on the current path $\mathbf{v}_\ell^*$ and may change depending on the route the path takes through the basis functions.

6.4.3 Convergence Behavior of the Final Path

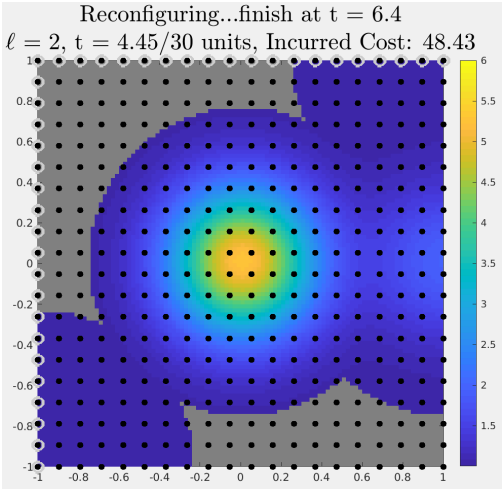
When defining the convergence condition in Proposition 2 we specifically state that the behavior is based on $\mathbf{v}_\ell^*$. This is for two reasons: (1) because our ultimate interest is in discovering and reducing the variance of the final accepted path, and (2) considering a fixed path makes the relationship between the $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ during reconfiguration, measurement, and planning more clear. Specifically, the circles in Fig. 6.26 denote the transition from $\mathrm{Var}[\hat{J}(\mathbf{v}_{\ell-1}^*)]$ to $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$. During this transition, a path $\mathbf{v}_\ell^*$ may pass through a different region with arbitrarily higher variance. In fact, because unmeasured regions are assumed to have parameter value of zero until measured, $(\hat{\theta}_0 = 0)$, paths will tend to pass through unknown regions in the initial iterations of the algorithm.



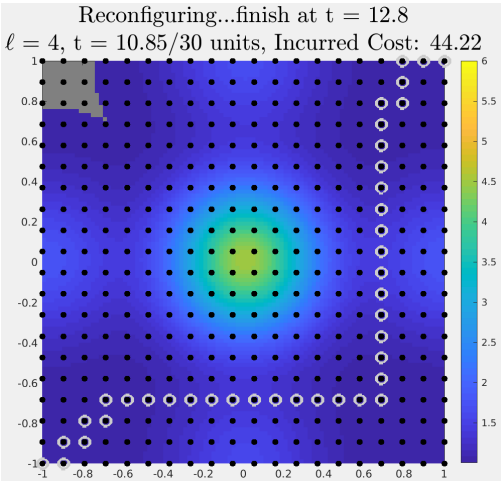
(a) First Plan.



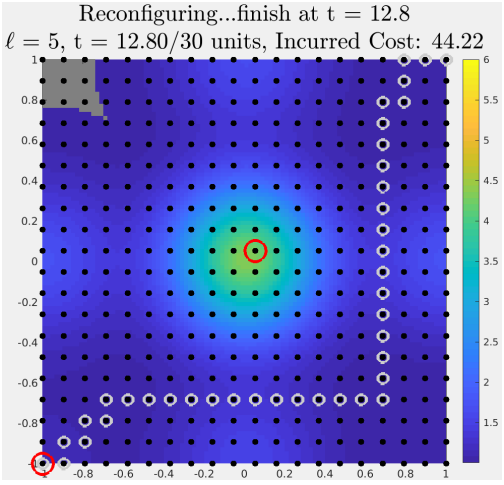
(b) Reconfiguration complete.



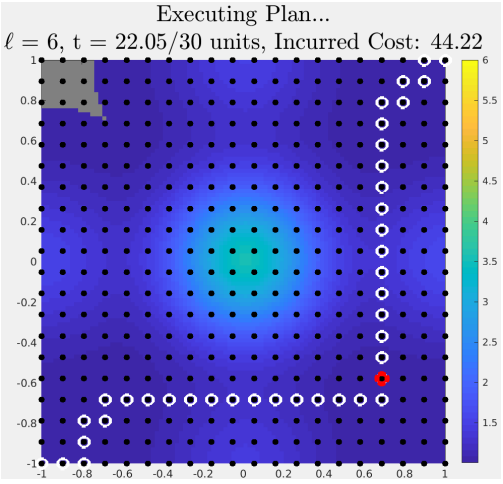
(c) Second Plan.



(d) True optimal found.



(e) Additional sensing.



(f) Executing final plan.

Figure 6.25: Visualization of the iterative and interactive planning and sensing for Time-varying fields (IPAST) process for $N_P = 9$ and $N_S = 2$.

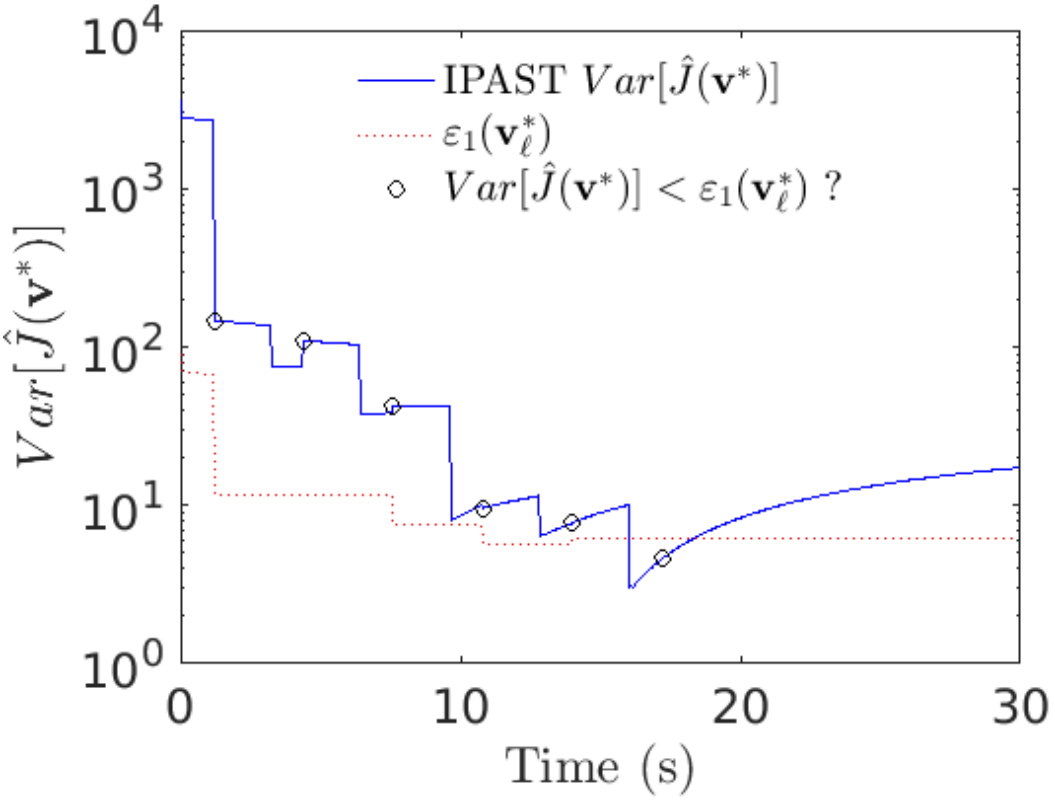


Figure 6.26: IPAST Path cost variance and stop threshold at every time instant.

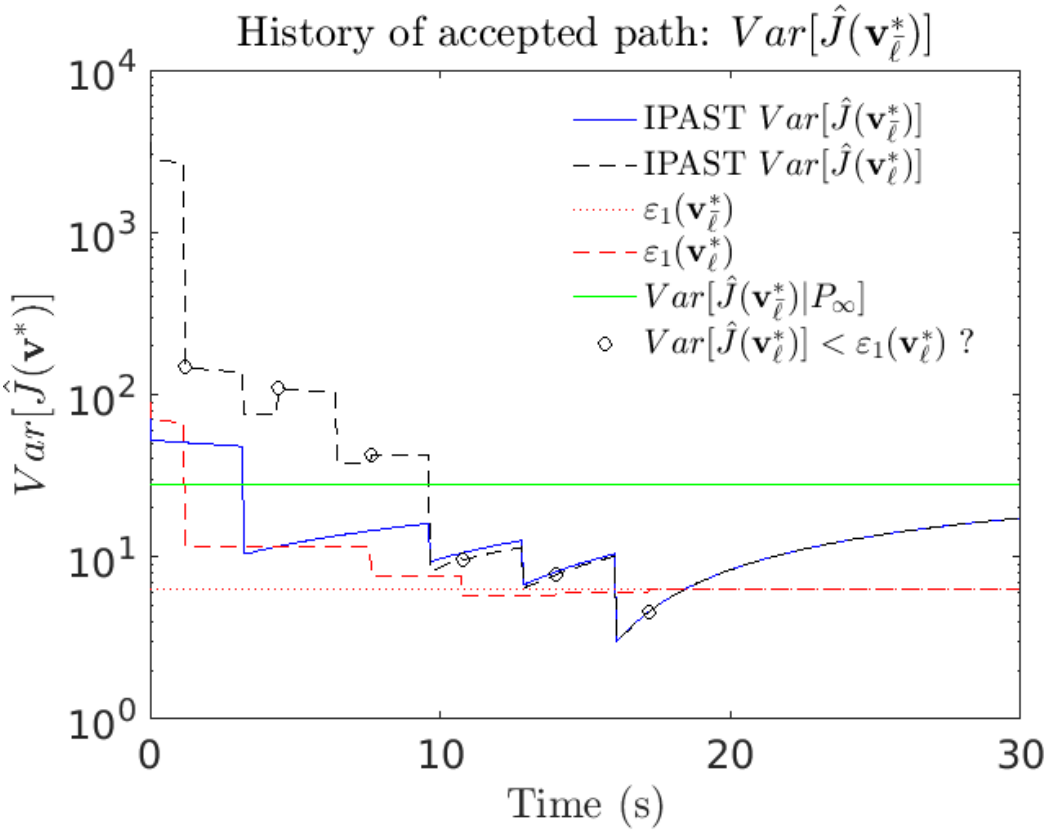


Figure 6.27: IPAST Path cost variance and stop threshold at every time instant for the final path as well as the steady state covariance P_∞.

By focusing on the behavior of the final path $\mathbf{v}_\ell^*$ we can see a more clear and consistent picture of the convergence behavior. In Fig. 6.27 the $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ is plotted alongside the $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$. From Fig. 6.27 it is more clear that the final path variance $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ is either targeted by sensors or converging on $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)|P_\infty]$. When the IPAST algorithm settles on $\mathbf{v}_\ell^*$, the past cost variances $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ and $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ will overlap and, if Proposition 2 holds, will converge.

6.4.4 High Parameter - Long Horizon Example

Finally, we illustrate the points made in Remarks 4 and 5 with an example with $N_{\mathrm{P}} = 100$, reconfiguration duration $\Delta t_{\mathrm{r}} = 10$, and planning duration $\Delta t_{\mathrm{p}} = 3$. This is sufficiently long that the error covariance of the Kalman filter achieves its steady state covariance P_∞ within 0.1%. In this extreme scenario, the relevant parameters are: $N_{\mathrm{P}} = 10, t_{\mathrm{final}} = 100, \alpha = 10^{-3}, Q \sim (0,1), R \sim (0,(0.1)^2), \Delta t_{\mathrm{r}} = 10, \Delta t_{\mathrm{p}} = 3, \Delta t_{\mathrm{m}} = 0.1, \Delta t_{\mathrm{c}} = 0.2, \Delta t_{\mathrm{kf}} = 0.05$ and starting

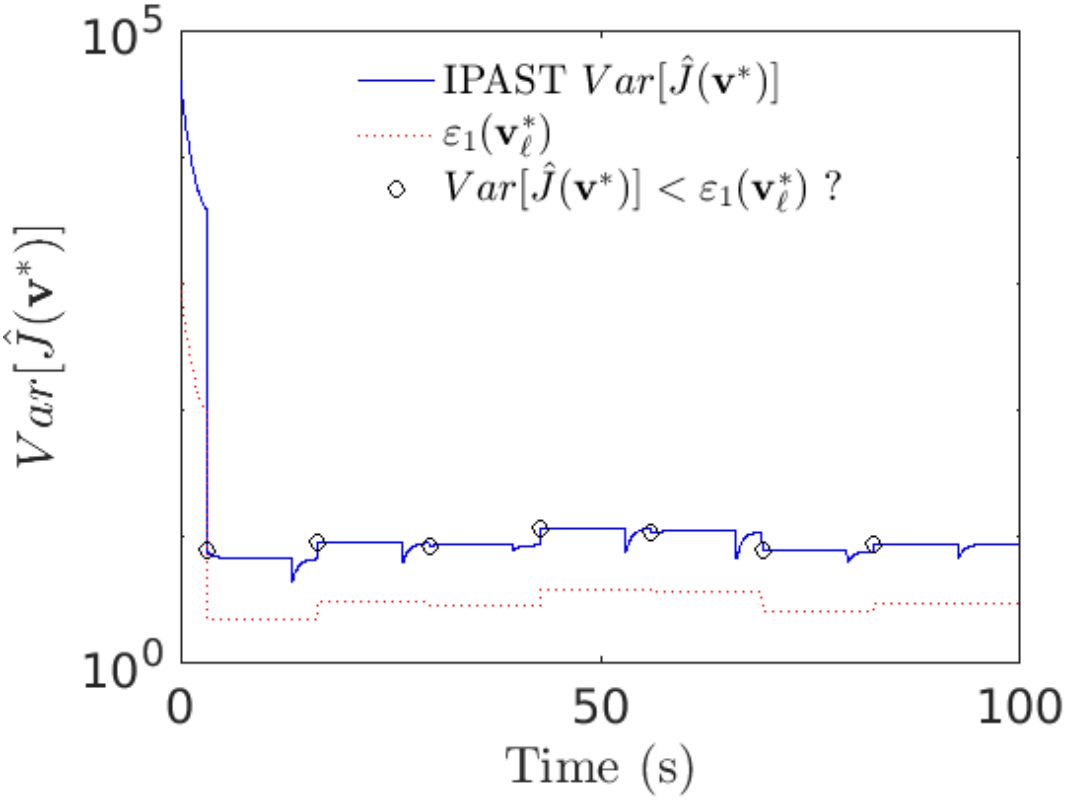


Figure 6.28: There are insufficient number of sensors to even reach $\varepsilon_1(\mathbf{v}^*)\ell$.

with $N_{\mathrm{S}} = 10$. Given these parameters, the steady state covariance P_∞ is achieved rapidly and is fully achieved during a reconfiguration phase. In Fig. 6.28, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ achieves $\mathrm{Var}[\hat{J}(\mathbf{v}^*)|P_\infty]$ during every reconfiguration phase. Furthermore, the small number of sensors is not sufficient to reduce $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ below $\varepsilon_1(\mathbf{v}_\ell^*)$ at any point. In Fig. 6.29, the number of sensors is increased to 25, however, the planning duration is too long and when a new path $\mathbf{v}_\ell^*$ is available for verification, $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ has already risen above the threshold. Providing significantly more sensors will not alleviate this problem. In Fig. 6.30, the planning duration is reduced to $\Delta t_{\mathrm{p}} = 0.15$, perhaps using a coarser map, or less sophisticated technique, and the algorithm is able to achieve convergence.

6.4.5 Waiting in IPAST Framework

In Section 6.4.1 it was noted that each subsequent path of iterations $\ell = 1, 2, ...$ had a lower cost than the original optimal path at $t_{\mathrm{start}} = 0$. This was an indication that waiting was beneficial in this environment. In this section we invoke the waiting based path-planning algorithm from

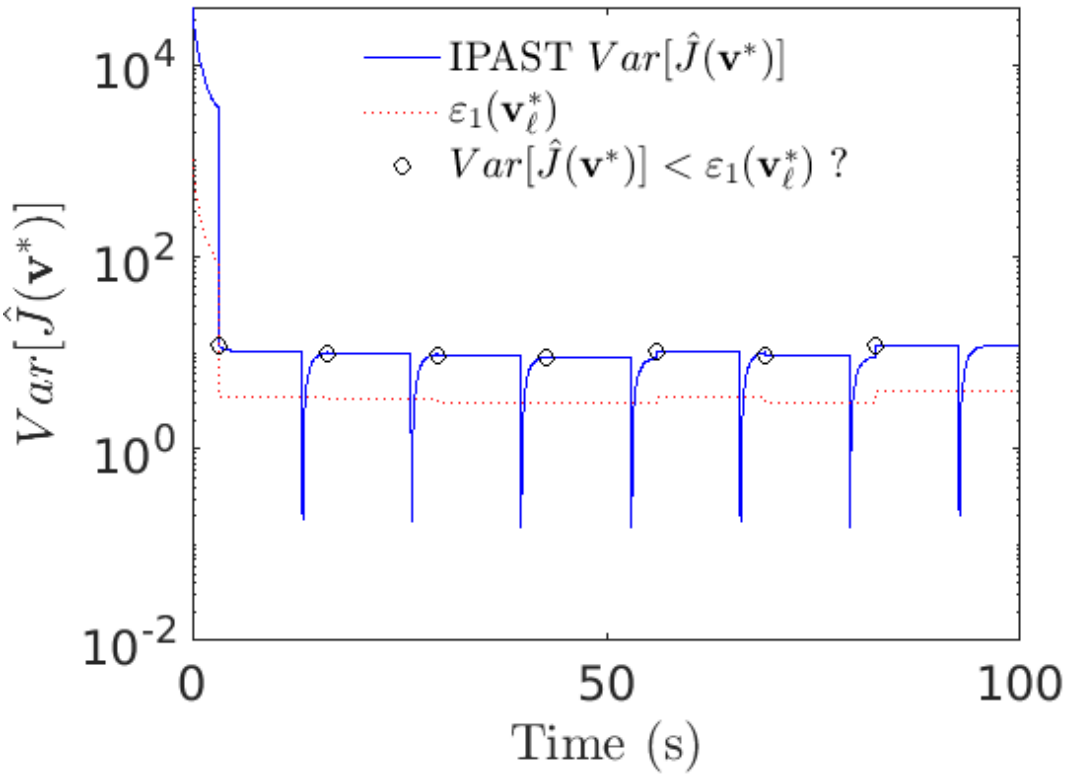


Figure 6.29: There are sufficient number of sensors, but the planning duration is too long before the variance threshold can be evaluated.

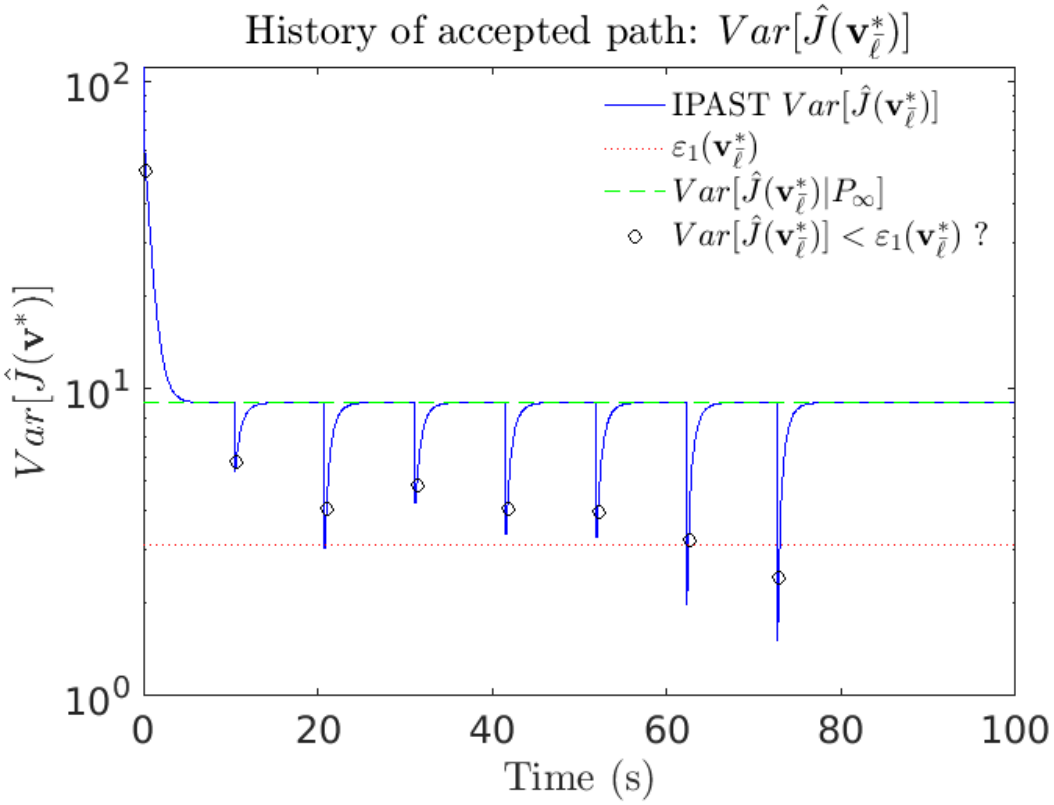


Figure 6.30: IPAST Path cost variance and stop threshold at every time instant for the final path as well as the steady state covariance P_∞.

Chapter 3 on an additional example to compare the non-waiting vs. waiting performance under the IPAST framework.

First, a minor modification is made to the threat field construction. Previously the threat field diffusivity α is constant and homogeneous over the entire field, in other words in the equation $\dot{\Theta}(t) = \alpha A\Theta(t)$, the diffusivity α is simply a constant. However, we let α be a diagonal matrix Λ with distinct diffusivity coefficients $\alpha_1, \alpha_2, \ldots, \alpha_{N_\mathrm{P}}$, then each basis function coefficient θ can have a distinct diffusivity value. This allows a non-uniform rate of decay for different regions of the environment. Physically, this could correspond to a region with a stronger heat "sink" due to either different materials or some active process drawing energy out of the field. This allows for some regions to decay faster in time than others.

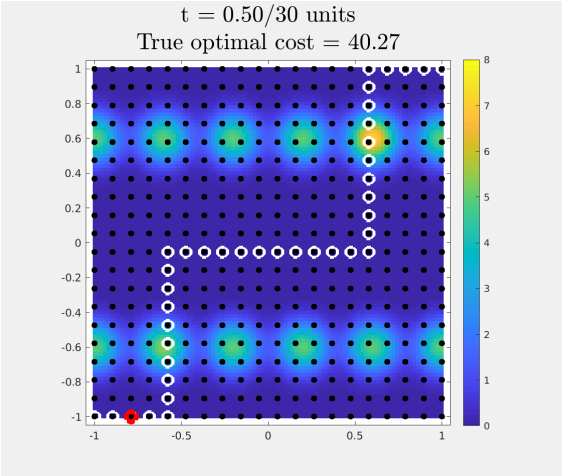
Based on the revised threat field construction,

$$\dot{\Theta}(t) = \Lambda \frac{\Phi^T}{|\Phi|^2} \nabla^2 \Phi \Theta(t) \tag{6.1}$$

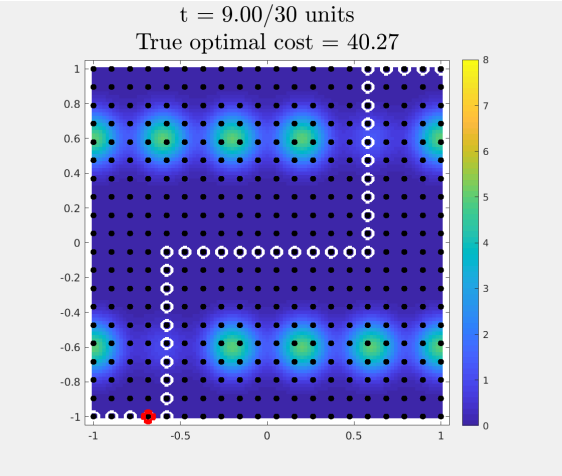
where $\Lambda = \mathrm{diag}(\alpha_1, \alpha_2, \ldots, \alpha_{N_\mathrm{P}})$, we can construct an example field as in Fig. 6.31. This field contains two "barriers" in which two "doors" open up after some amount of time. This diffusivity matrix Λ is defined such that a basis on the bottom left and basis on top right decay more quickly than the rest of the basis, allowing a low cost path to open up if the vehicle waits for enough time.

In Fig. 6.31, the solution for the true field (perfect knowledge) using a waiting algorithm is presented. The optimal path avoids exposure to the threats by first moving to a position equally far from the first two basis, as in Fig. 6.31(a). The vehicle waits at this position until $t = 9$ when the first basis along the path decays significantly in Fig. 6.31(b). It waits until $t = 17.75$ to move directly below the almost completely decayed threat in Fig. 6.31(c) and waits until the 2nd "door" threat in the upper right more fully decays. Then at $t = 21.5$ the vehicle moves all the way to the goal to arrive at $t = 29.75$, just before the mission time window closes. The non-waiting path immediately proceeds to the goal by traversing between the threats as best as possible, but still incurring additional cost over the waiting path.

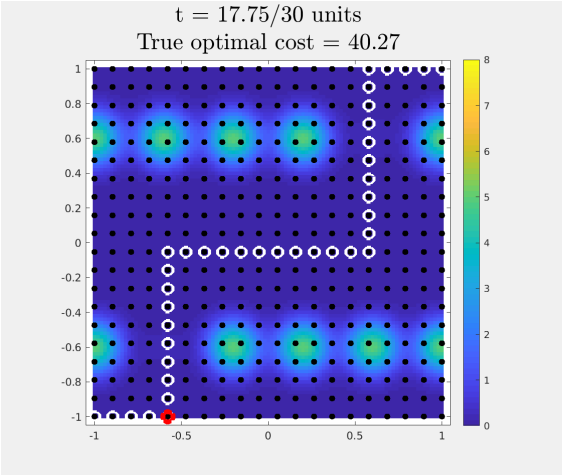
In Fig. 6.32, the resulting accepted IPAST path is shown. As expected from the IPAST framework, it is not necessary to uncover the full environment to find a nearly optimal path. The the



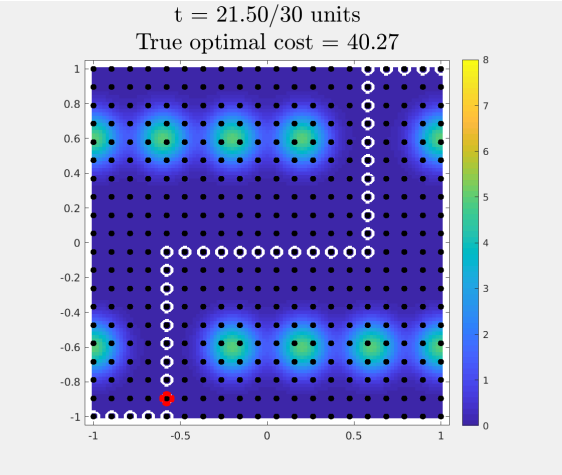
(a) Wait first position.



(b) Wait second position.

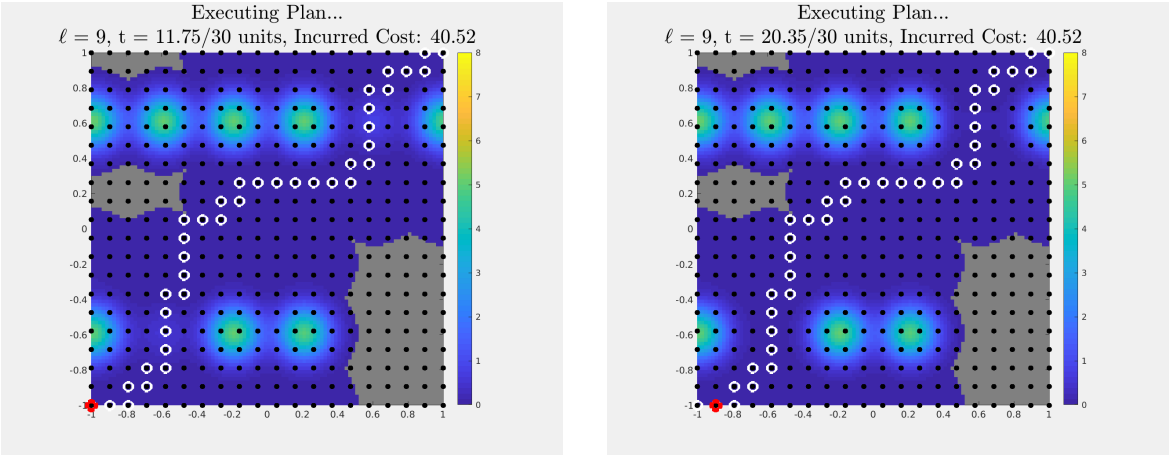


(c) Waiting third position.



(d) Start moving to goal.

Figure 6.31: Visualization of true optimal with waiting for non-uniform diffusivity threat field with $N_P = 36$ and $N_S = 4$.



(a) Final path accepted.

(b) First step after waiting.

Figure 6.32: Visualization of IPAST result with waiting for non-uniform diffusivity threat field with $N_P = 36$ and $N_S = 4$.

accepted path $\mathbf{v}_\ell^*$ is found and begins at $t = 11.75$. However, this plan includes waiting, and the first movement begins at $t = 20.35$ and the vehicle proceeds to the goal, arriving at $t = 29.6$, a bit before the mission time window closes. This demonstrates that the IPAST framework can additionally discover near optimal paths, even paths that include waiting as an optimal action.

Figure 6.33 shows the iteration history of the non-uniform diffusivity example for the variance of the estimated path cost $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ and the estimated path cost $\hat{J}$. While Figure 6.34 shows the history for the non-waiting algorithm. Two of the major differences to note are the history of the *True optimal* paths ($t_{\text{start}} = 0$ and $t_{\text{start}}(\ell)$), and the total iterations. In the waiting case, Fig. 6.33, the true optimal path cost at $t = 0$ and at later start times remains the same. Due to the consideration of waiting, the algorithm is always considering the best path, which includes waiting, at every new iteration. In contrast, in Fig. 6.34, the non-waiting algorithm is naive to waiting benefits, and the true optimal path cost gets better over time because the path is forced to start later and later by virtue of the iteration process. As to the second point, the waiting version terminates at $\ell = 9$ vs. the non-waiting version which requires $\ell = 11$ iterations. Although this is an isolated example a larger parametric study would be needed to confirm this trend, the intuition is that waiting algorithm already considers what the best path is for any starting time and therefore spends less time searching to verify other paths. Whereas, the optimal path for the non-waiting algorithm changes over time, and IPAST may spend time verifying a path which will be outdated

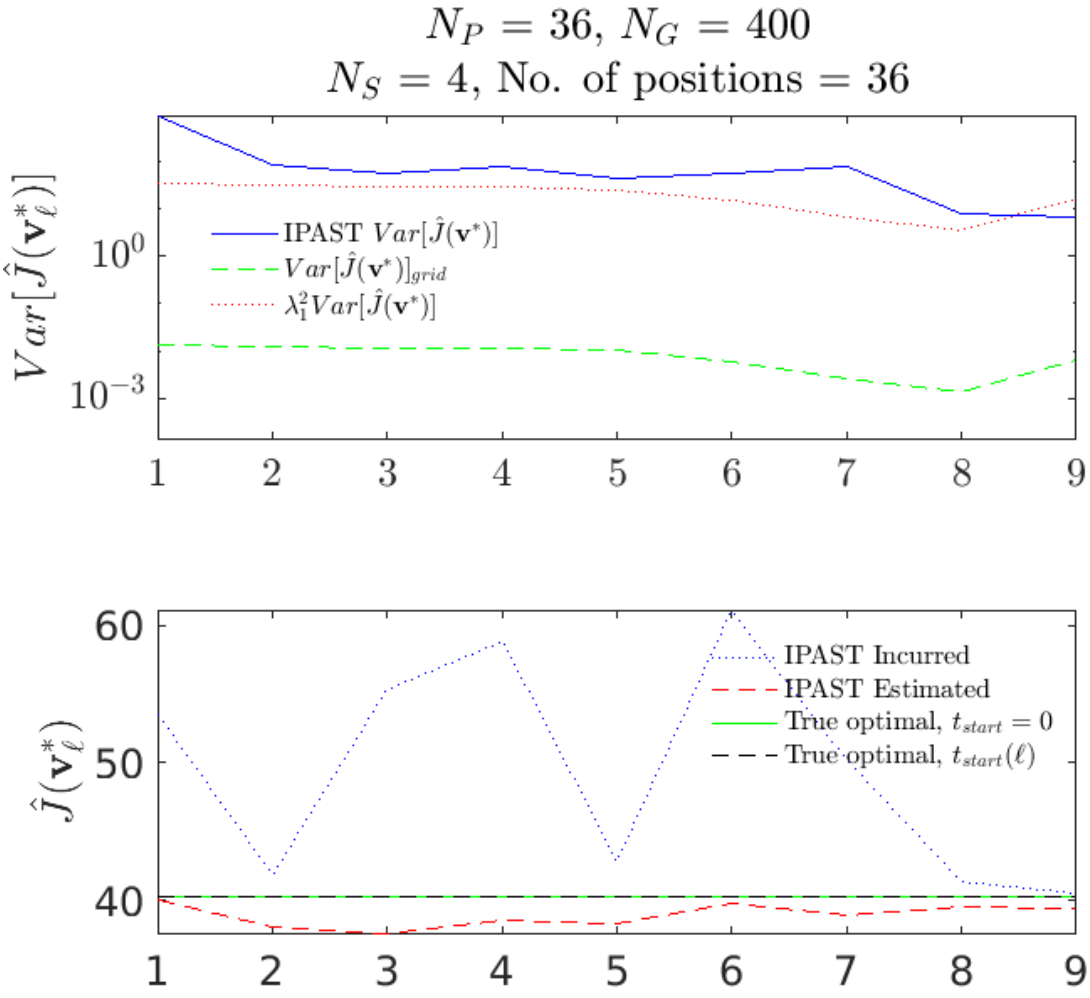


Figure 6.33: Variance of estimated path cost and path cost at each iteration of the IPAST process using the waiting algorithm for the non-uniform diffusivity field of $N_{\mathrm{P}} = 36$ *with* $N_{\mathrm{S}} = 4$.

in the next iteration.

Taking the final path cost and summarizing in Table 6.2 shows a few interesting points. First, the waiting based IPAST outperforms the non-waiting IPAST. This was desired due to the previous literature and results from Chapters 3 and 4, but perhaps not expected. However, it does follow from the intuition that if the planner considers all possible start times during each iteration it should converge to a better path than the non-waiting based IPAST. Note that the margin is not very large, this is due to the number of iterations of IPAST that were required to verify the path. If the variance threshold had been more forgiving, IPAST would have terminated early with a worse path cost. Second, the non-waiting IPAST performs better than the non-waiting path planner with full

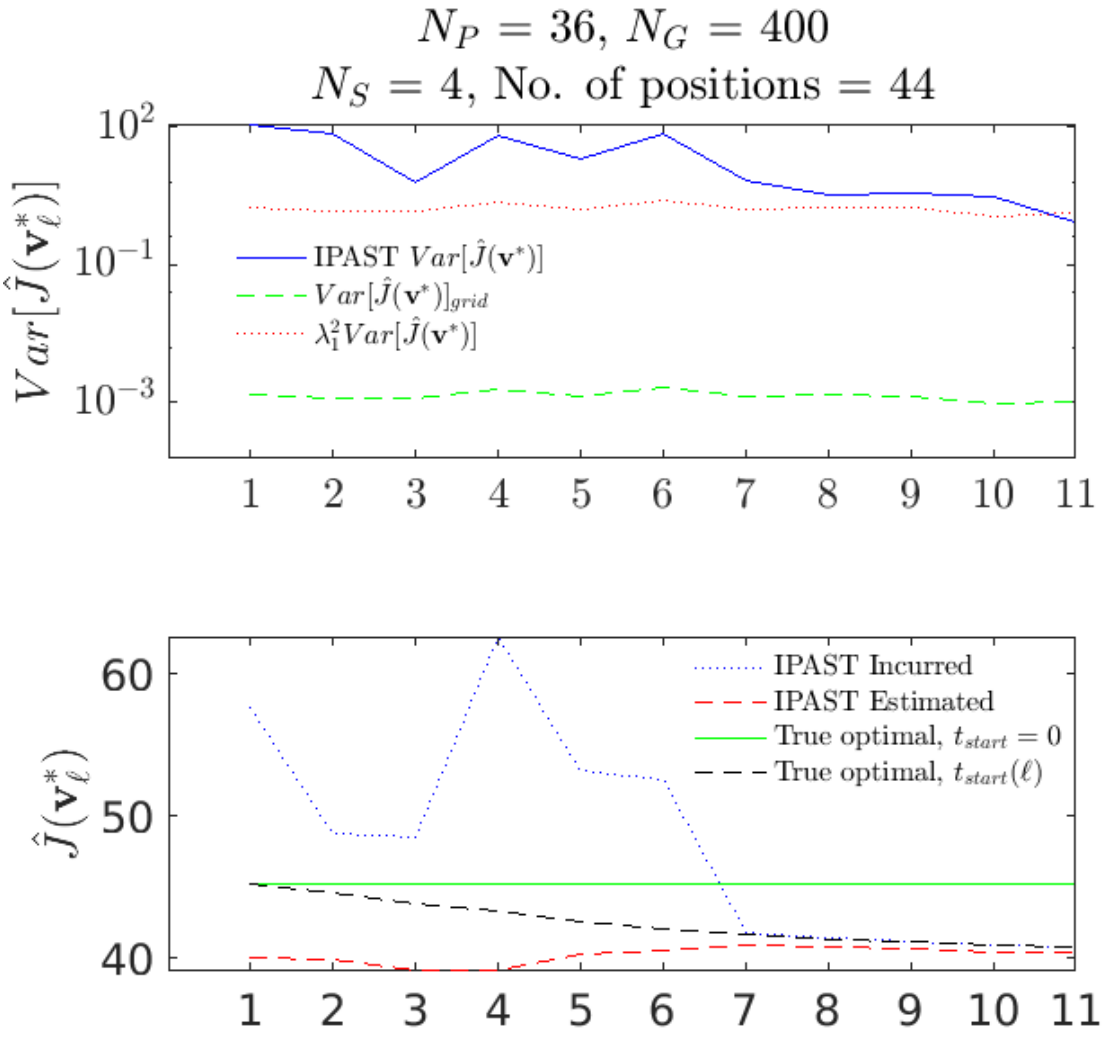


Figure 6.34: Variance of estimated path cost and path cost at each iteration of the IPAST process using the non-waiting algorithm for the non-uniform diffusivity field of $N_{\mathrm{P}} = 36$ *with* $N_{\mathrm{S}} = 4$.

Table 6.2: The waiting IPAST achieves lower cost over the non-waiting IPAST. Additionally, IPAST achieves lower cost over the true non-waiting path in either waiting or non-waiting versions.

	Waiting	Non-waiting
True	40.27	45.3
IPAST	40.52	40.8

information (True, Non-waiting cell). This relates to the first point, that the non-waiting version of IPAST benefits from the fact that it must wait several iterations to converge on path with acceptable path cost variance.

The conclusion returns back to the original problem discussed in Chapters 3 and 4, the tradeoff between choosing the optimal but computation heavy waiting planner or the sub-optimal but fast non-waiting planner.

Remark 6 (Environments with stable dynamics)**.** Waiting tends to be beneficial in environments with stable dynamics, such as environments which decay over time. If, in fact, the environment has some unstable dynamics, in which things get worse over time, the forced waiting that occurs in IPAST may lead to a higher path cost. However, the behavior of the environment is unknown a priori, as well as the corresponding optimal path, until measurements are made. Therefore, IPAST may be the best option in either case. An avenue of future work is to evaluate the potential that the environment will get worst and the vehicle should begin moving toward the goal on a sub-optimal path with the intention that additional measurements from the sensor network will better inform the vehicle as the environment is updated. It may be necessary to immediately vacate the start position, but then wait in some other region for the most optimal path.

Chapter 7

Conclusions and Future Works

In this thesis, we formulated and discussed time-varying path-planning algorithms under two different sensor constraints, limitations on measurement *resolution* and *uncertainty*. With regard to sensor measurement resolution, we presented a multiresolution time-varying path-planning algorithm which explicitly considers waiting. For the topic of measurement uncertainty, captured by sensor noise, we developed a iterative path-planning algorithm valid for static and time-varying environments, coined as Interactive Planning and Sensing (IPAS) and Interactive Planning and Sensing for Time-varying fields (IPAST) respectively.

We addressed the noisy sensor measurements problem using a sensor network to both converge to the optimal path in an unknown environment and provide a guarantee on the optimality of that path. Begining with a static environment, we developed the IPAS framework which iteratively identifies regions relevant to the path-planning problem, updates an environment map, and uses the updated path to identify the next relevant region for measurement. We define this process as *task-driven* sensor placement which is implemented in the *sensor reconfiguration* algorithms. In our task-driven approach, sensor reconfiguration prioritizes regions which have direct influence on the current path and high uncertainty as characterized by their estimated threat coefficient variance. We compare task-driven sensor placement with *information-driven* sensor placement approaches and find that the IPAS task-driven framework optimizes path planning performance while minimizing the area of the map needed to measure. A valuable finding with regard to resource allocation was that the mission planner can trade off sensor resources for path convergence time with an exponentially decaying relationship, i.e. for a few more sensors, the path can converge exponentially faster.

For the sensor with resolution limits, we expressed a vehicle-centric multiresolution environment in which *waiting* for finite intervals of time may provide the minimum cost exposure plan

to a time-varying threat field. First, it was demonstrated that waiting can provide meaningful path cost reductions, up to 25% in our studies, in the uniform environment representation. The path optimality comes at the cost of increased computational expense, and a pruning method as well as a study of path search heuristics demonstrated reduction in computation time while preserving optimal paths. Next, the multiresolution environment representation was applied to the waiting path-planning algorithm and we found that the vehicle-centric multiresolution approximation of the threat still allowed the discovery of beneficial waiting paths. Importantly, we observed that including waiting in to the multiresolution planner has a negligible computational expense.

Finally, we extend the IPAS results to the time-varying environment problem, and refer to this solution as IPAST. This is accomplished by applying a Kalman filter to the time-varying threat coefficients. The iterative nature of the IPAS algorithm remains in IPAST as well as the guarantees on convergence and optimality, with some caveats. There must be sufficient time in the mission window to allow for enough iterations to converge on a confident path. A mission planner can influence convergence in a number of ways, the most relevant including allowing higher risk (variance) in the accepted path, or allocating more sensor resources to speed up the convergence of the path, similar to the IPAS results. As a final connecting link, we briefly explore the behavior of IPAST with respect to waiting. It was seen that a waiting path-planning algorithm within the IPAST framework yields a lower cost path while maintaining the convergence properties of IPAST. This final result ties together the concepts of Chapters 2, 3, and 5.

After covering the IPAST work, the time is appropriate to look at Interactive Frameworks for Unified Time-varying Replanning or IFUTR work.

7.1 Generalizing the Environment

A limitation of the current IPAS frameworks is the form of the environment model. We assume that the threat field is composed of evenly distributed basis functions which collectively cover the entire workspace, are approximately disjoint, and their location ($\bar{\mathsf{x}}$) and shape (σ) are known. These assumptions allow for very clear identification of relevant basis functions and the use of efficient linear regression making the problem very tractable. Furthermore, as the density of the

basis increases over the workspace the model of the environment has the universal approximation property.

However, a more general version would allow for significant overlap of basis functions. The disjoint basis environment ensures that estimates of parameters made from measurements taken within that basis are independent from estimates of other parameters. This also avoids the under-determined estimation case where a single measurement is used to try to obtain estimate of two coefficients of basis that are adjacent and overlapping the sensor. This probably can be addressed through standard regularization techniques that minimize the norm of the estimates in some way such as Ridge regression and LASSO. However, the IPAS algorithm relies on characterizing the uncertainty of the path cost by utilizing the variance of the coefficients. When using regularization techniques, there are various methods to calculate the parameter estimate variance, but no standard method and the variance obtained may not reflect the true variance of the parameter.

Two issues that leads into the solution of the next subsection is the accuracy of modeling any arbitrary function. If the location and shape of the basis functions are specified, then any function can be approximated with arbitrary precision using an increasingly higher density of basis functions approaching infinity. If there is an upper limit on the number of basis functions, one could also allow the location $\bar{\mathrm{x}}$ and shape σ to be free to estimate, but this can cause the estimation procedure to be analytically intractable and possibly computationally expensive (see future work section on nonlinear extension). However, both of these issues are addressed by Gaussian processes.

7.1.1 Gaussian Processes

In some sense Gaussian processes are the limiting case when there are an infinite number of basis functions and parameters to fit. One way to consider a Gaussian process is as defining a distribution over functions where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$, is the covariance function, as in

$$f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), k(\mathbf{x}', \mathbf{x}')). \tag{7.1}$$

With this representation, one can take get an estimate at some location $\mathbf{x}_i$ as well as the variance of the estimate at that location, $k(\mathbf{x}_i, \mathbf{x}_i)$. See (Williams and Rasmussen, 2006) for detailed introduction and theory on Gaussian processes for regression. An example of the resulting mean and variance functions for a Gaussian process regression are shown in Fig. 7.1. The true function being measured and estimated is $y = \sin(3x)$ and it can be clearly seen that around the measurement locations the Gaussian process does a good job representing the function and giving a level of confidence as indicated by the shaded regions which is two standard deviations above and below the mean function. Note that in regions far from the measurements the function assumes a zero mean value and has high variance. This appropriate characterizes the knowledge available about those regions. But more importantly, for an extension to the IPAS work, this is exactly the type of assumption we make about unknown regions in the map, they are zero valued (optimistic) with high variance.

A straightforward application of Gaussian processes to IPAS can be outlined:

1. Obtain environment function estimate using Gaussian process regression given the current measurements.
2. Discretize the environment function and plan a path $\mathbf{v}_\ell^* = v_1, v_2, ...v_G$.
3. Evaluate the variance of the estimated path cost $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)]$ where $Var[\hat{c}(\mathsf{x}_k)] = k(\mathsf{x}_k, \mathsf{x}_k)$. and compare with stop threshold $\varepsilon_1(\mathbf{v}_\ell^*)$.
4. If path not converged, place N_S at path vertices $v_1, v_2, ...$ with the highest variance as evaluated at $\mathsf{x}_{v_1}, \mathsf{x}_{v_2}, ...$
5. Get new measurements to update environment Gaussian process. Return to Step 1.

The disadvantage of this approach is that the Gaussian processes can scale more poorly with data than the previous linear regression approach. However, if the environment required an increasingly large number of basis functions to be accurately represented, the Gaussian process regression may be preferable. However, Gaussian processes have already been applied to the multi-sensor environment modeling problem in (Erickson et al., 2015) so the advantages may outweigh any drawbacks.

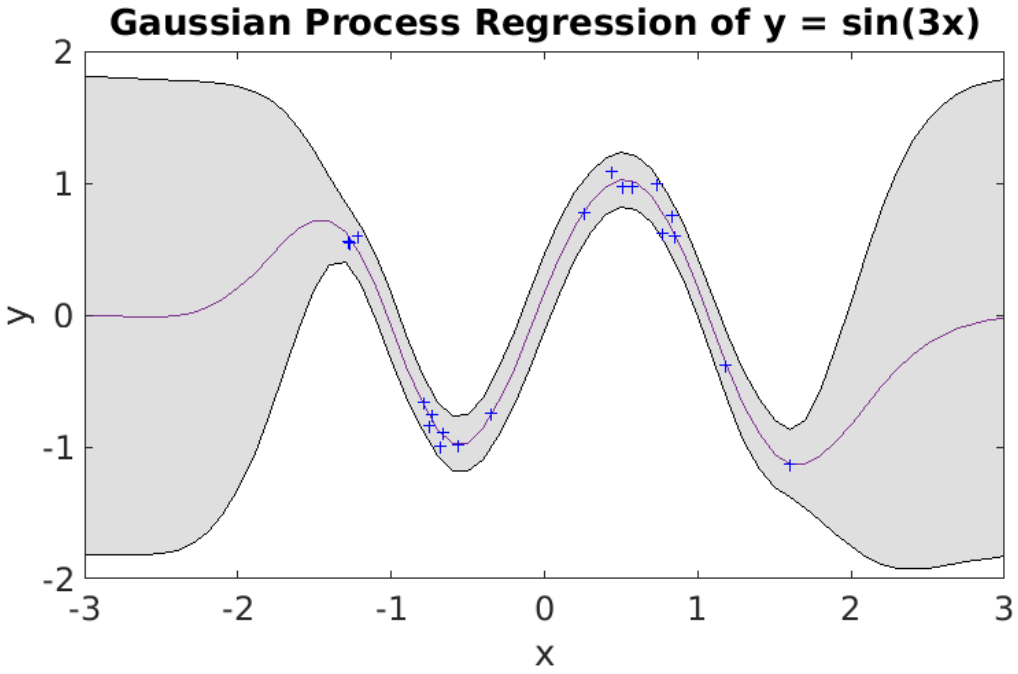


Figure 7.1: The resulting mean function and two standard deviations of variance. Measurement points indicated as blue crosses. Generated using GPML Matlab Toolbox, available at http://www.gaussianprocess.org/gpml/code/matlab/doc/

7.1.2 Transport Phenomenon or Convection-Diffusion Process

For the time-varying extension of IPAS, we considered a field governed by the heat diffusion equation

$$\frac{\partial c}{\partial t} = \alpha \left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} \right) \tag{7.2}$$

with the series solution as a sum of Gaussian functions,

$$c(\mathsf{x}, t) = \sum_{n=1}^{N_\mathrm{P}} \theta_n(t)\phi_n(\mathsf{x}) = \Phi(\mathsf{x})\Theta(t). \tag{7.3}$$

We related the rate of change in the field approximation to the rate of change of the parameter:

$$\frac{\partial c}{\partial t} = \Phi(x, y)\dot{\Theta}(t). \tag{7.4}$$

along with the two spatial derivatives, to obtain a representation of the heat diffusion process

as

$$\Phi\dot{\Theta}(t) = \alpha\nabla^2\Phi\Theta(t). \tag{7.5}$$

The ∇^2 is the Laplace operator which arises to describe the spatial derivatives of the Gaussian bases Φ.

However, many interesting phenomenon include a convection or transport component, particular when some flow field is involved. In this case, we process of interest would have the form

$$\frac{\partial c}{\partial t} = \alpha\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right) - \left(v_x\frac{\partial c}{\partial x} + v_y\frac{\partial c}{\partial y}\right) \tag{7.6}$$

where $\mathbf{v} = (v_x, v_y)$ are the velocity components of a flow field.

With the same basis approximation as in the diffusion equation, we would arrive at

$$\Phi\dot{\Theta}(t) = \left(\alpha\nabla^2\Phi - \mathbf{v}\cdot\nabla\Phi\right)\Theta(t) \tag{7.7}$$

as the representation of the transport process using the basis approximation. Some immediate concerns are the stability of the transition matrix $(\alpha\nabla^2\Phi - \mathbf{v}\cdot\nabla\Phi)$ of $\Theta(t)$, and modifications to the sensing strategy.

7.2 Sensor Reconfiguration Cost

In this work, we acknowledge but do not currently address the inclusion of sensor reconfiguration costs. This cost refers to the cost of physically relocating a team of mobile sensors to execute the iterative sensor placement provided by the proposed algorithm. Therefore, this cost captures the time and/or fuel spent in physically moving the sensors, while satisfying the kinematic and dynamic constraints of the mobile sensors. To this end, in the future, results from the field of multi-vehicle path planning may be leveraged, cf. (Bullo et al., 2009) and (Egerstedt and Hu, 2001).

In the illustrative example of Section 6.1.1, it is clear that the IPAS algorithm does not place

all N_{S} sensors in each iteration. Therefore, the problem arises of identifying *which* sensors should be physically moved to which new location. This issue can be formulated as a task assignment problem, which is also addressed in the literature, cf. (Choi et al., 2009).

7.3 Multiresolution Implementation of IPAS and IPAST

In chapter 1, we expressed that a major theme of this work was formulating the path planning problem given some *level* of information where the level was associated to two fundamental properties of sensor limitations: *resolution* and *noise*. The resolution limitation was explored in chapter 4 in which a multiresolution representation of the environment was used, but the value of the threat in each cell was known and noiseless. Then, the noise limitation was explored in chapters 2 and 5 resulting in a full resolution environment, but with uncertainty in the threat cost. Another formulation to explore is the combination of both sensor limitations of resolution and noise.

One way to approach this would be to take the multiresolution representation and each successively larger cell size would have proportionally higher noise variances. Another variant of multiresolution implemented with the IPAS framework would be to have a vehicle-centric multiresolution representation of the map where every mobile sensor in the sensor network contributes to the high resolution regions of the map. Any region in which a sensor takes a multiresolution measurement maintains that high resolution information which decays into lower resolution further away from the measurement location. Visually, this would look like 'pockets' of high resolution regions dotting the map. The same IPAS principle would hold where high resolution (low uncertainty) regions of the map are only produced where the optimal path should lie.

7.4 Nonlinear Estimation for IPAS

Throughout this work, the state dynamics and measurement models have all been linear. The threat field which was central to the IPAS algorithm with static threats and the IPAST extension to time-varying fields was a linear combination of Gaussian basis functions. Furthermore, we were

interested in estimating the coefficient θ which was linear in the basis functions, $c(\mathsf{x}) = \theta\phi(\mathsf{x})$. However, in an effort to generalize the applicability of IPAS, we may be interested in estimating the basis shape and locations parameters, $\sigma_i, \bar{\mathsf{x}}$. In the case of IPAS and the static field, the least squares approach can be replaced with non-linear least squares approach in the usual way. Define the error term,

$$\varepsilon_k = z_k - c(\mathsf{x}_k) = z_k - \theta_i\phi_i = z_k - \theta_i \exp(-\frac{(\mathsf{x}_k - \bar{\mathsf{x}}_i)}{2\sigma_i^2}) \tag{7.8}$$

for the k^{th} measurement of the i^{th} basis function ϕ_i.

Then take the sum of squares approach and define function

$$f(\theta_i, \sigma_i, \bar{\mathsf{x}}_i) = \sum_{k=1}^{N} \varepsilon_k^2 = \sum_{k=1}^{N} \left[z_k - \theta_i \exp(-\frac{(\mathsf{x}_k - \bar{\mathsf{x}}_i)}{2\sigma_i^2}) \right]^2 \tag{7.9}$$

for N measurements which can be minimized using various solvers. Alternatively, the partial of derivatives of $f(\theta_i, \sigma_i, \bar{\mathsf{x}}_i)$ can be found,

$$\frac{\partial f}{\partial \theta_i} = 2\sum_{k=1}^{N} \varepsilon_k \frac{\partial \varepsilon_k}{\partial \theta_i}$$
$$\frac{\partial f}{\partial \sigma_i} = 2\sum_{k=1}^{N} \varepsilon_k \frac{\partial \varepsilon_k}{\partial \sigma_i}$$
$$\frac{\partial f}{\partial \bar{\mathsf{x}}_i} = 2\sum_{k=1}^{N} \varepsilon_k \frac{\partial \varepsilon_k}{\partial \bar{\mathsf{x}}_i}$$

and a Gauss-Newton method can be used.

The nonlinear least squares problem could be solved repeatedly in as a batch each time a new measurement is received, however a recursive version would be advantageous. One recursive implementation can be found in (Alessandri et al., 2007). Of course, this assumes knowledge of the number of basis functions present in the field. And the question of which measurement below to which basis function if the location and width of the basis are unknown. At this point, it may be more appropriate to consider this type of threat field as a target identification and tracking problem, of which there is much literature (Bar-Shalom and Li, 1995; Martinez and Bullo, 2006;

Olfati-Saber and Sandell, 2008; Reid et al., 1979). Alternatively, it may be better to approach it as a Gaussian Process problem as mentioned previously.

7.4.1 Uncertainty Propagation

For the case of the time-varying system in which we employ the IPAST algorithm, the natural step for a nonlinear system would be to apply the extended Kalman filter (EKF). As has been the experience, the linearizion of the dynamics and/or measurement models may work well in practice. However, the IPAST method takes advantage of the fact that estimates from the Kalman filter remain Gaussian the the propagation and update stages. This makes calculation and interpretation of the variance of the estimated path cost $\mathrm{Var}[\hat{J}(\mathbf{v}^*)]$ fairly straightforward. In contrast, the EKF is not an optimal filter and the error covariance matrices obtained don't represent the true covariance of the estimates. The formulation and of $\mathrm{Var}[\hat{J}(\mathbf{v}^*)]$ would need to be evaluated. The general field of Uncertainty Quantification may hold many salient discussions and solutions to the issue of appropriately characterizing the uncertainty of estimates, see (Adurthi, 2016) which focuses on nonlinear filtering and uncertainty quantification within the realm of dynamic sensing.

7.5 Decentralized Sensor Network

Although never explicitly stated, the sensors considered in the IPAS and IPAST problems are considered part of a sensor network in which local information, the measurements of each sensor, is available globally. The estimates of the threat coefficients are determined in a *centralized* manner. This could be a result of all sensors having a communication link with every other sensor, or that all sensors transmit measurements to a centralized aggregator or base station. In the IPAS framework, the natural architecture is that the planning vehicle, referred to as the *actor*, is the base station. This is the most straightforward approach because the actor both needs the sensor information to update the map, and must command the sensor network where to move next.

However, much work has been done on decentralized estimation using sensor networks using the notion of estimate *consensus* (Olfati-Saber and Murray, 2004) including under the formula-

tion of the Kalman filter (Olfati-Saber, 2005, 2009; Olfati-Saber and Shamma, 2005). There is even work to estimate an environment model of the same summed basis form as presented in this thesis (Lynch et al., 2008).

If the size of the sensor network becomes fairly large, relying on a centralized and global estimation scheme may become impractical. The estimation part of the IPAS method would follow in a straightforward manner from the previous literature, all agents including the actor would reach a consensus on the threat estimates and the actor can update the map and plan accordingly. The actor must still command the rest of the sensor network where to measure next. These commands could be relayed across the network with each movement command containing and ID for the appropriate sensor agent. Alternatively, each agent in the system could perform it's update its own map and path plan, and identify the next location accordingly. Since each agent achieves a consensus on the map, each agent will have consensus on the planned path and identification of the next basis to measure. However, if the sensing agents are expected to be small with minimal computation resources, the map update and planning may be too much burden for the light platform.

7.6 Limited Sensor Trajectory Control - Parachute Drop

Consider the problem of accurately dropping a package by parafoil or parachute. In this scenario there is a target location on the ground in which a valuable package dropped from an aerial vehicle by either parafoil or parachute. In the case of a round canopy parachute, the trajectory (and final touchdown location) are completely dependent on the dynamics of the wind environment. The more wing-shaped parafoil may have some limited control over it's trajectory, but is still largely subject to the behavior of the wind environment. In this case, the initial position of the package release point and the accuracy of the wind environment play critical roles in the trajectory and final touchdown point of the primary package.

Now, we can frame this problem as an instance of the IPAS(T) problem. The *planning* portion is defined as finding either the optimal release point that will drop package in the target zone for the case of round canopy parachute, or the optimal point and set of commands to follow a trajectory to the target zone for a parafoil. If the package drop team has a set of sensor package parachutes (or

parafoils), these can be dropped and tracked to provide an estimate on wind conditions (O'Brian, 2016). Then the IPAS(T) formulation follows the usual procedure:

1. Choose initial release point and gather sensor data
2. Update estimated wind map, and choose new release point
3. Evaluate if release point would get package into target zone with sufficient confidence
4. Release payload or send out additional sensors to get data.
5. Repeat from Step 2.

Some of the challenges in this problem include:

- Using current wind map to determine release point that will achieve target zone destination.
- Characterizing the uncertainty in reaching target zone given the uncertainty in the wind map.
- Limited ability to direct where sensors gather data.
  - Expected sensor location will differ from actual location due to remaining uncertainty in wind field. Need to consider and respect that limitation when choosing release locations and measurement fusion.
- Should sensors be dropped sequentially or in batches?

The incentive to use an IPAS(T) formulation in this scenario is driven by the very *task* oriented goal of this mission. There is a specific location to drop the package and the environment is uncertain but may have a known structure. There may be seasonal patterns to the wind distribution as well as limited wind assessment in the area such as fixed weather stations. However, to ensure a high degree of confidence in the mission, the environment should be sampled in a manner directly related to the task to provide a more up to date estimation of the wind distribution with higher resolution.

Appendix A

Wavelet Decomposition

For each $m, k \in \mathbb{Z}$, we define scalar functions $\phi_{m,k}$ and $\psi_{m,k}$ by $\phi_{m,k}(t) := \sqrt{2^m}\phi(2^m t - k)$, and $\psi_{m,k}(t) := \sqrt{2^m}\psi(2^m t - k)$. The *discrete wavelet transform* of a scalar function $f \in \mathbb{L}^2(\mathbb{R})$ is defined by $a_{m_0,k} := \langle \phi_{m_0,k}(t), f(t) \rangle$, and $d_{m,k} := \langle \psi_{m,k}(t), f(t) \rangle$, where $m_0 \in \mathbb{Z}$. The 1D *reconstruction equation* is

$$f(t) = \sum_{k=-\infty}^{\infty} a_{m_0,k}\phi_{m_0,k}(t) + \sum_{m=m_0}^{\infty}\sum_{k=-\infty}^{\infty} d_{m,k}\psi_{m,k}(t).$$

The scalars $a_{m_0,k}$ and $d_{m,k}$ are known as *approximation* and *detail* coefficients respectively. For the 2D extension of the 1D DWT, a scaling function $\Phi_{m,k,\ell}(x,y)$ and three wavelets $\Psi^1_{m,k,\ell}, \ldots, \Psi^3_{m,k,\ell}$ are defined. The *2D DWT* coefficients of a scalar function $\overline{F} \in \mathbb{L}^2(\mathbb{R}^2)$ are

$$a_{m_0,k,\ell} := \left\langle \Phi_{m_0,k,\ell}(x,y), \overline{F}(x,y) \right\rangle, \tag{A.0.1}$$

$$d^p_{m,k,\ell} := \left\langle \Psi^p_{m,k,\ell}(x,y), \overline{F}(x,y) \right\rangle, \tag{A.0.2}$$

for $p = 1, 2, 3$, $k, \ell \in \mathbb{Z}$, and $m \geqslant m_0 \in \mathbb{Z}$. The *2D reconstruction equation* is analogous to the 1D case.

An example of a pair of scaling function and wavelet is the Haar family (Daubechies, 1994). For the 1-D Haar family, the functions $\phi_{m,k}$ and $\psi_{m,k}$ are compactly supported over the interval $I_{m,k} := [2^{-m}k, 2^{-m}(k+1)]$, and by consequence, the functions $\Phi_{m,k,\ell}$ and $\Psi_{m,k,\ell}$ are compactly supported over

$$S_{m,k,\ell} \;\; := \;\; I_{m,k} \times I_{m,\ell}. \tag{A.0.3}$$

Appendix B

Machine Learning for Waiting Problem

The machine learning problem was motivated by the visible pattern of the histogram results from Section 6.2.1. It was thought that if a machine learning algorithm could take a threat field or some characteristic features of the field, it could learn to classify between waiting beneficial and waiting non beneficial fields, as shown in Fig. B.1.

This supplemental chapter will cover the data generation procedure, the different machine learning methods used including traditional and deep learning approaches, and the results of the investigations.

B.1 Data Sources

For this stage of the investigation, we can generate data arbitrarily to train and test the search and learning algorithms. Each example can be easily labeled as Go or $Wait$ based on comparing the path cost for Go and $Wait$ algorithms. We consider a threat field constructed as the weighted sum of a finite number of 2D Gaussian basis functions $c(\mathsf{x}, t) = \sum_{n=1}^{N_\mathrm{P}} w_n(t)\phi_n(\mathsf{x}, t)$. Each Gaussian basis function ϕ_n can be specified by its location, size, and intensity level. Each of these

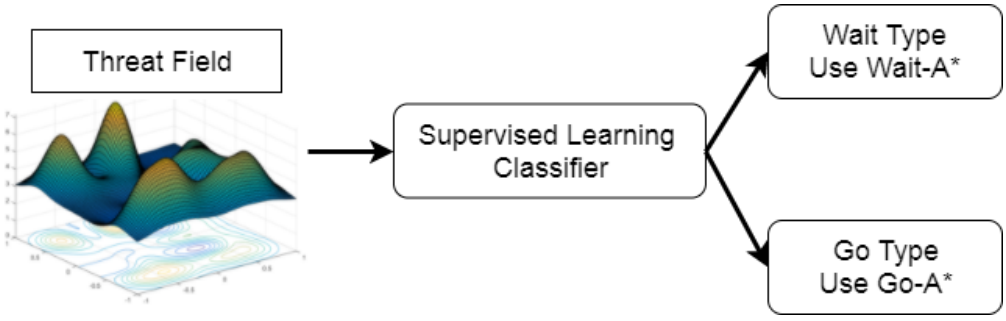


Figure B.1: A potential outcome is the use of a classifier to select the appropriate path-planner given a particular instance of the Gaussian field. The classifier will be trained on occurrences of the field labeled as Go *or* Wait

parameters can vary in time, in other words, each Gaussian can move across the *workspace* as well as grow or decay.

$$c(\mathsf{x}, t) = \sum_{n=1}^{N_{\mathrm{P}}} w_n(t)\phi_n(\mathsf{x}, t).$$

Here, the basis function ϕ_n is defined for each $n = 1, \ldots, N_{\mathrm{P}}$ by

$$\phi_n(\mathsf{x}, t) := \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-(\mathsf{x} - \mu_n(t))^{\mathrm{T}}(\mathsf{x} - \mu_n(t))}{2|\Sigma_n(t)|^2}\right),$$

where $\mu_n(t) = (\mu_{nx}(t), \mu_{ny}(t))$ defines the spatial mean of ϕ_n and $\Sigma_n(t) = (\sigma_{nx}(t), \sigma_{ny}(t))$ defines the spatial spread of ϕ_n. In this work, we consider affine functions μ_n and Σ_n of the form $\mu_n(t) = \mu_{n0} + \mu_{n1}t$, and $\Sigma_n(t) = \Sigma_{n0} + \Sigma_{n1}t$, where $\mu_{n0}, \mu_{n1}, \Sigma_{n0}, \Sigma_{n1} \in \mathbb{R}^2$ are prespecified constants.

The finite parameterization of the threat field c using Gaussian functions is justified by the fact that a large class of functions on $\mathbb{R}$, namely, square integrable functions, can be approximated with arbitrary precision by linear combinations of Gaussian functions (Calcaterra, 2008). In addition, Gaussian function appear in series solutions to several partial differential equation's such as the diffusion equation (Crank, 1979), which enables the application of the proposed work to path- and motion-planning with threat fields modeled by physical phenomena such as advection-diffusion of gases or radiation in the atmosphere (Demetriou et al., 2013).

B.2 Methods

B.2.1 Field Classification using Traditional Supervised Learning

As described in Section B.1, simulated environments were generated with various, randomly placed Gaussian threat fields throughout to collect data for supervised learning. For each environment, the A* algorithm that considers waiting and the A* algorithm that ignores waiting were both run. Afterwards, post-processing was conducted to extract features from each simulation,

standardize the features to make them suitable as input to the classification algorithms, and label the simulation as 'Wait' or 'Go'. The environment was labeled as 'Wait' if the algorithm that considered waiting produced a lower path cost than the algorithm that did not consider waiting, otherwise it was labeled as 'Go'.

Features included the number of threats in the environment, threat intensity features, threat X/Y location features, threat X/Y shape features, global threat value features, and global threat time derivative features.

As data was collected, the there was a strong imbalance in the 'Wait' vs. 'Go' cases, the classical problem of *class imbalance*. Throughout the data collection process, the percentage of simulations labeled as 'Wait', the minority class, composed anywhere from 1 to 15% of the dataset. The problem of class imbalance is not a unique phenomenon, and has been studied extensively in the medical domain for learning to produce a diagnosis, and in the industrial inspection domain for learning to identify defective products, among others (Jazi and Liu, 2017; Shenfield and Rostami, 2017).

Oversampling, undersampling, cost-sensitive learning, and the use of probability thresholds are some common approaches for addressing class imbalance (Jazi and Liu, 2017; Kotsiantis et al., 2006; Shenfield and Rostami, 2017). Oversampling involves randomly sampling with replacement from the minority class until the minority class has reached the desired size. Conversely, undersampling involves randomly selecting examples from the majority class to be removed until the majority class is reduced to the desired size. Cost-sensitive learning applies heavier penalties to the misclassification of the minority class during the training process. Finally, probability thresholds (PT) can be used to tune supervised classifiers that output the probability of an example belonging to a certain class by biasing that prediction to a particular class.

Using each of these techniques, we develop a variety of supervised classifiers to produce a performance comparison for this machine learning problem. The classifiers chosen for the performance comparison include Support Vector Machines (SVM), k-Nearest Neighbors (KNN), Multilayer Perceptron Neural Network (MLP), Random Forest (RF), and ensemble methods using the aforementioned classifiers, such as Majority Voting (MV) and Summation of Probabilities (SP) (Breiman, 2001; Chang and Lin, 2013; Omohundro, 1989; Polikar, 2006; Rojarath et al., 2016;

Rumelhart et al., 1986).

SVM with a Gaussian kernel was chosen because it is known to generally perform well with a small feature set and an intermediate-sized training dataset. KNN was selected due to its simplicity and to test the hypothesis that similar classes might have similar feature values. A neural network (MLP) was constructed because of its known ability to learn complex, non-linear models. Similarly, RF was implemented, as it is also known to classify non-linearly separable data better than other classifiers such as SVM. Finally, ensemble learning techniques such as MV and SP were employed because the combination of multiple classifiers tends to improve classification, and these methods of classifier combination were easy to implement while also being commonly used in current scientific literature.

To train and assess the individual classifiers, the dataset of features and labels was partitioned into a training dataset, composed of 80% of the examples, and a testing dataset, made up of the remaining 20% of the examples. After training the classifiers using the training dataset, each classifier was tested using the data from the testing dataset. The results were analyzed using a confusion matrix and supporting statistics. By convention, the minority class, the 'Wait'-labeled class, corresponded with a *positive* prediction, while the 'Go'-labeled class corresponded with a *negative* prediction. Therefore, using the number of true positive (TP) and negative (TN) predictions and false positive (FP) and negative (FN) predictions, supporting statistics were calculated to better inform the success in relation to path-planning in time-varying environments.

Traditional metrics of machine learning, such as Accuracy and Precision, provide general information for the comparison of algorithm performance on training and testing data. However, they do not adequately characterize classifier performance due to the inherent class imbalance, as a high accuracy rate can be achieved by classifying all examples as the majority class.

$$Accuracy\ (ACC) = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision\ (PPV) = \frac{TP}{TP + FP}$$

Instead, there are two statistics that may be of much more importance depending on the goal of the path-planning agent. For example, if the agent is undergoing a task that is time-sensitive, then they want to minimize the computation time required to find a path. In this case, they want to minimize the False Positive Rate (FPR), as this statistic indicates the percentage of cases where the computationally-expensive A* algorithm with waiting is chosen, but waiting provides no advantage in terms of path-cost.

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP + TN}$$

On the other hand, a path-planning agent may be carrying an important payload, meaning that it should avoid threat exposure to the best of its ability. Then, the agent should minimize its False Negative Rate, as this statistic indicates the percentage of cases where the A* algorithm without waiting is selected even though it produces a suboptimal path cost.

$$False\ Negative\ Rate\ (FNR) = \frac{FN}{FN + TP}$$

B.2.2 Field Classification using Deep Neural Networks

A major source of performance degradation is our lack of understanding about the relevant features that differentiate a 'Wait' field from a 'Go' field. We know that we seek to travel along minimum points in the environment. The path of travel has both spatial and temporal structure, and the previous features do not strongly consider the spatial correlations of minimum points. If we consider that the time-varying threat field is similar in concept to a video, we can draw inspiration from current video classification techniques (Jha et al., 2016; Ng; Tran et al., 2015; Wu et al., 2015).

Spatial structure can be dealt with using convolutional neural networks (CNN). This is the current favorite when performing image classification. When training a CNN over a set of images, the trained network tends to look for low level features in the bottom CNN layer such as horizontal and vertical lines and simple curves. Later layers find patterns composed of the low level features to build of shapes such as boxes or circles. The final layers are intended to match the desired

training examples such as faces, animals, foods, etc...

The temporal structure is learned by recurrent neural networks (RNN). These networks are used to learn sequences of events and are popular with text generation and prediction. But they can also be used to predict and classify videos. RNN's can be used to learn spatial features that change throughout a sequence. An example of learning patterns from a video might the classification between two people dancing and two people fighting. Recurrent neural networks often suffer from 'forgetting', and struggle to classify long term sequences. In this case, there is modified RNN architecture known as the Long-Short Term Memory (LSTM) unit. This special RNN finds a balance between short term motion and long term motion patterns.

The state of the art for video classification is to connect a two dimensional CNN to an LSTM network as seen in Figure B.2. In this architecture, the CNN learns the spatial features and the LSTM looks for those spatial features across a sequence. In the dancing versus fighting example, the CNN might find features such as arms, legs and body positions. The LSTM might learn that certain sequences of arm and/or leg placements are more likely to be dancing rather than fighting. With this neural network structure in mind, we develop a CNN-LSTM which is intended to find features identifying minimum cost paths through space and time.

Alternatively, rather than using an RNN on two dimensional CNN's, we can consider the space-time data as a volume of information. By considering the stack of frames (time being the third dimension) as a three dimensional object, we can look for 'shapes' in a volume that indicate pro-waiting environments. As in (Tran et al., 2015), we can pass this field volume into a 3D CNN and train the network using the same labels as the CNN-LSTM approach. The idea for this approach is that fields with beneficial waiting should contain vertical columns of low threat intensity in the volume.

B.2.3 Overall Optimized Path-Planner

Given the results of the classification and heuristic implementation, we can consider a general hierarchical approach to path-planning in time-varying environments. First, if the environment is small (Small Field case), we will show that the heuristic speedup makes classification unnecessary

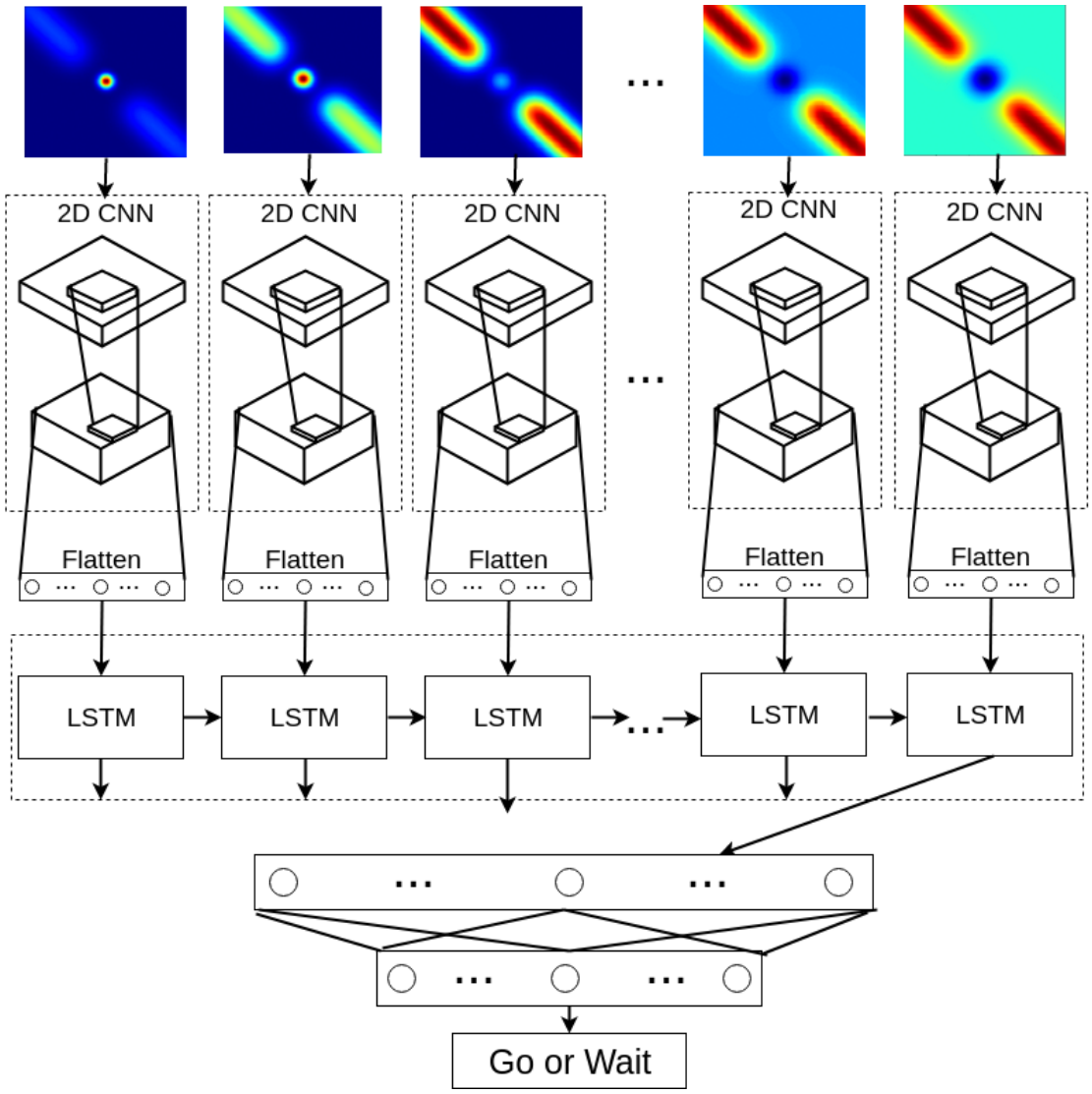


Figure B.2: A common architecture used in video classification is the CNN-LSTM consisting of a layer of two-dimensional convolutional neural networks (CNN) that feed spatial features into a layer of Long-Short Term Memory (LSTM) units that identify temporal patterns. The output of the LSTM is sent through a standard multilayer neural network for prediction.

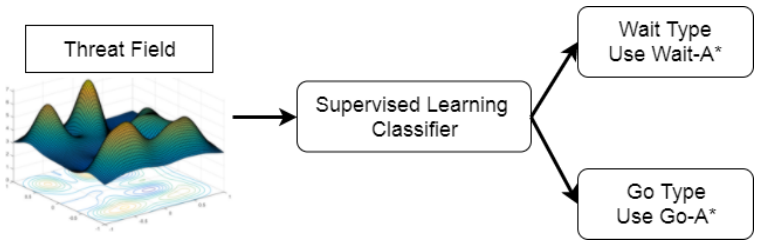


Figure B.3: A potential outcome is the use of a classifer to select the appropriate path-planner given a particular instance of the Guassian field. The classifier will be trained on occurrences of the field labeled as Go or Wait.

and it is prudent to always select the A* that always considers waiting. Second, if the performance of the heuristic does not scale as well to larger field sizes and the classification accuracy improves in future efforts, then we consider a Large Field case. Therefore if we have a Small Field case, always use Waiting A*. If we have a Large Field case, classify the field as 'Wait' or 'Go', then select the appropriate A* algorithm as in Figure B.3.

B.3 Results

B.3.1 Field Classification through Supervised Learning

SVM, RF, and KNN were the three individual classifiers that performed the best. Unfortunately, the results were not very useful for distinguishing between 'Wait' and 'Go' environments from the test set. As shown in Table B.1(a), SVM exhibited underfitting, evident by its accuracy rates of just over 50% when tested on both the training and the testing datasets. Meanwhile, RF exhibited overfitting, as shown in Table B.1(b), achieving a precision of over 70% when tested on the training dataset and a precision of just over 10% when tested on data not used to train the algorithm. KNN also exhibited overfitting, but not to the degree seen in RF, as can be seen in Table B.1(c).

In addition to traditional metrics for machine learning classifiers, there are two additional strategies for comparing classifiers when examining path-planning problems in a time-varying threat field. These strategies depend on the nature of the task at hand. For example, a path-planning agent may have precious cargo and therefore want to avoid threat exposure at all costs. In this case,

Table B.1: Results of three best algorithms for Supervised Classification.

(a) SVM exhibited underfitting.

SVM-RBF		Actual Class			
		Wait		Go	
Predicted	Wait	Train:	2067	Train:	11146
		Test:	314	Test:	2813
	Go	Train:	1031	Train:	15264
		Test:	461	Test:	3790

(b) RF exhibited overfitting.

RF		Actual Class			
		Wait		Go	
Predicted	Wait	Train:	813	Train:	305
		Test:	20	Test:	171
	Go	Train:	2285	Train:	26105
		Test:	755	Test:	6432

(c) KNN exhibited overfitting, but performed slightly better than RF on the testing dataset.

KNN		Actual Class			
		Wait		Go	
Predicted	Wait	Train:	2378	Train:	3503
		Test:	160	Test:	1271
	Go	Train:	720	Train:	22907
		Test:	615	Test:	5332

Table B.2: Results of three ensemble methods for Supervised Classification.

(a) Majority Voting with Probability Thresholds.

MV w/ PT		Actual Class	
		Wait	Go
Predicted	Wait	330	2842
	Go	445	3761

(b) Majority Voting without Probability Thresholds.

MV w/o PT		Actual Class	
		Wait	Go
Predicted	Wait	102	780
	Go	673	5823

(c) Summation of Probabilities with Probability Thresholds.

SP w/ PT		Actual Class	
		Wait	Go
Predicted	Wait	377	3356
	Go	398	3247

a threat-averse strategy would be adopted, and the False Go Rate should be minimized to reduce the probability of the path-planning agent selecting a path with a suboptimal threat exposure. On the other hand, a path-planning agent may have a mission that is time-sensitive but less sensitive to threat exposure. In this case, the agent should take a computation-averse approach, meaning that the False Wait Rate should be minimized to reduce the probability of significantly increasing computational cost for no improvement in path cost.

Despite the aggressive overfitting of the RF classifier, its FPR, just above 1% for the training dataset and 2.5% for the testing dataset, indicates that the algorithm would be beneficial for A* algorithm selection in a time-sensitive, computation-averse situation.

After combining the three best individual classifiers, SVM, RF, and KNN, using three different ensemble techniques, the classification performance on data not used to train the algorithms improved slightly but still ultimately failed to produce a robust classifier for the problem of labeling a field as suitable for waiting or not. As can be seen in Table B.2(a), Majority Voting with Probability Thresholds performed best out of the ensemble techniques when adopting a threat-averse

policy. When compared with the individual classifiers, this technique is a slight improvement over the individual classifier that performs the best on the training dataset, SVM, since the ensemble technique increases the number of unseen examples that are correctly classified as a field in which waiting is beneficial. The other two techniques for ensemble classifiers proved less effective; Table B.2(b) shows that Majority Voting without Probability Thresholds has a high false negative rate, and Table B.2(c) shows that Summation of Probabilities with Probability Thresholds exhibits underfitting with high false negative and false positive rates.

B.3.2 Classification by CNN-LSTM

As with the traditional methods mentioned previously, the CNN-LSTM is unable to obtain both good precision and good sensitivity. The classifier predicts the dominant class at all times, giving an accuracy equal to the proportion of the majority class, roughly between 85 to 95% depending on the dataset used. This trend continues with the 3D-CNN architecture.

The class imbalance problem persisted when varying the number of layers, size and number of CNN filters, as well as increasing the hidden unit count in the final fully connected layers for both CNN-LSTM and 3D-CNN. We varied the choice and parameters of the optimizer including Adam, Adadelta, and SGD with varying learning rates. All networks were trained from scratch, i.e., we are not using any pre-trained layers at this time.

B.4 Conclusions

We presented a path-planning problem in which the environment describes a time-varying environment, the threat field, inspired by natural phenomena dictated by partial differential equations. The threat field was a summation of Gaussian basis functions in which we allowed all parameters of the Gaussian basis to vary in time. We noted that the computational complexity between a path-planner that considers strictly movement of the agent and path-planner that allows for waiting at nodes differed by a factor of $\mathcal{O}(|\bar{\mathcal{G}}\mathcal{T}^2|)$, as well as providing empirical evidence of this by simulation.

In order to make the best decision for any given field we took a two pronged approach: identification of fields by machine learning, and heuristic augmentation of the search algorithm. We explored various traditional machine learning methods including, Support Vector Machines, Random Forests, k-Nearest Neighbors, and more. In general we had poor predictive performance, overfitting and underfitting all models by various degrees. Noting that the path-planning problem is inherently spatially and temporally correlated, we struggled to determine appropriate features to the field in the same manner that the search algorithm evaluates the field (the extreme of this would be reconstructing the path cost function as a feature itself, defeating the point of the classifier.) We then drew inspiration from the video classification community to implement our classification with a CNN-LSTM network.

The Convolutional Neural Network stacked with the Long-Short Term Memory units (CNN-LSTM) seems like the next logical step since our problem shares many features with image and video classification. However, our initial training results are poor. The classifier predicts the majority class (Go Field) in all cases, still suffering from the class imbalance problem. To progress further, we need to optimize the layers and hyper-parameters of the CNN-LSTM network to achieve better accuracy, an effort that remains largely an art. In addition, many of the state of the art efforts build on top of pre-trained layers that have already discovered the appropriate lower level features, such as with the Sports-1M dataset (Ng). As a final note, we may also consider our spatiotemporal dataset as a three-dimensional object where the time dimension is just another dimension. This approach is similar to 3D shape recognition and uses a 3D CNN (Tran et al., 2015). Our current results with training a 3D-CNN still fail to obtain reasonable classification accuracy. Currently, the goal for the deep learning architectures is to first obtain overfitting through adjusting the network layers (wider/deeper), ensuring that the weights are not being saturated, and different weight initialization approaches. After we can achieve good classification through overfitting, then we can employ the standard techniques of Dropout and regularization to gain good generalization of the model.

Appendix C

Detailed IPAST Algorithm

The IPAST algorithm discussed in Chapter 5 is a compressed version of the simulation code. The detailed algorithm in Fig. C.1 includes the full specifications in the algorithm including three minor "book keeping" flags: $PlanAvailable, PlanUpdateNew,$ and $IncrementedIter.$

The $PlanAvailable$ flag is only necessary to handle the first iteration and addresses when the iteration counter ℓ should be updated in Line 18. During the first iteration, $t_{\mathrm{rd}} = 0$ and won't be updated until first a plan is available, Line 8, and subsequently sensor reconfiguration has been performed. It is necessary for t_{rd} to be initialized to 0 so that Line 21 executes properly on the first iteration. Therefore, $PlanAvailable$ covers the startup period where the algorithm has yet to enter a planning or sensor reconfiguration phase.

The $PlanUpdateNew$ flag handles the period when $t_k \geqslant t_{\mathrm{pd}}$, but the algorithm has not entered the planning phase, and therefore t_{pd} is outdated and would incorrectly trigger Line 7. This would occur during the measurement collection (Δt_{c}) and sensor reconfiguration (Δt_{r}) phases. Informally, $PlanUpdateNew$ is set true in Line 24 to alert the algorithm that it should now be monitoring when $t_k \geqslant t_{\mathrm{pd}}$ because an updated plan is coming. The $PlanUpdateNew$ is set to false in Line 10 so that subsequent steps will not trigger Line 7 until a new planning phase has begun.

The $IncrementedIter$ flag accounts for when the algorithm is not in the reconfiguration phase. The logic is similar to the $PlanUpdateNew$ flag which also prevents premature execution of the planning and stop condition logic. While not in the reconfiguration phase, the value of t_{rd} is outdated and the $IncrementedIter$ flag is true indicated we already finished the previous reconfiguration phase. Once a sensor reconfiguration phase is entered, t_{rd} is set and $IncremetedIter :=$ false indicating that the algorithm is in the reconfiguration phase but has not yet finished and incremented ℓ for the current phase.

Interactive Planning and Sensing, Time-varying

Set initial sensor placement $\mathbf{s}_0 \subset \{1, \ldots, N_{\mathrm{G}}\}$.
$\hat{\Theta}_0 := \mathbf{0},\ P_0 := \lambda_0 \mathbb{I}_{(N_{\mathrm{P}})},\ \ell := 1,\ t_{\mathrm{rd}} = 0$
Set $SRflag :=$ false, $PlanFlag :=$ true.
Set $PlanFinal :=$ false, $PlanFail :=$ false.
Set $PlanAvailable :=$ false, $PlanUpdateNew :=$ false, $IncrementedIter :=$ false
for $t_k = t_0$ **to** t_f **do**
  if $t_k \geqslant t_{\mathrm{pd}}$ **and** $PlanUpdateNew$ **then**
    $\{\mathbf{v}_\ell^*, PlanFail\} \leftarrow$ Dijkstra's algorithm.
    $PlanAvailable :=$ true
    $PlanUpdateNew :=$ false
    if $\neg PlanFail$ **then**
      if $\mathrm{Var}[\hat{J}(\mathbf{v}_\ell^*)] > \varepsilon_1(\mathbf{v}_\ell^*)$ **then**
        $SRflag :=$ true
      else
        $PlanFinal :=$ true
    else
      return failure.
  if $PlanAvailable$ **and** $t_k \geqslant t_{\mathrm{rd}}$ **and** $\neg IncrementedIter$ **then**
    Set $\ell := \ell + 1$
    $IncrementedIter :=$ true
  if $PlanFlag$ **and** $\neg PlanFail$ **and** $t_k \geqslant t_{\mathrm{rd}} + \Delta t_{\mathrm{c}}$ **then**
    $PlanFlag :=$ false
    $t_{\mathrm{pd}} = t_k + \Delta t_{\mathrm{p}}$
    $PlanUpdateNew :=$ true
    Execute Dijkstra's algorithm.
  if $SRflag$ **and** $\neg PlanFail$ **and** $t_k \geqslant t_{\mathrm{pd}}$ **then**
    $SRflag :=$ false
    $IncrementedIter :=$ false
    SENSOR RECONFIGURATION
  Predict:$\{\hat{\Theta}_k^-, P_k^-\}$
  if $t_k \geqslant t_{\mathrm{rd}}$ **and** $\mod(t_k, \Delta t_{\mathrm{m}}) = 0$ **then**
    Meas. Update: $\{\hat{\Theta}_k, P_k\} \leftarrow$ by (5.5)–(5.8).
  if $PlanFinal$ **then**
    Execute $\mathbf{v}_\ell^*$ for each corresponding time step t_k.

Figure C.1: Detailed Pseudocode for Kalman filter based IPAST algorithm to solve Problem 4.

Bibliography

a.B. Philpott and a.I. Mees. Continuous-time shortest path problems with stopping and starting costs. *Applied Mathematics Letters*, 5(5):63–66, 1992. ISSN 08939659. doi: 10.1016/0893-9659(92)90066-I.

N. Adurthi. *Conjugate Unscented Transform Based Methods for Uncertainty Quantification, Nonlinear Filtering, Optimal Feedback Control and Dynamic Sensing*. PhD thesis, State University of New York at Buffalo, 2016.

N. Adurthi and P. Singla. Information driven optimal sensor control for efficient target localization and tracking. In *Proceedings of the 2014 American Control Conference*, pages 610–615, Portland, OR, USA, June 4–6 2014.

V. Akgüna, A. Parekh, R. Batta, and C. M. Rump. Routing of a hazmat truck in the presence of weather systems. *Computers & Operations Research*, 34:1351–1373, 2007.

W. H. Al-Sabban, L. F. Gonzalez, and R. N. Smith. Wind-energy based path planning for unmanned aerial vehicles using markov decision processes. *2013 IEEE International Conference on Robotics and Automation*, pages 784–789, 2013. ISSN 1050-4729. doi: 10.1109/ICRA.2013.6630662. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6630662.

A. Alessandri, M. Cuneo, S. Pagnan, and M. Sanguineti. A recursive algorithm for nonlinear least-squares problems. *Computational Optimization and Applications*, 38(2):195–216, 2007.

R. Alterovitz, T. Siméon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. *Proceedings of Robotics: Science and Systems (RSS)*, 2007.

A. B. Asghar, S. T. Jawaid, and S. L. Smith. A complete greedy algorithm for infinite-horizon sensor scheduling. *Automatica*, 81:335–341, 2017.

Y. Bar-Shalom and X.-R. Li. *Multitarget-multisensor tracking: principles and techniques*, volume 19. YBs London, UK:, 1995.

S. Behnke. Local multiresolution path planning. *Lecture Notes in Artificial Intelligence*, 3020: 332–43, 2004.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 2000.

J. F. Berube, J. Y. Potvin, and J. Vaucher. Time-dependent shortest paths through a fixed sequence of nodes: Application to a travel planning problem. *Computers and Operations Research*, 33 (6):1838–1856, 2006. ISSN 03050548. doi: 10.1016/j.cor.2004.11.021.

J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–204, 1998.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, oct 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

R. A. Brooks and T. Lozano-Pérez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15(2):224–233, Mar–Apr 1985.

F. Bullo, J. Cortés, and S. Martinez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, 2009.

X. Cai, T. Kloks, and C. K. Wong. Time-varying shortest path problems with constraints. *Networks*, 29(3):141–150, 1997.

C. Calcaterra. Linear combinations of gaussians with a single variance are dense in l2. In *Proceedings of the World Congress on Engineering*, volume 2, 2008.

C. Calcaterra and A. Boldt. Approximating with gaussians. *arXiv preprint arXiv:0805.3795*, 2008.

L. Carrioli. Unsupervised path planning of many asynchronously self-moving vehicles. In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems IROS '91*, pages 555–559, 1991.

I. Chabini. Discrete dynamic shortest path problems in transportation applications: Complexity and algorithms with optimal run time. 53(9):1689–1699, 2013. ISSN 1098-6596. doi: 10.1017/CBO9781107415324.004.

S. Chakravorty and R. Saha. Simultaneous planning localization and mapping: A hybrid bayesian/frequentist approach. *Proceedings of the American Control Conference*, pages 1226–1231, 2008. ISSN 07431619. doi: 10.1109/ACC.2008.4586660.

C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. 2013. URL https://www.csie.ntu.edu.tw/{~}cjlin/papers/libsvm.pdf.

Y. L. Chen and H. H. Yang. Finding the first K shortest paths in a time-window network. *Computers and Operations Research*, 31(4):499–513, 2004. ISSN 03050548. doi: 10.1016/S0305-0548(02)00230-7.

J. T. Cho and B. H. Nam. A study on the fuzzy control navigation and the obstacle avoidance of mobile robot using camera. In *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Conference*, volume 4, pages 2993 – 2997, 2000.

H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transaction on Robotics*, 25(4):912–926, 2009.

H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, 2005.

B. Cipra. Parlez-vous wavelets? In *What's Happening in the Mathematical Sciences*, volume 2. American Mathematical Society, 1994.

CNET Editorial Board. Best GPS systems for 2018. Online: https://www.cnet.com/topics/gps/best-gps/, Dec. 18 2017. Accessed Apr. 9, 2018.

D. Cochran and A. O. Hero. Information-driven sensor planning: Navigating a statistical manifold. *2013 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013 - Proceedings*, (0):1049–1052, 2013. doi: 10.1109/GlobalSIP.2013.6737074.

B. S. Cooper and R. V. Cowlagi. Interactive planning and sensing in uncertain spatiotemporal threat fields. *Automatica*. In review. Draft available at https://goo.gl/Uny1JC.

B. S. Cooper and R. V. Cowlagi. Interactive planning and sensing in uncertain environments with task-driven sensor placement. In *Proceedings of the 2018 American Control Conference*, Milwaukee, WI, USA., June 2018. To appear. Draft available at https://goo.gl/b3nnqG.

R. V. Cowlagi. Multiresolution path-planning with traversal costs based on time-varying spatial fields. In *Proceedings of the 53rd IEEE Conference on Decision & Control*, pages 3745–3750, Los Angeles, CA, USA, December 15–17 2014.

R. V. Cowlagi and P. Tsiotras. Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles. *IEEE Transactions on Robotics*, 28(2):379 – 395, 2012a.

R. V. Cowlagi and P. Tsiotras. Multi-resolution motion planning for autonomous agents via wavelet-based cell decompositions. *IEEE Transactions on Systems, Man and Cybernetics: Part B - Cybernetics*, 42(5):1455–1469, 2012b.

J. Crank. *The mathematics of diffusion*. Oxford university press, 1979.

I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Lecture Notes, 61, SIAM, 1994.

B. C. Dean. Algorithms for minimum-cost paths in time-dependent networks with waiting policies. *Networks*, 44:41–46, 2004a.

B. C. Dean. Shortest paths in FIFO time-dependent networks: Theory and algorithms. *Rapport technique, Massachusetts Institute of . . .* , pages 1–13, 2004b. doi: 10.1145/1113830.1113838. URL http://people.csail.mit.edu/bdean/tdsp.pdf.

M. Dell'Amico, M. Iori, and D. Pretolani. Shortest paths in piecewise continuous time-dependent networks. *Operations Research Letters*, 36(6):688–691, 2008. ISSN 01676377. doi: 10.1016/j.orl.2008.07.002. URL http://dx.doi.org/10.1016/j.orl.2008.07.002.

M. Demetriou, N. Gatsonis, and J. Court. Coupled controls-computational fluids approach for the estimation of the concentration from a moving gaseous source in a 2-d domain with a Lyapunov-

guided sensing aerial vehicle. *IEEE Transactions on Control Systems Technology*, 22(3):853–867, 2013. doi: 10.1109/TCST.2013.2267623.

M. A. Demetriou and D. Ucinski. State estimation of spatially distributed processes using mobile sensing agents. *American Control Conference (ACC)*, (January 2011):1770–1776, 2011.

M. B. Egerstedt and X. Hu. Formation constrained multi-agent control. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46 – 57, 1989.

P. Erickson, M. Cline, N. Tirpankar, and T. Henderson. Gaussian processes for multi-sensor environmental monitoring. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2015 IEEE International Conference on*, pages 208–213. IEEE, 2015.

N. Farmani, L. Sun, and D. Pack. Optimal uav sensor management and path planning for tracking. *The ASME 2014 Dynamic System and Control Conferences*, pages 1–8, 2014. doi: 10.1115/DSCC2014-6232.

E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

S. Gangulay, A. Sen, G. Xue, B. Hao, and B. Shen. Optimal routing for fast transfer of bulk data files in time-varying networks. *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, 2(c):1182–1186, 2004. ISSN 05361486. doi: 10.1109/ICC.2004.1312686.

D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington. A unified framework for the numerical solution of optimal control problems using pseudospectral methods. *Automatica*, 46:1843–1851, 2010.

Google Inc. Google maps. Online: https://maps.google.com, 2018.

A. Goshtasby and W. D. Oneill. Curve fitting by a sum of gaussians. *CVGIP: Graphical Models and Image Processing*, 56(4):281–288, 1994.

V. Gupta, T. H. Chung, B. Hassibi, and R. M. Murray. On a stochastic sensor selection algorithm with applications in sensor scheduling and sensor coverage. *Automatica*, 42(2):251–260, 2006.

J. Y. Hwang, J. S. Kim, S. S. Lim, and K. H. Park. A fast path planning by path graph optimization. *IEEE Transactions on Systems, Man, and Cybernetics– Part A: Systems and Humans*, 33(1): 121–127, January 2003.

J. Ilkyun, J. Seewong, and K. Youngouk. Mobile robot navigation using difference of wavelet SIFT. In *Proceedings of the 2009 Second International Conference on Machine Vision*, pages 286 – 292, 2009.

S. T. Jawaid and S. L. Smith. Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems. *Automatica*, 61:282–288, 2015.

A. Y. Jazi and J. J. Liu. Handling class imbalance and multiple inspection objectives in design of industrial inspection system. In *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pages 606–611. IEEE, may 2017. ISBN 978-1-5090-4397-2. doi: 10.1109/ADCONIP.2017.7983849. URL http://ieeexplore.ieee.org/document/7983849/.

D. K. Jha, A. Srivastav, and A. Ray. Temporal Learning in Video Data Using Deep Learning and Gaussian Processes. pages 1–11, 2016.

D. Jung. *Hierarchical Path Planning and Control of a Small Fixed-wing UAV: Theory and Experimental Validation*. PhD thesis, Georgia Institute of Technology, 2007.

S. Kambhampati and L. S. Davis. Multiresolution path planning for mobile robots. *IEEE Journal of Robotics and Automation*, RA-2(3):135–45, September 1986.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30:846–894, 2011.

C.-T. Kim and J.-J. Lee. Mobile robot navigation using multi-resolution electrostatic potential field. In *Proceedings of the 32nd Annual Conference of IEEE Industrial Electronics Society, 2005, IECON 2005*, 2005.

S. Kotsiantis, D. Kanellopoulos, and P. Pintelas. Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, 30, 2006. URL https://pdfs.semanticscholar.org/95df/dc02010b9c390878729f459893c2a5c0898f.pdf.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9: 235–284, 2008. ISSN 15324435. doi: 10.1145/1102351.1102385.

C. Kreucher, A. O. Hero, and K. Kastella. A comparison of task driven and information driven sensor management for target tracking. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, 2005:4004–4009, 2005. doi: 10.1109/CDC.2005.1582788.

H. Kurniawati, T. Bandyopadhyay, and N. M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, 33(3):255–272, 2012. ISSN 09295593. doi: 10.1007/s10514-012-9307-y.

J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

S. M. LaValle and J. J. Kuffner, Jr. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

R. Lerner, E. Rivlin, and I. Shimshoni. Landmark selection for task-oriented navigation. *IEEE Transactions on Robotics*, 23(3):494–505, 2007. ISSN 15523098. doi: 10.1109/TRO.2007.895070.

M. Lowe. Get real-time commute info and more in one tap. Online: https://blog.google/products/maps/get-real-time-commute-info-and-more-one-tap/, Feb. 6 2017.

W. L. D. Lui and R. Jarvis. A pure vision-based approach to topological SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3784 – 3791, Taipei, Taiwan, October 18 – 22 2010.

Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, April 2015. ISSN 1524-9050. doi: 10.1109/TITS.2014.2345663.

K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman. Decentralized environmental modeling by mobile sensor networks. *IEEE transactions on robotics*, 24(3):710–724, 2008.

R. Madankan, S. Pouget, P. Singla, M. Bursik, J. Dehn, M. Jones, A. Patra, M. Pavolonis, E. B. Pitman, T. Singh, and P. Webley. Computation of probabilistic hazard maps and source parameter estimation for volcanic ash transport and dispersion. *Journal of Computational Physics*, 271: 39–59, 2014.

S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–93, 1989.

S. Martinez. Distributed interpolation schemes for field estimation by mobile sensor networks. *IEEE Transactions on Control Systems Technology*, 18(2):491–500, 2010.

S. Martinez and F. Bullo. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, 2006.

J. N. Miller, J. C. Miller, et al. Statistics and chemometrics for analytical chemistry. 2005.

P. E. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. *Proceedings - IEEE International Conference on Robotics and Automation*, 2006:1261–1267, 2006. ISSN 10504729. doi: 10.1109/ROBOT.2006.1641882.

Y. Mo, R. Ambrosino, and B. Sinopoli. Sensor selection strategies for state estimation in energy constrained wireless sensor networks. *Automatica*, 47(7):1330–1338, 2011.

Y. Mo, E. Garone, and B. Sinopoli. On infinite-horizon sensor scheduling. *Systems & control letters*, 67:65–70, 2014.

B. Mu, L. Paull, M. Graham, J. How, and J. Leonard. Two-stage focused inference for resource-constrained collision-free navigation. *Robotics: Science and Systems*, 2015.

R. Narayanaswami and J. Pang. Multiresolution analysis as an approach for tool path planning in nc machining. *Computer-Aided Design*, 35:167–78, 2003.

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical programming*, 14(1):265–294, 1978.

J. Y.-h. Ng. Beyond Short Snippets : Deep Networks for Video Classification. ISSN 10636919. doi: 10.1109/CVPR.2015.7299101.

N. J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kauffman Publishers, Inc., San Francisco, CA, USA, 1998.

H. Noborio, T. Naniwa, and S. Arimoto. A quadtree-based path planning algorithm for a mobile robot. *Journal of Robotic Systems*, 7(4):555–74, 1990.

M. D. O'Brian. Wind assessment for aerial payload delivery systems using gps and imu sensors. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA MONTEREY United States, 2016.

R. Olfati-Saber. Distributed Kalman filter with embedded consensus filters. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, 2005:8179–8184, 2005. doi: 10.1109/CDC.2005.1583486.

R. Olfati-Saber. Kalman-Consensus filter: Optimality, stability, and performance. *Proceedings of the IEEE Conference on Decision and Control*, pages 7036–7042, 2009. ISSN 01912216. doi: 10.1109/CDC.2009.5399678.

R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

R. Olfati-Saber and N. F. Sandell. Distributed tracking in sensor networks with limited sensing range. *Proceedings of the American Control Conference*, pages 3157–3162, 2008. ISSN 07431619. doi: 10.1109/ACC.2008.4586978.

R. Olfati-Saber and J. Shamma. Consensus Filters for Sensor Networks and Distributed Sensor Fusion. *Proceedings of the 44th IEEE Conference on Decision and Control*, (0):6698–6703, 2005. doi: 10.1109/CDC.2005.1583238. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1583238.

S. M. Omohundro. Five Balltree Construction Algorithms. 1989. URL http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=DEA1784A46135C8E544AE806965AA22A?doi=10.1.1.91.8209{&}rep=rep1{&}type=pdf.

A. Orda and R. Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990. ISSN 00045411. doi: 10.1145/79147.214078.

A. Orda and R. Rom. Minimum-weight paths in time-dependent networks. *Networks*, 21:295–319, 1991.

S. L. Padula and R. K. Kincaid. Optimization strategies actuator placement sensor and actuator placement. *NASA Report*, (April), 1999.

D. K. Pai and L.-M. Reissell. Multiresolution rough terrain motion planning. *IEEE Transactions on Robotics and Automation*, 14(1):19–33, February 1998.

S. Pallottino. Shortest path algorithms in transportation models: classical and innovative aspects. *Equilibrium and advanced transportation*, 13(1):245–281, 1998. ISSN 0254-5330. doi: 10.1007/BF02288320. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.5192$\delimiter"026E30F$nhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.4398{&}rep=rep1{&}type=pdf.

C. Paulson, S. Ezekiel, and D. Wu. Wavelet-based image registration. In T. H. O'Donnel, M. Blowers, and K. Priddy, editors, *Evolutionary and Bio-Inspired Computation: Theory and Applications IV, Proceedings of the SPIE*, volume 7704, 2010.

A. Peters, S. von Klot, M. Heier, I. Trentinaglia, A. Hrmann, H. E. Wichmann, and H. Lwel. Exposure to traffic and the onset of myocardial infarction. *New England Journal of Medicine*, 351(17):1721–1730, 2004. doi: 10.1056/NEJMoa040203. URL https://doi.org/10.1056/NEJMoa040203. PMID: 15496621.

A. B. Philpott and A. I. Mees. A finite-time algorithm for shortest path problems with time-varying costs. *Applied Mathemtical Letters*, 6(2):91–94, 1993.

E. Plaku, L. E. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Transactions on Robotics*, 26(3):469–482, 2010.

R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6 (3):21–45, 2006. ISSN 1531-636X. doi: 10.1109/MCAS.2006.1688199. URL http://ieeexplore.ieee.org/document/1688199/.

R. J. Prazenica, A. J. Kurdila, R. C. Sharpley, and J. Evers. Multiresolution and adaptive path planning for maneuver of micro-air-vehicles in urban environments. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pages 1–12, San Francisco, CA, 2005.

R. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.

O. Raaschou-Nielsen, Z. J. Andersen, M. Hvidberg, S. S. Jensen, M. Ketzel, M. Srensen, S. Loft, K. Overvad, and A. Tjnneland. Lung cancer incidence and long-term exposure to air pollution from traffic. *Environmental Health Perspectives*, 119(6):860–865, 2011.

J. Ranieri, A. Chebira, and M. Vetterli. Near-Optimal Sensor Placement for Linear. 62(5):1135–1146, 2014.

R. M. Rao and A. S. Bopardikar. *Wavelet Transforms - Introduction to Theory and Applications*. Addison-Wesley, 1998. ISBN 0-201-63463-5.

D. Reid et al. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, 24(6):843–854, 1979.

M. B. Rice, P. L. Ljungman, E. H. Wilker, K. S. Dorans, D. R. Gold, J. Schwartz, P. Koutrakis, G. R. Washko, G. T. OConnor, and M. A. Mittleman. Long-term exposure to traffic emissions and fine particulate matter and lung function decline in the framingham heart study. *American journal of respiratory and critical care medicine*, 191(6):656–664, 2015.

A. Rojarath, W. Songpan, and C. Pong-inwong. Improved ensemble learning for classification techniques based on majority voting. In *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 107–110. IEEE, aug 2016. ISBN 978-1-4673-9904-3. doi: 10.1109/ICSESS.2016.7883026. URL http://ieeexplore.ieee.org/document/7883026/.

T. Routtenberg and L. Tong. Estimation after Parameter Selection: Performance Analysis and Estimation Methods. *IEEE Transactions on Signal Processing*, 64(20):1–13, 2015. ISSN 1053587X. doi: 10.1109/TSP.2016.2580533. URL http://arxiv.org/abs/1503.02045.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, oct 1986. URL http://dx.doi.org/10.1038/323533a0http://10.0.4.14/323533a0.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Inc., NJ, USA, 2003.

H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(2):187–260, June 1984.

A. Shenfield and S. Rostami. Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. In *2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8. IEEE, aug 2017. ISBN 978-1-4673-8988-4. doi: 10.1109/CIBCB.2017.8058553. URL http://ieeexplore.ieee.org/document/8058553/.

D. Shi and T. Chen. Approximate optimal periodic scheduling of multiple sensors with constraints. *Automatica*, 49(4):993–1000, 2013.

M. Shim, J. Kurtz, and A. Laine. Multi-resolution stereo algorithm via wavelet representations for autonomous navigation. In *Proceedings of the SPIE*, volume 3723, pages 319 – 328, Orlando, FL, April 1999.

R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, Cambridge, MA, USA., 2011.

B. Sinopoli, M. Micheli, G. Donato, and T. J. Koo. Vision based navigation for an unmanned aerial vehicle. In *Proceedings of the 2001 IEEE Conference on Robotics and Automation*, pages 1757–64, 2001.

P. Skoglar, J. Nygards, and M. Ulvklo. Concurrent path and sensor planning for a uav - towards an information based approach incorporating models of environment and sensor. *IEEE International Conference on Intelligent Robots and Systems*, pages 2436–2442, 2006. doi: 10.1109/IROS.2006.281685.

R. F. Stengel. *Optimal Control and Estimation*. Dover, New York, NY, 1994.

A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3310 – 3317, 1994.

A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 95, pages 1652–1659, 1995.

K. Sung, M. G. H. Bell, M. Seong, and S. Park. Shortest paths in a network with time-dependent flow speeds. *European Journal of Operational Research*, 121(1):32–39, 2000. ISSN 03772217. doi: 10.1016/S0377-2217(99)00035-1.

H. Tan, Y. Wu, B. Shen, P. J. Jin, and B. Ran. Short-term traffic prediction based on dynamic tensor completion. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2123–2133, Aug 2016. ISSN 1524-9050. doi: 10.1109/TITS.2015.2513411.

R. F. Thomas and J. Scotto. Estimating increases in skin cancer morbidity due to increases in ultraviolet radiation exposure. *Cancer Investigation*, 1(2):119–126, 1983. doi: 10.3109/07357908309042414. URL http://dx.doi.org/10.3109/07357908309042414. PMID: 6667401.

D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 International Conference on Computer Vision, ICCV 2015:4489–4497, 2015. ISSN 15505499. doi: 10.1109/ICCV.2015.510.

P. Tsiotras and E. Bakolas. A hierarchical on-line path planning scheme using wavelets. In *Proceedings of the European Control Conference*, pages 2806–2812, Kos, Greece, July 2–5 2007.

V. Tzoumas, A. Jadbabaie, and G. J. Pappas. Sensor placement for optimal kalman filtering: Fundamental limits, submodularity, and algorithms. In *American Control Conference (ACC), 2016*, pages 191–196. IEEE, 2016.

V. Tzoumas, L. Carlone, G. J. Pappas, and A. Jadbabaie. Sensing-constrained lqg control. In *2018 Annual American Control Conference (ACC)*, pages 197–202. IEEE, 2018.

D. Ucinski. Sensor network scheduling for identification of spatially distributed processes. *Conference on Control and Fault-Tolerant Systems, SysTol'10 - Final Program and Book of Abstracts*, 20(3):493–504, 2010. ISSN 1641-876X. doi: 10.1109/SYSTOL.2010.5675945.

B. J. H. Verwer. A multiresolution workspace, multiresolution configuration space approach to solve the path planning problem. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 2107–12, 1990.

L. Wei and T. F. Fwa. Characterizing road roughness by wavelet transform. *Transportation Research Record: Journal of the Transportation Research Board*, 1869:152 – 158, 2004.

T. Wiesemann, J. Schiefele, and J. Bader. Multi-resolution terrain depiction and airport navigation function on an embedded SVS. In J. G. Verly, editor, *Enhanced and Synthetic Vision 2002, Proceedings of the SPIE*, volume 4713, pages 106 – 117, 2002.

C. K. Williams and C. E. Rasmussen. Gaussian processes for machine learning. *the MIT Press*, 2 (3):4, 2006.

Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Modeling Spatial-Temporal Clues in a Hybrid Deep Learning Framework for Video Classification. 2015. doi: 10.1145/2733373.2806222. URL http://arxiv.org/abs/1504.01561.

L. Xie, D. Popa, and F. L. Lewis. *Optimal and robust estimation: with an introduction to stochastic control theory*. CRC press, 2007.

B. Xu, D. J. Stilwell, and A. Kurdila. A receding horizon controller for motion planning in the presence of moving obstacles. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 974 – 979, Anchorage, AK, May 3 – 8 2010.

L. Ye, S. Roy, and S. Sundaram. On the complexity and approximability of optimal sensor selection for kalman filtering. In *2018 Annual American Control Conference (ACC)*, pages 5049–5054. IEEE, 2018.

M. Yguel, O. Aycard, and C. Laugier. Wavelet occupancy grids: A method for compact map building. In P. Corke and S. Sukkarieh, editors, *Field and Service Robotics, STAR 25*, pages 219 – 230. Springer-Verlag, 2006.

H. Zhang, R. Ayoub, and S. Sundaram. Sensor selection for kalman filtering of linear dynamical systems: Complexity, limitations and greedy algorithms. *Automatica*, 78:202–210, 2017.